

# **Tree testing for Websites**

**A free comprehensive guide for evaluating site structures**

**Last updated: 23 October 2016**

1. Home	5
1.1 1 - Getting our users unlost	8
1.1.1 Two designers, two approaches	9
1.1.2 Good IA starts with an effective site tree	14
1.1.3 Why search is not enough	15
1.1.4 Chapter 1 - key points	16
1.2 2 - Tree testing - a quick intro	17
1.2.1 What is tree testing?	18
1.2.2 Why run a tree test?	22
1.2.3 When should we do a tree test?	23
1.2.4 How long will it take?	24
1.2.5 Which types of sites are most suitable?	25
1.2.6 What are the basic steps?	27
1.2.7 Chapter 2 - key points	29
1.3 3 - IA in the design process	30
1.3.1 How does IA fit into design?	31
1.3.2 How does tree testing fit into design?	32
1.3.3 The research phase	33
1.3.4 The design phase: creating new trees	37
1.3.5 The design phase: going wide	38
1.3.6 The design phase: going deep	42
1.3.7 Putting it all together	43
1.3.8 Comparing to other IA methods	45
1.3.9 Chapter 3 - key points	47
1.4 4 - Planning a tree test	48
1.4.1 Why are we running this test?	49
1.4.2 How many rounds of testing?	51
1.4.3 Which trees will we test?	52
1.4.4 Who will we test?	53
1.4.5 When will we test?	54
1.4.6 Which tool will we use?	55
1.4.7 Where will we test?	57
1.4.8 Who will do what?	58
1.4.9 How will we handle problems?	59
1.4.10 Documenting our plan	60
1.4.11 Chapter 4 - key points	61
1.5 5 - Creating trees	62
1.5.1 Basing new trees on research	64
1.5.2 Common schemes to organize sites	65
1.5.3 Combining and flipping schemes	68
1.5.4 Wide/shallow vs. narrow/deep	70
1.5.5 Labelling and terminology	72
1.5.6 Team-sourcing ideas	76
1.5.7 Roughing out alternative trees	78
1.5.8 Picking candidates to test	79
1.5.9 Posing questions about tree elements	81
1.5.10 More on creating trees	82
1.5.11 Chapter 5 - key points	83
1.6 6 - Preparing a tree for testing	84
1.6.1 Working in an electronic format	85
1.6.2 Which part of the tree?	89
1.6.3 Which headings to include/exclude?	93
1.6.4 Spotting missing content	99
1.6.5 Dealing with shortcuts and duplicated content	101
1.6.6 Breaking up double-level topics	103
1.6.7 Using link names instead of page titles	105
1.6.8 What to call "Home"	106
1.6.9 Transferring the tree to a testing app	107
1.6.10 Chapter 6 - key points	110
1.7 7 - Writing tasks	111
1.7.1 Which tasks to include?	112
1.7.2 How many tasks?	115
1.7.3 Mapping tasks to the tree	117
1.7.4 Different tasks for different user groups	120
1.7.5 Collaborating on tasks	123
1.7.6 Writing a good task	124
1.7.7 Identifying correct answers	130
1.7.8 Entering tasks and their answers	135
1.7.9 Randomizing the order of tasks	136
1.7.10 Letting participants skip tasks	138
1.7.11 Asking questions after a task	139
1.7.12 Chapter 7 - key points	140

1.8 8 - Setting up a test	141
1.8.1 Naming the test	142
1.8.2 Disguising the test address	143
1.8.3 Selecting languages	144
1.8.4 Password-protecting the test	145
1.8.5 Setting closing rules	146
1.8.6 Redirecting after the test	147
1.8.7 Setting up the tree and tasks	148
1.8.8 Writing supporting text	149
1.8.9 Adding survey questions	151
1.8.10 Choosing a visual look	155
1.8.11 Providing a support contact	156
1.8.12 Alerting the organization about our study	157
1.8.13 Chapter 8 - key points	158
1.9 9 - Recruiting participants	159
1.9.1 How many participants?	160
1.9.2 Different user groups	162
1.9.3 Using web ads	164
1.9.4 Using email lists	168
1.9.5 Using social media	171
1.9.6 Using commercial panels	173
1.9.7 Using integrated recruitment tools	175
1.9.8 Other ways to recruit	176
1.9.9 Dealing with selection bias	177
1.9.10 Coordinating audiences and channels	179
1.9.11 Screening for specific participants	181
1.9.12 Restricting access with a password	183
1.9.13 Writing a good invitation	184
1.9.14 Offering incentives	186
1.9.15 Recruiting for in-person sessions	188
1.9.16 Chapter 9 - key points	189
1.10 10 - Piloting the test	190
1.10.1 Trying out the task wording	191
1.10.2 Previewing a test	192
1.10.3 Running a pilot test	193
1.10.4 Checking for technical problems	194
1.10.5 Revising the test	195
1.10.6 Chapter 10 - key points	196
1.11 11 - Running the test	197
1.11.1 Splitting users randomly among tests	198
1.11.2 Launching the test(s)	200
1.11.3 Monitoring the test's progress	201
1.11.4 Keeping stakeholders informed	203
1.11.5 Asking "why?" with in-person sessions	204
1.11.6 Closing the test	205
1.11.7 Chapter 11 - key points	206
1.12 12 - Analyzing results	207
1.12.1 Cleaning the data	208
1.12.2 Sharing the data	211
1.12.3 Reviewing overall results	212
1.12.4 Analyzing by task	216
1.12.4.1 Task success rate	217
1.12.4.2 Where they went	218
1.12.4.3 What they clicked first	226
1.12.4.4 Directness – where they backtracked	228
1.12.4.5 Task speed - where they slowed down	232
1.12.4.6 Where they gave up	234
1.12.4.7 Confidence	236
1.12.4.8 Dealing with outliers	237
1.12.4.9 Finding patterns among tasks	239
1.12.5 Analyzing by branches	240
1.12.6 Analyzing by user group or other criteria	242
1.12.7 Discovering evil attractors	244
1.12.8 Chapter 12 - key points	247
1.13 13 - Communicating results	248
1.13.1 Recording findings	249
1.13.2 Turning findings into actions	254
1.13.3 Summing up the basics	255
1.13.4 Reporting in more depth	258
1.13.5 Passing along participant feedback	260
1.13.6 Chapter 13 - key points	261
1.14 14 - Revising and retesting	262

1.14.1	Revising trees	263
1.14.2	Cherrypicking and hybrid trees	265
1.14.3	Rewording and replacing tasks	267
1.14.4	Tuning survey questions	269
1.14.5	Using fresh participants	270
1.14.6	When are we done?	271
1.14.7	Chapter 14 - key points	272
1.15	15 - Special considerations	273
1.15.1	Testing trees for mobile apps	274
1.15.2	Multi-language testing	275
1.15.3	Tree testing in Agile/Lean projects	276
1.15.4	Tree testing on paper	277
1.15.5	Chapter 15 - key points	278
1.16	16 - Beyond tree testing	279
1.16.1	Testing navigation	280
1.16.2	Testing search	283
1.16.3	Testing visual design	284
1.16.4	Testing content	285
1.16.5	Testing overall usability	286
1.16.6	Chapter 16 - key points	287
1.17	List of all templates	288
1.18	About this guide	289

# Home



***"What is tree testing?"***



***"How does it improve my website?"***



***"How do I run a good tree test?"***

***Tree Testing for Websites is a community guide to the "tree test" method - a quick, quantitative way to make things easier to find on your website.***

- If you're just getting started, this guide offers step-by-step instructions for running a good tree test and analyzing the results.
- If you have previously run tree tests, you'll find tips on how to run more advanced (and more effective) tests, and solutions to the most common mistakes that people make.
- If you have additional tree-testing wisdom to share, become a contributor to this guide!

## 1 - Getting our users unlost

- Two designers, two approaches
- Good IA starts with an effective site tree
- Why search is not enough
- Chapter 1 - key points

## 2 - Tree testing - a quick intro

- What is tree testing?
- Why run a tree test?
- When should we do a tree test?
- How long will it take?
- Which types of sites are most suitable?
- What are the basic steps?
- Chapter 2 - key points

## 3 - IA in the design process

- How does IA fit into design?
- How does tree testing fit into design?
- The research phase
- The design phase: creating new trees
- The design phase: going wide
- The design phase: going deep
- Putting it all together
- Comparing to other IA methods
- Chapter 3 - key points

## 4 - Planning a tree test

- Why are we running this test?
- How many rounds of testing?
- Which trees will we test?
- Who will we test?
- When will we test?
- Which tool will we use?
- Where will we test?
- Who will do what?

## 9 - Recruiting participants

- How many participants?
- Different user groups
- Using web ads
- Using email lists
- Using social media
- Using commercial panels
- Using integrated recruitment tools
- Other ways to recruit
- Dealing with selection bias
- Coordinating audiences and channels
- Screening for specific participants
- Restricting access with a password
- Writing a good invitation
- Offering incentives
- Recruiting for in-person sessions
- Chapter 9 - key points

## 10 - Piloting the test

- Trying out the task wording
- Previewing a test
- Running a pilot test
- Checking for technical problems
- Revising the test
- Chapter 10 - key points

## 11 - Running the test

- Splitting users randomly among tests
- Launching the test(s)
- Monitoring the test's progress
- Keeping stakeholders informed
- Asking "why?" with in-person sessions
- Closing the test
- Chapter 11 - key points

## 12 - Analyzing results

- How will we handle problems?
- Documenting our plan
- Chapter 4 - key points

## 5 - Creating trees

- Basing new trees on research
- Common schemes to organize sites
- Combining and flipping schemes
- Wide/shallow vs. narrow/deep
- Labelling and terminology
- Team-sourcing ideas
- Roughing out alternative trees
- Picking candidates to test
- Posing questions about tree elements
- More on creating trees
- Chapter 5 - key points

## 6 - Preparing a tree for testing

- Working in an electronic format
- Which part of the tree?
- Which headings to include/exclude?
- Spotting missing content
- Dealing with shortcuts and duplicated content
- Breaking up double-level topics
- Using link names instead of page titles
- What to call "Home"
- Transferring the tree to a testing app
- Chapter 6 - key points

## 7 - Writing tasks

- Which tasks to include?
- How many tasks?
- Mapping tasks to the tree
- Different tasks for different user groups
- Collaborating on tasks
- Writing a good task
- Identifying correct answers
- Entering tasks and their answers
- Randomizing the order of tasks
- Letting participants skip tasks
- Asking questions after a task
- Chapter 7 - key points

## 8 - Setting up a test

- Naming the test
- Disguising the test address
- Selecting languages
- Password-protecting the test
- Setting closing rules
- Redirecting after the test
- Setting up the tree and tasks
- Writing supporting text
- Adding survey questions
- Choosing a visual look
- Providing a support contact
- Alerting the organization about our study
- Chapter 8 - key points

- Cleaning the data
- Sharing the data
- Reviewing overall results
- Analyzing by task
- Analyzing by branches
- Analyzing by user group or other criteria
- Discovering evil attractors
- Chapter 12 - key points

## 13 - Communicating results

- Recording findings
- Turning findings into actions
- Summing up the basics
- Reporting in more depth
- Passing along participant feedback
- Chapter 13 - key points

## 14 - Revising and retesting

- Revising trees
- Cherry-picking and hybrid trees
- Rewording and replacing tasks
- Tuning survey questions
- Using fresh participants
- When are we done?
- Chapter 14 - key points

## 15 - Special considerations

- Testing trees for mobile apps
- Multi-language testing
- Tree testing in Agile/Lean projects
- Tree testing on paper
- Chapter 15 - key points

## 16 - Beyond tree testing

- Testing navigation
- Testing search
- Testing visual design
- Testing content
- Testing overall usability
- Chapter 16 - key points

## List of all templates

[About this guide](#)

[Download PDF](#)

---

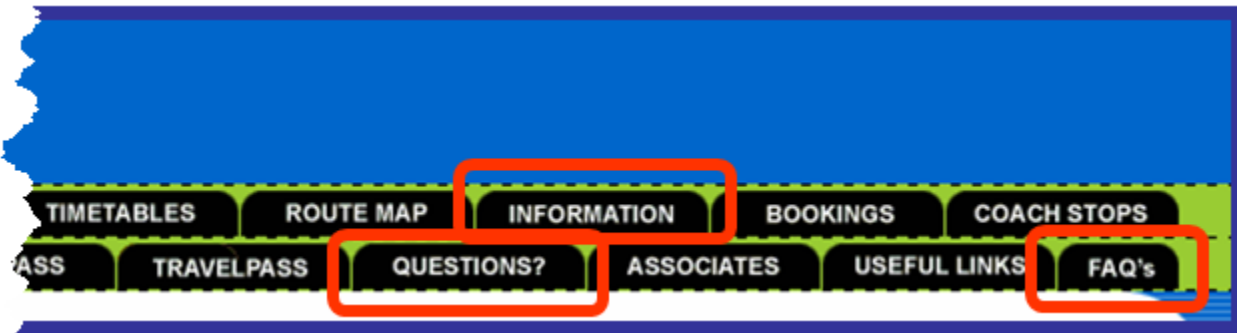
# 1 - Getting our users unlost

*"I've never been lost, but I was mighty turned around for three days once." Daniel Boone*

Few things are as frustrating as not being able to find something, especially **when we know it's there**.

As website creators, we put a lot of effort into our content, but **it all comes to naught if our users can't locate what they're looking for**.

Everyone has visited sites that were poorly organized, and that wasted their time as a result. Suppose we need some help from this bus website - should we visit the *Questions* section or the *FAQ*? And just what does that mysterious *Information* tab offer?



When other sites are just a click away, we need to make sure the sites that we design **get people to the content they want, quickly and easily**.

**We make sure by testing.**

---

## Two designers, two approaches

How test-driven site design puts us at less risk than "genius" design

## Good IA starts with an effective site tree

Getting users to the right spot the first time, with a safety net just in case

## Why search is not enough

Browsing headings is still crucial to our users finding the right content

## Key points

---

**Next:** [Chapter 2 - Tree testing - a quick intro](#)



## Two designers, two approaches

- Tom and the genius method
- Maria and the empirical method
- The moral of our story

In our experience, there are two basic ways that designers approach the creation of a website:

- The **genius method**  
“I know exactly what to design, so I'll just get on with it.”
- The **empirical method**  
“I know roughly where to start, so I'll design, test, and then revise as needed.”

To illustrate this, let's imagine two designers, **Tom** and **Maria**, who are both asked to design a medium-size content website for cycling – say about 300 pages of well-designed, clearly written information.

- **Tom is from the school of genius design.** He believes that he creates great websites because he is a talented designer and his past work was praised.
- **Maria is from the school of empirical design.** She believes she is a good designer, but she knows that she will not get everything right the first time, so she uses a variety of research tools to inform (and correct) her designs as she goes.

### Tom and the genius method

Having studied the content and talked to the site's internal stakeholders, Tom thinks a lot, then opens a new spreadsheet and creates a text tree of possible headings and subheadings, **based on different audiences**:

/			
8	BikeSite		
9		For commuters	
0			Bike routes
1			Skills training
2			Cycling and the law
3			Tips for commuting
4		For families	
5			Places to cycle
6			Learning to ride
7			Safety tips
8		For roadies	
9			etc.
0		For mountain bikers	
1			etc.
2		For cycle tourers	
3		News & events	
4			

When he reviews his tree ideas with others on the team, most of them like the audience-based scheme. They make some comments and suggestions, and he accordingly makes a few revisions before delivering the design.

Everyone is happy until several weeks after the site is released. The analytics show that certain parts of the site that expected heavy traffic are getting very little, while the web-feedback channel is choked with users complaining they can't find this or that on the new site. (They love the new look and feel, but it's much harder to **find** things now that everything has changed.) This feedback persists for several months.

*"Way too much work to find what I want. Maps used to be all in one place, but now they're scattered all over the place." - a disgruntled site visitor*

Eventually, a consultant is brought in to do some usability testing on the site and recommend fixes. One of the major issues she finds is that the site is organized in a way that makes sense to the project team, but not to some of their major audiences. Some of this can be fixed with simple terminology changes, but some of it will require fundamental changes to the site structure, which will in turn require a good deal of content rework.

No one (including Tom) wanted this result, but now that they know about it, it can be fixed in release 2 (if they can get funding for it).

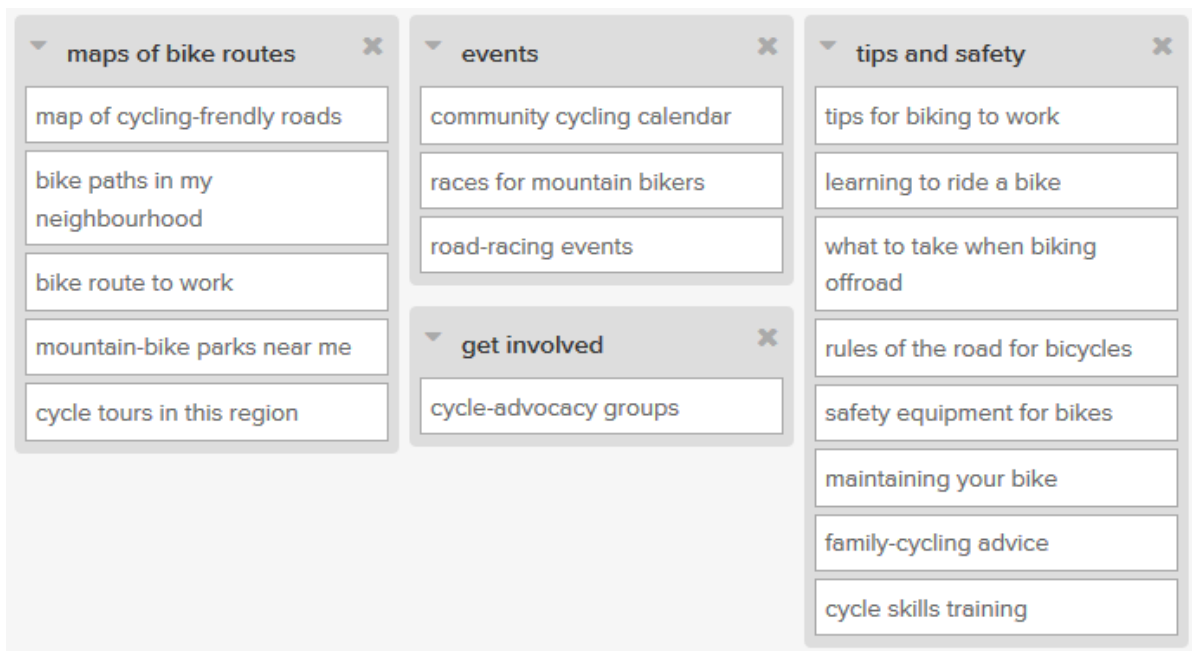
*"We were hoping to add a stolen-bike registry next, but that will have to wait until we solve some basic navigation issues." - site owner*

## Maria and the empirical method

Like Tom, Maria starts by studying the content and talking to the site's internal stakeholders. And, like Tom, her initial instinct is an audience-based scheme.

She **asks about any existing user research**, and receives the results of a site survey done the previous year, which suggests some new content but doesn't give any clues to better ways to organize the site.

She has some ideas on how to structure the content, but she also knows that **she is not the target audience**, so she decides that she needs to get some user input to help generate structure ideas. She **runs an open card sort** online using some representative content, and discovers that most users actually organize the cards according to activity, not according to audience. **Her initial hunch was wrong, but it's easy to change direction this early in the game.**



She then opens a new spreadsheet and creates a text tree of possible headings and subheadings, **based on the various types of topics** that the site offers:

15			
16	BikeSite		
17		Maps and routes	
18			Routes for commuting
19			Routes for touring
20			Routes for road training
21			MTB trails
22		The law and your safety	
23			Cycling and the law
24			Cycle safety
25			Cycle skills training
26		Cycling tips	
27			General tips
28			Commuting tips
29			Racing/training tips
30			MTB tips
31			Touring tips
32		Events calendar	
33			

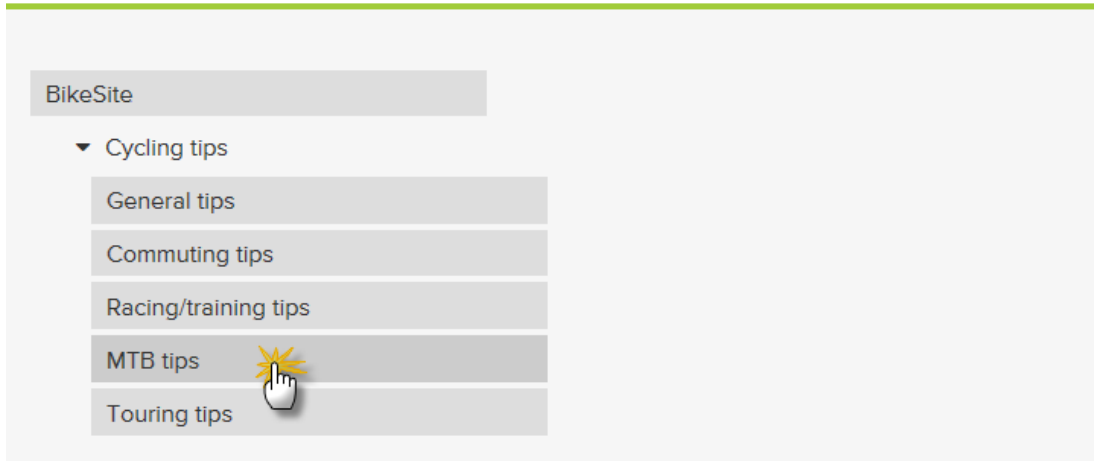
Instead of finalizing the structure then and there, **she wants some proof** that a topic-based tree is better than her initial idea of an audience-based tree. So she creates an audience-based tree as well, and decides to test them against each other. (The project sponsor happens to prefer the audience scheme, so this is an additional reason to get objective data on both before deciding.)

7			
8	BikeSite		
9		For commuters	
0			Bike routes
1			Skills training
2			Cycling and the law
3			Tips for commuting
4		For families	
5			Places to cycle
6			Learning to ride
7			Safety tips
8		For roadies	
9			etc.
0		For mountain bikers	
1			etc.
2		For cycle tourers	
3		News & events	
4			

**Maria spends a week running side-by-side tree tests** (one for each tree), and gets about 100 people to try out each tree:

**Task 1 of 1**[Skip this task](#)

You want to know what to take when you go biking offroad.



The results are revealing – **the topic-based tree performs much better**, except for a few tasks where the audience-based tree wins:

Metric	Topic tree	Audience tree
Success rate	72%	55%
Directness	77%	64%
Average speed	14 secs	18 secs

Maria reviews the results of the test with the team, and they agree to go with the second tree, but incorporating some elements of the first tree that worked particularly well. This is the structure they build the site with, and **when they run a usability test on the alpha version**, it performs well except for a few minor changes that they can make before the site is released.

Once the site is live, the analytics show the expected areas of traffic. A few users complain about not being able to find things because they've been moved, but **these complaints dwindle after the first month** as users become familiar with the new structure.

*"Took a bit to get used to, but it's now much easier to find what I want." - a satisfied site visitor*

Maria, her project team, and upper management are all happy with the new site:

*"Great feedback from users, and traffic has jumped too. The committee was happy to give us the go-ahead for the exciting stuff in release 2." - site owner*

## The moral of our story

Don't dismiss this as a straw-man example; over the years we've seen an astonishing number of websites are created using Tom's single-design genius method (or something very close to it). The Toms of the design world may be talented, but they are often **curiously reluctant to use empirical tools to test their designs** before the website ships. And the odds of them getting everything right the first time are very low. That translates into a high risk for the organization.

Geniuses do have great ideas (that's why they're geniuses) but **they also tend to think that all their ideas are good**. Geniuses are also hard to find, sometimes hard to work with, and usually expensive. And they don't come with guarantees.

On the other hand, designers like Maria who use empirical tools to improve their work lower the risk of delivering mistakes to users. By **testing their ideas before the website is finished** (or even before it's started), they can see what works and what doesn't for the intended audiences. Instead of doing "genius design", they're doing "user-centered" design.

We don't have to be geniuses to recommend the latter approach. 😊

**Next:** [Good IA starts with an effective site tree](#)

## Good IA starts with an effective site tree

- [Finding the right content the first time](#)
  - [Providing a safety net](#)
- 

When users visit our website, whether they arrive at the home page or a deeper page, they usually skim the main content first, in the top center of the page. After all, a certain piece of content is really what they're after.

However, they are also starting to build a **mental model** of our site. This model includes:

- The **layout of items** on the page (header, sidebars, comments, etc.), and
- The **shape and scope of the site itself** (breadcrumb, main navigation, utility links, footer, etc.)

### Finding the right content the first time

If the content satisfied them, they may leave (which is great – we answered their question instantly – well done!).

Or they may decide that this site is pretty good and see what else we have that they want, in which case they'll need a way to see **where they are, what's available, and how to get there**.

A **clearly visible main navigation**, showing the top-level sections of our site and where the user currently is, **is what our visitor is looking for and expecting**.

### Providing a safety net

If the initial page did **not** satisfy our visitor, they're likely to ask themselves **two very quick questions**:

- **Am I at the right site?**  
When we search the web, we get all kinds of hits, but many of them are a waste of time. As a result, we've gotten very good at separating the wheat from the chaff.  
When we visit that first page of a site, we're instantly looking around to see if this is the kind of site that has what we want, or if we should just click Back now and cut our losses. Seeing a clear navigation tree and specific terms helps our visitor make this decision.
    - **If this is the right site** for them, we provide them ways to get to the right content, even though the first page they saw wasn't the correct one.
    - **If this isn't the right site** for them, a clear navigation tree will make it clear to them that they should leave now (painlessly) instead of wasting their time hunting around our site looking for content it doesn't have. It's the polite thing to do.
  - **Where in this site should I go?**  
If they decide to delve further into our site, we need to give them an idea of what we cover, what we don't, and where to go next. If they're looking for content related to the page they stumbled upon, a "related pages" box on the page can help. But it's also helpful for them to be able to see the overall structure of the site, and where they are in it, so they can orient themselves and make an informed choice about where to click next.
- 

**Next:** [Why search is not enough](#)

## Why search is not enough

---

*“Do we really need to spend time designing a good site tree? Doesn't everybody just use Search now?”*

Search is huge these days, and for good reason. If we know what we're looking for, search engines like Google are very good at finding it. And we expect every site of even medium size to have a big friendly search box waiting for us.

But for most websites, a **good search function is not enough**.

Here's why:

- **Most users start by browsing.**  
According to **Jeff Sauro** (a usability guru known for his rigorous quantitative approach), a [meta-analysis](#) of studies of 1500 users showed that “on average, about 14% of users started with search”. Other studies vary in their estimates of browsing vs. searching, but it's clear that browsing is too common a behavior to ignore.
- **Browsing is even more likely if the navigation looks promising.**  
Users make snap judgments when they visit a site. As **Alan McFarland** describes in this [short article](#), if the site navigation seems to be clear, distinguishable, and offer a good range of choices, visitors are more likely to start with browsing.
- **Search only works well if we know what we're looking for.**  
As **Laura Larsell** succinctly states in [this Mashable article](#), “Search assumes a direct path between the seeker and the sought. Ironically, ‘search’ works best when you have a pretty good sense of what you are looking for. But most people, most of the time, do not have concrete ideas of what they really want.”
- **Even searchers need structure to orient themselves.**  
As **Jakob Nielsen** says in this [article on search](#), “users who get to a page through search still need structure to understand the nature of the page relative to the rest of the site. They also need navigation to move around the site in the neighborhood of the page they found by searching.”
- **And the list goes on...**  
**Raluca Budi** has written [an excellent summary of 5 reasons](#) why websites cannot rely only on search.

**Users also need to be able to “see” the scope and shape of our site**, and this is most easily done by seeing the grouping of content on our site – the headings and subheadings. We all do this now when we visit websites, but we've learned that some sites are clearly organized and some are, shall we say, **somewhat less clear**.

---

**Next:** [Chapter 1 - key points](#)

## Chapter 1 - key points

Empirical design (testing our site as we go) puts us at less risk than “genius” design (where we only discover the truth once the site is live).

If our users can't find the content they want, it doesn't matter how good the content is.

To make things easy to find on our site, search is not enough. We need a clear site tree – a “skeleton” of headings and subheadings - to help our users navigate.

---

**Next:** [Chapter 2 - Tree testing - a quick intro](#)



## 2 - Tree testing - a quick intro

*"Tree testing is like analytics for a site you haven't built yet." - Sam Ng*

This chapter is a short overview of tree testing – what it is, why and when to use it, and what the basic steps are.

By the end of this chapter, we should have a general idea of how tree testing works, and whether it might be a good addition to our own UX toolkit.

---

### What is tree testing?

About trees, tasks, paper vs. online, and synonyms

### Why run a tree test?

5 reasons to try this method on our next IA project

### When should we do a tree test?

The earlier, the better - ideally before we even have a site design

### How long will it take?

Simple test = 1 week. Full-fat test = 2-3 weeks.

### Which types of sites are most suitable?

Sites with traditional headings and subheadings, where the nav reflects the tree.

### What are the basic steps?

Planning, prepping the tree(s), writing tasks, recruiting, running, and analyzing the results.

### Key points

---

**Next:** [Chapter 3 - IA in the design process](#)

## What is tree testing?

- What is a “tree”?
- What is a task?
- Paper vs. online
- Video introductions to tree testing
- Other names for tree testing

In a nutshell, **tree testing is a simplified way to measure how easy it is to find items in a site structure.**

We run a tree test to see if our users can find what they’re looking for in a “tree” of headings and subheadings.

Tree testing only uses the bare text “skeleton” of our site, without any page layout, visual design, or even content. By design, it only looks at **two elements of our information architecture:**

- **Organisation** – how we divide our content into headings, subheadings, and so on
- **Labeling** – how we name those headings so they are (hopefully) clear and distinguishable

If our users consistently find the correct answer without too much time or effort, then our site structure is effective. If they can’t, tree testing helps us find out which parts of the tree are not working as well as they should.

For more on how tree testing fits into the bigger field of Information Architecture (IA) and how it compares to other IA methods, see [Chapter 3 - IA in the design process](#).

### What is a “tree”?

The “tree” in “tree testing” means the **hierarchical structure of our website** or information space, represented as a multi-level list of text headings:

/			
8	BikeSite		
9		For commuters	
0			Bike routes
1			Skills training
2			Cycling and the law
3			Tips for commuting
4		For families	
5			Places to cycle
6			Learning to ride
7			Safety tips
8		For roadies	
9			etc.
0		For mountain bikers	
1			etc.
2		For cycle tourers	
3		News & events	
4			

Typically this corresponds to the **global navigation of our website** (for example, the top-level tabs and their respective sections and subsections).

We may be testing an **existing tree** (e.g. the structure of our existing site) or trying out a **new tree** (revised, or completely rethought) to see how well it works.

## What is a task?

When users visit our site, they're almost always looking for something specific – a phone number, the features of a product, and so on.

In a tree test, we simulate this by giving them **tasks** – scenarios where they are asked to look for specific items in our site structure. These are typically the **most common and/or critical things our site visitors are seeking**.

Here's an example of a task for a banking site:

*"How would you arrange to get notified when your account balance is low?"*

## Paper vs. online

Tree testing was originally done using **index cards**, where each card showed a set of topics or subtopics. Given a written task...

**Your 9-year-old son asks  
for a new belt with a  
cowboy buckle**

...participants navigated down through the cards until they found what they were looking for:



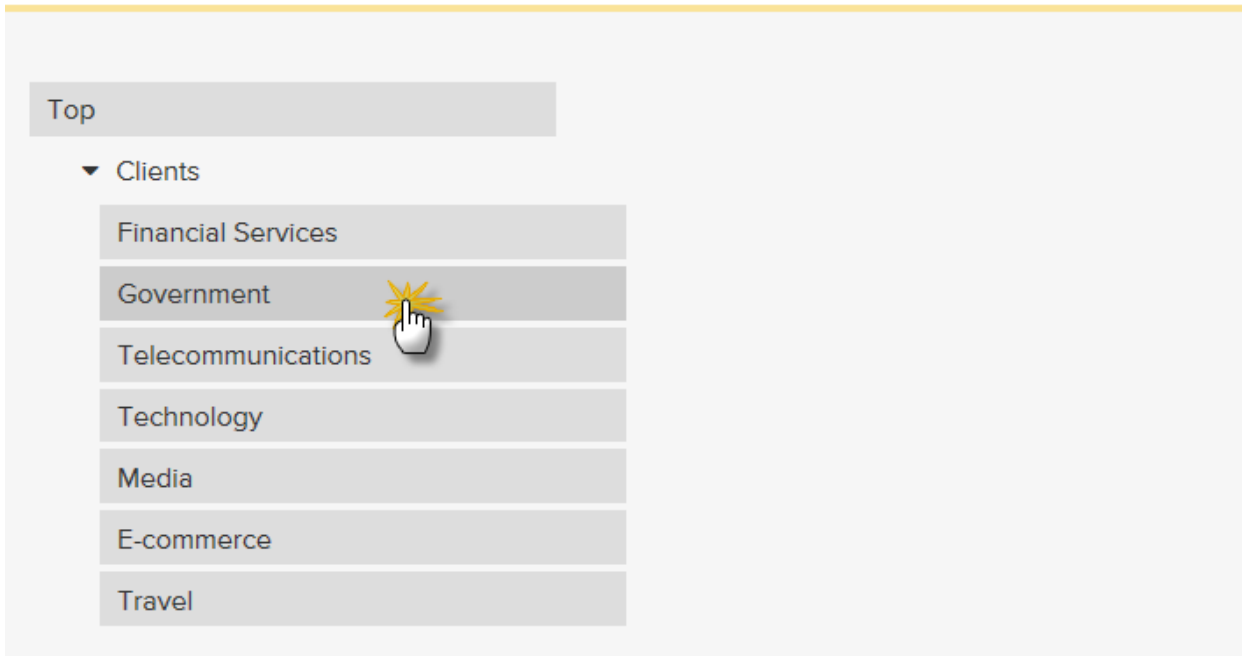
The moderator recorded the steps in a spreadsheet, then looked for patterns across participants.

Most tree testing is now done using [online web tools](#). The basics are the same, but **online studies are usually unmoderated** – that is, the participant can do the test any time, at their own computer, without a moderator present.

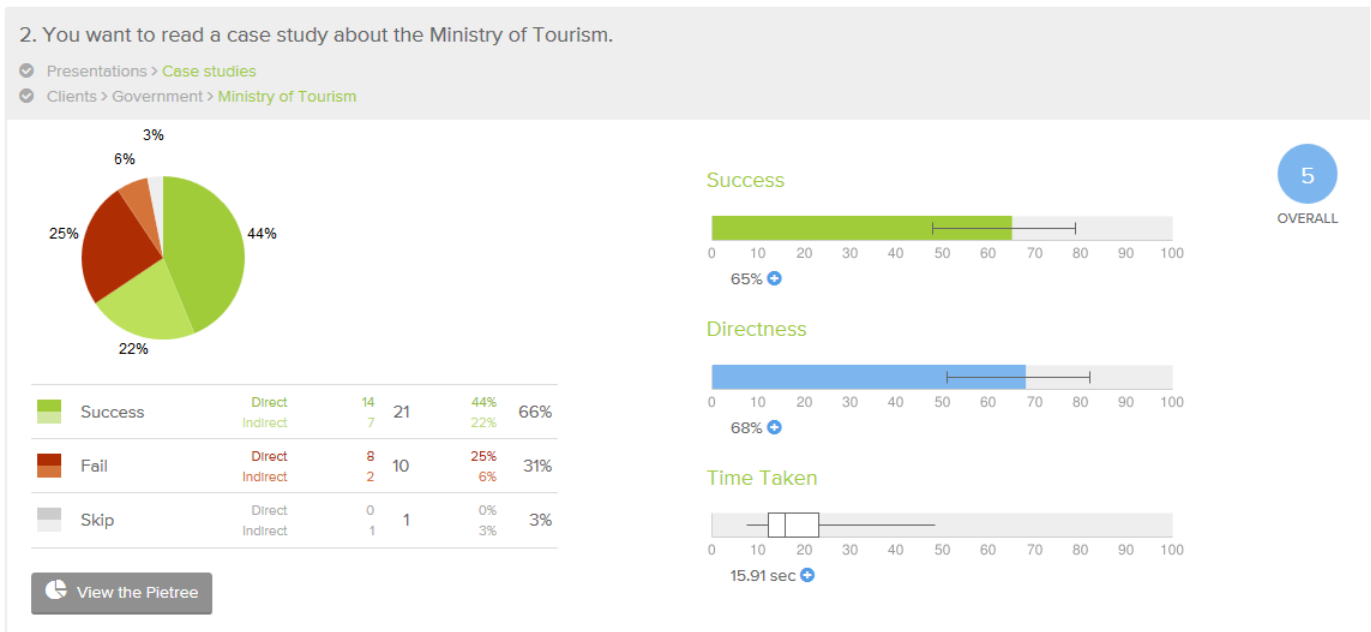
### Task 3 of 6

[Skip this task](#)

You want to read a case study about the Ministry of Tourism.



The other big difference with online tree testing is that the **analysis is much easier**, because the tool itself compiles the results and offers several ways to analyze and visualize them:



### Video introductions to tree testing

To understand the fundamentals of tree testing better, check out these short introductory videos from [tree-test tool vendors](#):

Treejack by Optimal Workshop	Tree testing with UserZoom

## Other names for tree testing

As tree testing has emerged as a separate technique in the IA toolkit, people have given it a number of different names. To avoid confusion, here is a list of terms that all refer to the method most popularly known as "tree testing":

- **Card-based classification evaluation**  
This was the original term used by Donna Spencer, who first documented the paper method in this [2003 Boxes & Arrows article](#).
- **Reverse (or inverse) card sort**  
This term links the method to card sorting (a complementary tool), but misleadingly implies that it involves sorting cards.
- **Task-based category testing**  
This term is more accurate, but still a bit of a mouthful. 😞
- **Information-architecture testing**  
This term is too broad; IA also includes navigation and search, which tree testing does not cover.
- **Taxonomy testing**  
In the web world, taxonomies usually mean site trees, so this term is accurate, though harder to spell than "tree test". 😊

---

Next: [Why run a tree test?](#)

## Why run a tree test?

---

We already have card sorting, usability testing, and a variety of other UX methods. What does tree testing offer that these other methods don't?

- **It's quick.**  
We can set up, run, and analyze a tree test in a week or less.
- **We can test *before* we have a site or even a prototype.**  
All we need is a tree of topics, some representative tasks, and some willing participants.
- **It focuses on site structures only.**  
By testing organization and labeling by themselves, tree testing provides a solid foundation that we can build on later, adding page design, visual design, and search.
- **It's better at testing structures than closed card sorting.**  
Closed card sorting was the traditional way to test site categories, but tree testing is more realistic and deals with all levels of the tree.
- **We get an objective evaluation of our headings and terms.**  
Qualitative feedback from users and opinions from stakeholders are good, but checking that against quantitative results from our target audience is much, much better.

For more on how tree testing fits into UX design (and how it compares to other UX methods), see [Chapter 3 - IA in the design process](#).

---

Next: [When should we do a tree test?](#)

## When should we do a tree test?

---

Tree testing is designed to give us **rough answers early** in the design process:

- **Initial research**

Along with the other up-front research we would normally do (surveys, interviews, card sorting, and so on), it's useful to do a **baseline tree test of the existing site**. This shows us problem areas to investigate, and provides a benchmark score that we can compare our new site structure to.

- **Early design phase**

It's common to start drafting our site structure while doing the first conceptual round of interaction design. As soon as we have one or two tree ideas roughed out, we can run tree tests to see how well they work (compared to each other and the existing structure), and what needs to change.

- **NOT late in the project**

Once we have a working prototype or a real site constructed, **it's better to do traditional usability testing** because it combines everything that affects our IA - not just organisation and labelling, but also navigation, page design, visual design, and content.

---

Next: [How long will it take?](#)

## How long will it take?

---

If we have a simple tree and a willing audience of participants, we can get a simple tree test up and running in a day or two, and have results to act on a day or two after that. **Call it a week.**

If we're testing several trees (which we recommend when designing a new site or redesigning an existing one) and collaborating with a project team, that week will likely grow to 2 or 3 as we brainstorm ideas, clean them up, test them, and decide how to act on the results.

Note that **this does not mean it takes 3 weeks of full-time effort** to run a set of tree tests. Some of that time will likely be spent waiting for stakeholders to review our trees and tasks, or waiting for participants to do their thing and generate results. In the meantime, we can be working on other aspects of our design.

For a sample timeline, see [When will we test?](#) in Chapter 4.

---

Next: [Which types of sites are most suitable?](#)

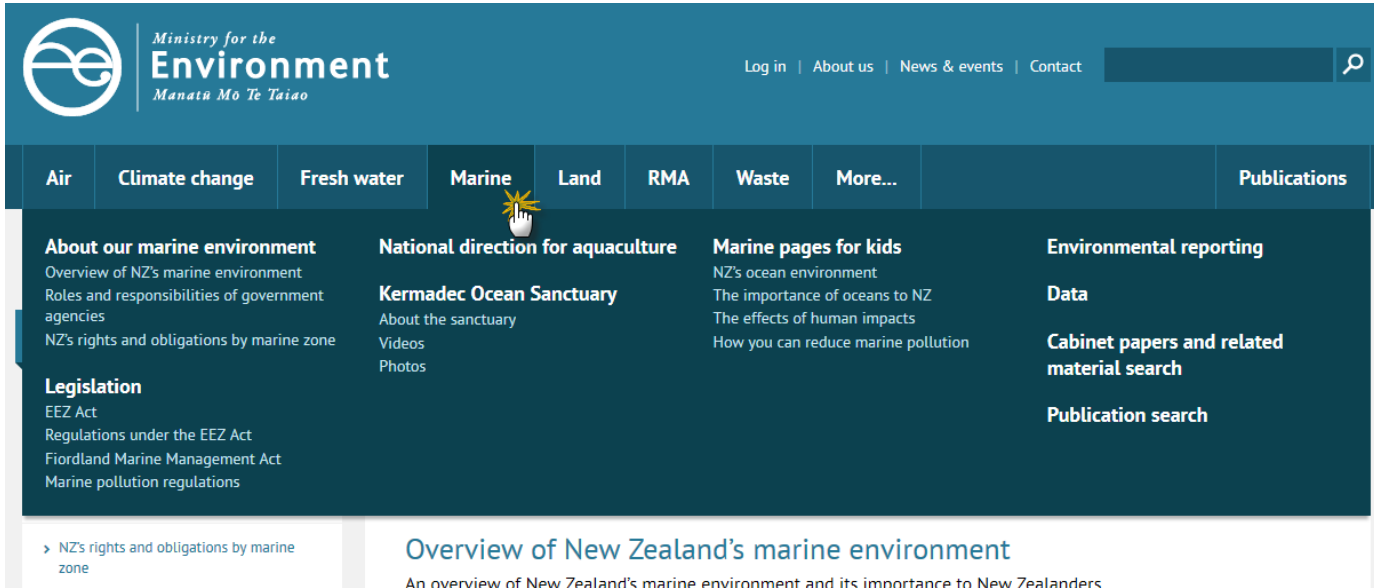


## Which types of sites are most suitable?

Tree testing is likely to suit if:

- We can outline our site as a tree (a hierarchical site map), with headings and subheadings and so on down.
- Our global navigation follows this site tree.

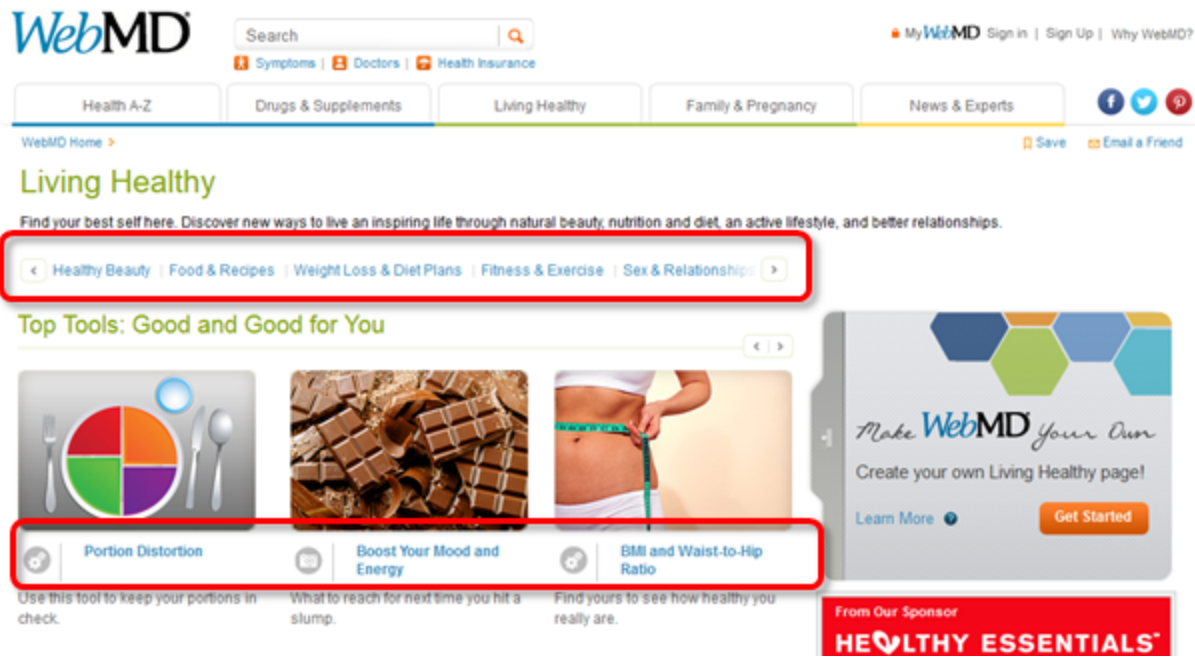
**Most content-centric sites are like this.** For example, here's a government site showing a main row of tabs and a menu of subheadings for one of them:



The screenshot shows the Ministry for the Environment website. The header includes the logo, the text "Ministry for the Environment" and "Manatū Mo Te Taiao", and navigation links for "Log in", "About us", "News & events", and "Contact". A search icon is also present. Below the header is a main navigation menu with tabs for "Air", "Climate change", "Fresh water", "Marine", "Land", "RMA", "Waste", "More...", and "Publications". The "Marine" tab is selected, and a sub-menu is displayed below it. The sub-menu includes sections for "About our marine environment", "Legislation", "National direction for aquaculture", "Kermadec Ocean Sanctuary", "Marine pages for kids", and "Environmental reporting". The "About our marine environment" section includes links for "Overview of NZ's marine environment", "Roles and responsibilities of government agencies", and "NZ's rights and obligations by marine zone". The "Legislation" section includes links for "EEZ Act", "Regulations under the EEZ Act", "Fiordland Marine Management Act", and "Marine pollution regulations". The "National direction for aquaculture" section includes links for "About the sanctuary", "Videos", and "Photos". The "Marine pages for kids" section includes links for "NZ's ocean environment", "The importance of oceans to NZ", "The effects of human impacts", and "How you can reduce marine pollution". The "Environmental reporting" section includes links for "Data", "Cabinet papers and related material search", and "Publication search". Below the sub-menu, there is a link for "NZ's rights and obligations by marine zone" and a section titled "Overview of New Zealand's marine environment" with a sub-heading "An overview of New Zealand's marine environment and its importance to New Zealanders".

However, some sites show the user several ways to navigate on any given page, so that it's hard to make out which is the "main" top-down structure.

For example, here's the [WebMD](#) site, showing the *Living Healthy* section. The topics offered here are circled in red:



However, the *Living Healthy* mega-menu shows a different set of topics:



It would be hard to run a realistic tree test on a site like this. Usability testing of the actual site (or a high-fidelity prototype) would yield more reliable results.

Next: What are the basic steps?

## What are the basic steps?

---

Tree testing is very much like usability testing – we ask participants to try doing typical tasks, except that we present them with a simple text tree rather than a real website.

To run a tree test, we typically do the following:

### 1. Plan the test

Before we do anything else, we need to answer a few basic questions such as “What are we trying to find out?”, “Which tree ideas should we test?”, “Who will we ask to participate?”, and so on.

For more on this, see [Chapter 4 - Planning a tree test](#).

### 2. Prepare the tree(s)

Whether we're testing an existing structure (as a baseline) or a new one (as described in [Chapter 5 - Creating trees](#)), we'll need to decide if we're testing the whole tree or not, which headings to include/exclude, how to spot missing content, and so on.

For more on this, see [6 - Preparing a tree for testing](#)

### 3. Write tasks for users to do

We typically pick the most common and critical activities for our various user groups, while making sure we cover the parts of the tree are most doubtful. And we need to be careful about task wording to avoid confusing users or giving away the answer.

For more on this, see [Chapter 7 - Writing tasks](#).

### 4. Set up the test using our chosen tool

Online tree-testing tools offer several features beyond the bare test itself, including survey questions, integration with online user panels, and so on.

For more on this, see [Chapter 8 - Setting up a test](#).

### 5. Recruit participants

We need to decide how many participants we need from each user group, how to invite them (e.g. web ads, email blasts, etc.), whether to offer incentives, and so on.

For more on this, see [Chapter 9 - Recruiting participants](#).

### 6. Pilot and run the test

We preview the test to shake out the bugs, then launch it and keep an eye on its progress, looking for response rates, initial scores, and drop-out rates.

For more on this, see [Chapter 10 - Piloting the test](#) and [Chapter 11 - Running the test](#).

### 7. Analyze the results

The online tools handle most of the analytical gruntwork, but we still need to know what to look for – success rates, backtracking, slow response times, and patterns across tasks. And we need to be able to communicate our results to our project team and management.

For more on this, see [Chapter 12 - Analyzing results](#) and [Chapter 13 - Communicating results](#).

### 8. Revise our tree and retest

Our analysis will have suggested several parts of the tree that need fixing. But after we make those changes, we should retest to make sure we got things right.

For more on this, see [Chapter 14 - Revising and retesting](#).

---

Next: [Chapter 2 - key points](#)

## Chapter 2 - key points

Tree testing is a **simplified way to measure how easy it is to find items** in a site structure.

In a tree test, we ask participants to **look for specific items** in a “tree” of headings and subheadings.

Tree testing involves two fundamental elements of IA – **organization** and **labeling**.

Tree testing is **quick, focused, better than closed card sorting** at testing structures, and **can be run before we have a website**.

Tree testing is **best done in the early stages of design** – ideally as part of the initial research phase.

It takes **1-2 weeks** to run a tree test, depending on the complexity of our site and design process.

Tree testing suits websites that have a **fundamentally “tree-like” structure**, and where the **main navigation reflects this tree**.

---

**Next:** [Chapter 3 - IA in the design process](#)

## 3 - IA in the design process

*"Even the coolest feature or the most compelling content is useless if people can't find it." - Jakob Nielsen*

When we're designing a website, nothing happens in a vacuum. Every facet of design – research, IA, interaction design, visual design, content strategy - tends to affect everything else.

So it's important to know where **Information Architecture** (in general) and **tree testing** (in particular) fit into the design process.

---

### How does IA fit into design?

It's not just for breakfast any more

### How does tree testing fit into design?

Do it early in the design process - later is too late

### The research phase

Contextual inquiry, open card sorting, and baseline tree testing

### The design phase: creating new trees

Using research to create new trees

### The design phase: going wide

Testing several alternative trees against each other

---

### The design phase: going deep

Iterating until we get the tree right

### Putting it all together

Diagrams of the full-fat and hurry-up approaches

### Comparing to other IA methods

Tree testing vs. closed card sorting, usability testing, and analytics

### Key points

**Next:** [Chapter 4 - Planning a tree test](#)

## How does IA fit into design?

---

**Information Architecture** (IA) concerns itself with the very broad (overall site structure and navigation) and the very detailed (labeling and content).

Early in the design process, information architects look at the scope and state of their content, and ask questions such as:

- How much of this content are we keeping/deleting/adding?
- Which content is up to date? Which needs reworking for our target audiences?
- How will we **organize** the content? What do our users mentally group this material?
- Can we standardize **terms**, and if so, which terms should we use?

Later in the design process, information architects shift to thinking about how users will find what they're looking for, including basic elements such as:

- Global **navigation** (global and contextual)
- **Search** (content and metadata)

Once a prototype or working site is up and running, information architects want to see how all of these factors combine with interaction design, visual design, and actual content to get users to the answers they want. Typically they find out by using traditional UX evaluation methods such as:

- Usability testing (in-person or remote)
- Analytics (page visits, abandons, etc.)

If done properly, **IA doesn't just happen at the beginning of the project**; it progresses from beginning to middle to end.

---

Next: [How does tree testing fit into design?](#)

## How does tree testing fit into design?

---

We mentioned four basic IA elements above – **organization, labeling, navigation, and search**. These are taken from the classic "polar-bear book" - *Information Architecture For the Web and Beyond (4th edition)* - by Rosenfeld, Morville, and Arango.

In general, we tend to deal with the first two (**organization and labeling**) as foundations before addressing the second two (**navigation and search**) which segue into UI design.

Because tree testing is only concerned with the first two elements (organization and labeling), it makes sense that **tree testing should be done early in the design process**, not so much later.

To get more specific, tree testing typically happens at two points early in the design of a website:

1. **During the research phase**, before new site structures are even conceived.
2. **Early in the design phase**, when our research has suggested ways to structure the site, and we have started creating site trees to exercise those ideas.

Let's take a closer look at each of these phases and see exactly where tree testing fits in...

---

Next: [The research phase](#)



## The research phase

- Using contextual inquiry
- Using card sorting to generate ideas
- Baseline the existing tree

Of the three phases of user experience (research, design, and testing), research is the one that tends to get abbreviated (or skipped entirely) when budgets and proposals are drawn up. Experienced designers know that this introduces big risks to the success of the new site, because it's harder to design something good when we don't really know who the users are, what they know, what they need, and what causes them the most pain.

While the first method discussed below (contextual inquiry) does take an investment of time and money, the other two methods (card sorting and baseline tree testing) can be done quickly and cheaply, so even with a shoestring research budget we should be able to get some useful insights to help us design a better site structure.

### Using contextual inquiry

When we don't know enough about our users, one of the best methods of finding out more is **contextual inquiry** – a fancy term for “watch, listen, then ask”. [Wikipedia](#) describes it as:

*...an approximately two-hour, one-on-one interaction in which the researcher watches the user do their normal activities and discusses what they see with the user*

Ideally, we visit the user in their natural habitat (home, office, commute, whatever makes sense for our purposes) and observe them doing tasks related to our website. During or after the observation, we ask them questions about what they did and why, so we can get a clear idea of what they knew, how they behaved, what they wanted, which issues they encountered, and so on.

As information architects, we pay particular attention to the respective elements of IA:

- which **content** they were after, and why
- how they mentally **grouped** different kinds of content
- which **terms** they used (and which terms they knew of)
- how they **browsed** or **searched** for items

For example, if we're researching how cyclists buy bike gear online, we might observe 20 people and discover the following:

<b>Content</b>	Most are looking for parts and accessories, not bikes (perhaps because they prefer examining and trying bikes in person at a store).
<b>Grouping</b>	Non-experts preferred items to be grouped by topic (e.g. parts, clothing, etc.), not by brand.
<b>Terms</b>	Most understood <i>Parts</i> vs. <i>Accessories</i> , but did not know the difference between <i>one-speed</i> and <i>fixed-gear</i> bikes.
<b>Browsing/searching</b>	They all started by browsing the menus unless they already knew the exact model name/number to search for.

Contextual inquiry takes time and some practice, but it's great for showing what users really care about, how they behave, and why. **We consider it the most fundamental user-research method in our toolbox.**

For more on contextual inquiry, see:

- [Usability Body of Knowledge: Contextual Inquiry](#)
- [User and Task Analysis for Interface Design](#), by JoAnn Hackos and Ginny Redish
- [Contextual Design: Defining Customer-Centered Systems](#), by Hugh Beyer and Karen Holtzblatt

### Using card sorting to generate ideas

**Card sorting** is probably the best known (and most used) method in information architecture, and with good reason – it’s a great way to find out how users think about content.

Early in a project, when we’re looking for ideas for structuring our site, we can run an **open card sort** to see how our users mentally organise topics.

Whether it’s done in person using index cards, or online using web apps, the fundamentals are the same:

- We create about 40 cards that represent a range of topics on our site.
- We ask participants to sort these cards into groups.
- We ask participants to name the groups they’ve created.

Here’s an example from the New Zealand Film Commission, where we asked filmmakers to sort typical content from the Film Commission’s website:

The screenshot shows a digital card sorting interface. On the left is a vertical list of 6 unsorted cards. On the right, five groups of sorted cards are displayed in grey panels with expandable headers:

- Funding** (8 cards):
  - definition of a low-budget film
  - what NZFC can do for aspiring film-makers
  - about professional development awards
  - financial help to develop my feature film
  - what funding is available for shorts
  - types of films that the NZFC supports with money
  - find out if my film has significant NZ content
  - tax/financial incentives available in NZ
- Learning** (7 cards):
  - I have written a screenplay, what happens next
  - film-making workshops for low-budget features
  - what training/support is available for me
  - how other film-makers got their films made
  - I've made several films, what can I do to upskill
  - practical guides on film-making
- Selling & distributing** (6 cards):
  - find local distributors in NZ and Australia
  - a person to contact about sales and marketing
  - who I can speak to about selling my film internationally
  - submission dates for international film festivals
  - how do I sell my film
  - best way to sell my documentary
  - countries that NZ has co-production treaties with
- Find films** (5 cards):
  - list of top 10 NZ box-office films
  - find short films with Maori content
  - get permission to screen "Goodbye Pork Pie" for my fundraiser
  - which film festivals "Whale Rider" was screened at
  - what films have recently been made
  - short films that Grant Lahood made
- About us** (4 cards):
  - location of the NZFC offices
  - purpose of the NZ Film Commission
  - where else to find info on NZ films
  - a contact at NZFC for making documentaries

Below the 'Funding' group, there is a separate 'News & events' group with 2 cards:

- NZ films that are shooting now
- what film-related events are currently on and where

We then analyze the data to see if there are patterns in how the participants grouped the items, and what they called those groups.

62	what training/support is available for me
53	I've made several films, what can I do to upskill
44	film-making workshops for low-budget features
44	film-making help
36	film-making help
26	film-making help
10	film-making help
13	film-making help
11	film-making help
4	film-making help
1	film-making help
3	film-making help
3	film-making help
5	film-making help
11	film-making help
40	film-making help
35	film-making help
4	film-making help
1	film-making help
2	film-making help
2	film-making help
10	film-making help
21	film-making help
17	film-making help
8	film-making help
4	film-making help
11	film-making help
1	film-making help
1	film-making help
2	film-making help
4	film-making help
7	film-making help
6	film-making help
22	film-making help
13	film-making help
20	film-making help
6	film-making help
22	film-making help
18	film-making help
28	film-making help
24	film-making help
21	film-making help
19	film-making help
22	film-making help
18	film-making help
20	film-making help
12	film-making help
10	film-making help
9	film-making help
11	film-making help
5	film-making help
8	film-making help
13	film-making help
10	film-making help
11	film-making help
9	film-making help
3	film-making help
3	film-making help
4	film-making help
5	film-making help
8	film-making help
7	film-making help
14	film-making help
11	film-making help
19	film-making help
20	film-making help
19	film-making help
29	film-making help
20	film-making help
18	film-making help
9	film-making help
8	film-making help
4	film-making help
6	film-making help
4	film-making help
7	film-making help
3	film-making help
5	film-making help
7	film-making help
6	film-making help
3	film-making help
2	film-making help
3	film-making help
9	film-making help
6	film-making help
20	film-making help
19	film-making help
29	film-making help
2	film-making help
3	film-making help
1	film-making help
6	film-making help
2	film-making help
1	film-making help
6	film-making help
4	film-making help
5	film-making help
1	film-making help
1	film-making help
3	film-making help
1	film-making help
7	film-making help
7	film-making help
26	film-making help
42	film-making help
57	film-making help
59	film-making help
60	film-making help
61	film-making help
62	film-making help
63	film-making help
64	film-making help
65	film-making help
66	film-making help
67	film-making help
68	film-making help
69	film-making help
70	film-making help
71	film-making help
72	film-making help
73	film-making help
74	film-making help
75	film-making help
76	film-making help
77	film-making help
78	film-making help
79	film-making help
80	film-making help
81	film-making help
82	film-making help
83	film-making help
84	film-making help
85	film-making help
86	film-making help
87	film-making help
88	film-making help
89	film-making help
90	film-making help
91	film-making help
92	film-making help
93	film-making help
94	film-making help
95	film-making help
96	film-making help
97	film-making help
98	film-making help
99	film-making help
100	film-making help

The findings from a card sort can fundamentally change how we structure our site. For example, suppose we are designing a recipe website.

- We might have initially thought about grouping the recipes by time of day:

- Breakfast
  - Hot
  - Cold
- Lunch
- Dinner
- Snacks

- If the card sort revealed that most of our participants grouped the recipes by cuisine, we should really reconsider our main headings:

- French
- Italian
- German
- Japanese
  - Sushi
  - Teppanyaki
  - Yakitori
- Chinese

And card sorts are not restricted to top-level headings. **We can also run card sorts on subsets of our content**, to generate ideas for the next few levels down.

For any medium or large website, we recommend running open card sorts as part of the research that happens before we jump into design.

The definitive book on card sorting is Donna Spencer's [Card Sorting: Defining Usable Categories](#). For a quick primer, see her [2004 Boxes & Arrows article](#).

## Baselining the existing tree

During the research phase, we also recommend running a tree test, even before we've created draft site structures.

How can we test a tree if we don't have a tree yet?

If we're redesigning an existing site, we **do** have a tree (our existing site structure), and we should definitely run a tree test on it, even if we know the existing tree isn't very effective.

Why? Because **it gives us something to measure our new ideas against**. Remember, we don't just want to create a new site structure; we want to create a **better** one. And the way we ensure that is by baselining the old tree and later measuring it against our new ideas.

For example, one of the first organizations to use tree testing was ACC, a public-health service in New Zealand. When they redesigned their website a few years ago, they did a baseline test *before* they revised their site tree. Here's what they found:

Tree	Overall score
Existing site	30%
Revised site	67%

And the scores are only part of the story. Baselining the old tree also helps us find out:

- **Which parts of the old tree are problems** that we can fix or rethink
- **Which parts of the old tree are actually working well**, so we can reuse those in the new site. We don't want to throw the baby out with the bathwater.

Over and over, we see redesign projects head off in the wrong direction because the stakeholders *think* they know what to change. Doing proper research (such as contextual inquiry, card sorts, and baseline tree testing) helps us find the right direction before we start designing in earnest.

---

Next: [The design phase: creating new trees](#)

## The design phase: creating new trees

- [Creating new trees](#)
  - [Going wide and going deep](#)
- 

**Primetime for tree testing is early in the design phase**, once we've done enough research to feel we have a good handle on our audiences, their background, and their needs.

We start creating drafts of new site structures **immediately after we finish a content audit** – that is, after we decide which content we will be adding, updating, or deleting. While content is always a moving target, it really helps to have most of it identified before trying to design a structure for it.

This work on content and structure can be done in parallel with conceptual design, but usually comes before more detailed work such as page layouts, fine-grained interactions, and visual design.

### Creating new trees

**From our research, we should have several ideas** about what to change (and what not to) in a new site tree – not just grouping, but labeling too.

**We can then start sketching out new trees by looking at these ideas and our list of planned content.** It's typical to rough out 2-5 different trees at this stage, down to level 2 or level 3, just to explore how they might work. We might do this ourselves, or (even better) we might involve the whole team to get a wider variety of informed ideas.

For more on creating trees, see [Chapter 5 - Creating trees](#).

### Going wide and going deep

In the design phase, we can increase the quality of our site tree by doing two critical things:

- **Going wide** (testing several alternative trees at the start)
- **Going deep** (testing and revising down to a single tree that performs well)

The next two sections describe each of these approaches in more detail.

---

**Next:** [The design phase: going wide](#)

## The design phase: going wide

- Testing several alternatives at once
- Data for the CEO
- Guarding against genius design
- Learning from other industries
- Going wide – an example

In the design phase of a project, **it's a mistake to create just a single new site tree.**

### Testing several alternatives at once

It's too early to finalize our thinking; we probably haven't done enough research and we haven't actually tested our idea with users. If it doesn't work out, what do we do – start over? **If we only create (and test) a single site tree, we're taking a big (and unnecessary) risk.**

The smart thing to do here is “**go wide**” – that is, generate several different site trees to exercise our various ideas, then pick the 2 or 3 most promising trees to test against each other.

### Current site

- **Why join us?**
  - Residential
  - Business
  - Agribusiness
  - Residential pricing
  - Join
  - Offers
- **Save energy & money**
  - Residential
  - Business
  - etc.
- **My account**
  - My bill
  - etc.
- **About us**

### Idea 1

- **Residential**
  - Plans and pricing
  - Offers
  - etc.
- **Small business**
  - Plans and pricing
  - etc.
- **Corporate**
  - Products and pricing
  - etc.
- **Farm**
  - Plans and pricing
  - etc.
- **My Account**
- **About us**

### Idea 2

- **Join**
  - Join
  - Moving
- **Plans and rates**
  - Residential plans
  - Business plans
  - etc.
- **Better off with us**
  - Great online tools
  - etc.
- **Your account**
  - My bill
  - How to pay
  - etc.
- **About us**

This may seem like a lot of extra work, but it's not. **It turns out that trying to settle on a single tree at this stage is usually very difficult.** There are different groupings to try, different terms to try, and (if we're working with a team) diverging ideas from other members. We've found it easier to generate several trees that incorporate these inputs than it is to try hacking and slashing them all in a single structure.

**By “going wide”, we raise our chances of hitting on the best design.** We start with several candidate trees, then use a reliable method (tree testing) to choose the best one.

For more on going wide in UX design, see Jared Spool's [short article on exploring multiple variations](#).

### Data for the CEO

Consider also that the “other members” that we get input from might include the CEO. And (trust us), if we need to shoot down their

out-of-left-field idea, it's much easier to do that with objective data from testing than just our personal opinions. 😊

## Guarding against genius design

Perhaps most importantly, testing lots of ideas early on avoids **the peril of genius design**. By that, we mean designers who believe they are talented, and tend to believe that all their ideas are good ones.

If we have a "genius" designer on the project, we must tread very carefully. Because when a designer falls in love with a single idea early on, it's really hard for them to get away from it.

## Learning from other industries

Fixing on a single idea too early is **not a new problem**. Other industries encountered this (and solved it) years ago.

Consider traditional graphic design and advertising. In both professions, the classic approach is to go wide in the early conceptual phase:

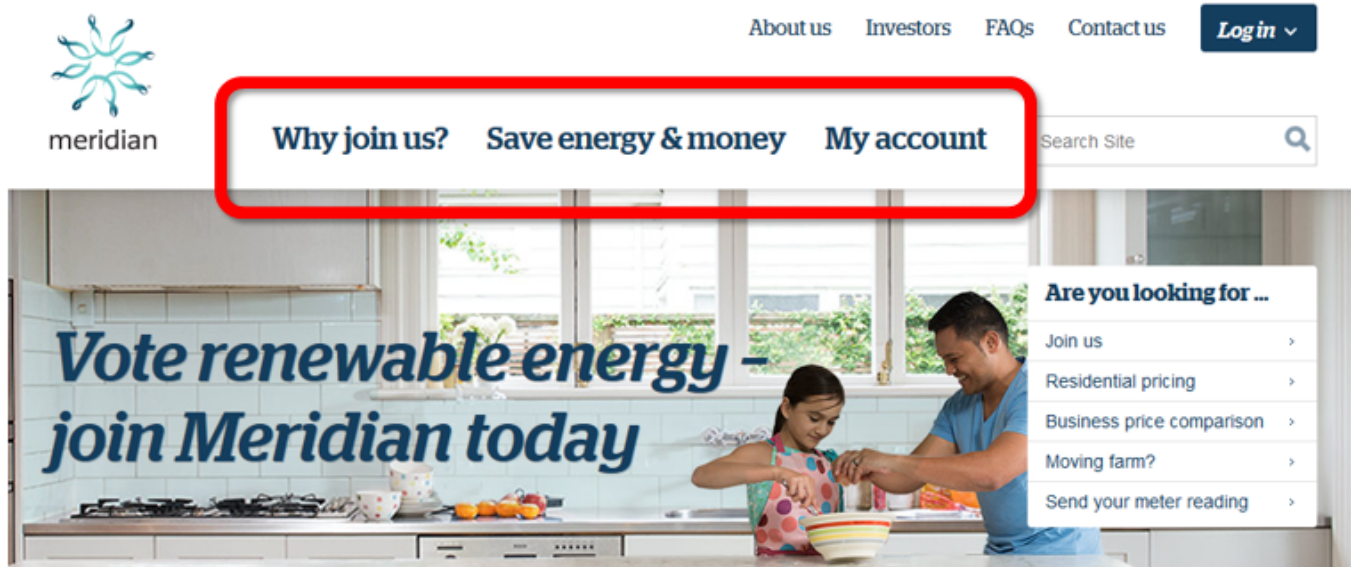
- Brainstorm dozens of ideas
- Explore 5-10 in more detail
- Pitch the top 3 to the client

It brings to mind the advertising person who said that if they went away and worked and then came back with a single idea to pitch, they would be fired on the spot.

## Going wide – an example

For an example of testing alternative ideas, let's look at Meridian Energy, a renewable-power company that needed to redesign its site tree.

When they ran an **open card sort** with their users, the results suggested that the current top-level headings didn't match their mental model.

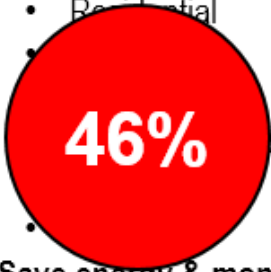


So they brainstormed a bunch of ideas, and decided that two of them were worth testing. They set up **three tree tests** - the two ideas plus the current site.

When the results came back a week later, the current tree perform poorly (as they expected). **But so did the two new ones:**

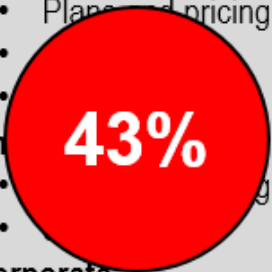
## Current site

- **Why join us?**
  - Residential
  - Pricing
- **Save energy & money**
  - Residential
  - Business
  - etc.
- **My account**
  - My bill
  - etc.
- **About us**



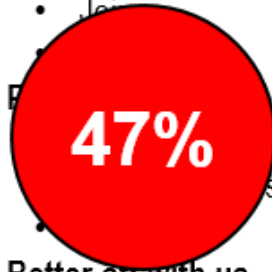
## Idea 1

- **Residential**
  - Plans and pricing
- **Small Business**
  - etc.
- **Corporate**
  - Products and pricing
  - etc.
- **Farm**
  - Plans and pricing
  - etc.
- **My Account**
- **About us**



## Idea 2

- **Join**
  - Join
- **Energy**
  - Plans and pricing
- **Better on with us**
  - Great online tools
  - etc.
- **Your account**
  - My bill
  - How to pay
  - etc.
- **About us**



Obviously, they were not happy about this. Time to start over, right?

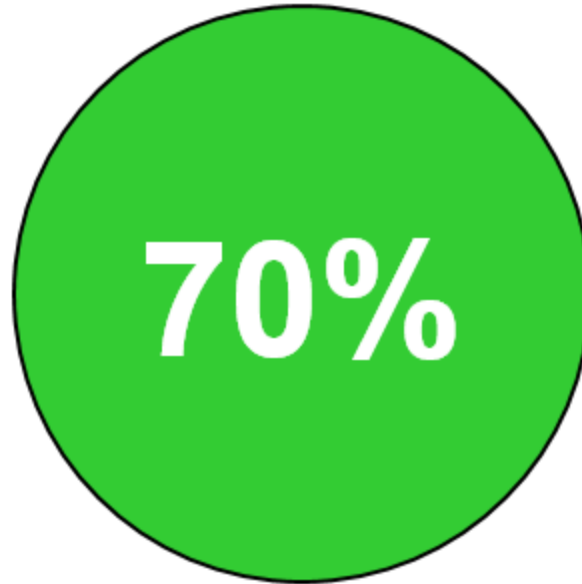
Well, not really. When they dug deeper into the results, they found patterns in both trees that performed very well. The problem was that there were other patterns and potholes that sabotaged the overall scores.

So, in round 2, they created a third tree that was a hybrid of the best of the earlier designs. And the third tree proved to be the charm:



## Tree 3

- **Your home**
  - New customers
  - Current customers
  - etc.
- **Agribusiness**
  - New customers
  - etc.
- **Small/med business**
  - New customers
  - etc.
- **Large business**
  - New customers
  - etc.
- **About us**



---

**Next:** The design phase: going deep

## The design phase: going deep

- [Iterating until we get it right](#)
  - [Keeping it cheap and fast](#)
- 

Making several site trees compete against each other is great, but at some point, we need to reduce them down to a single high-performing tree.

### Iterating until we get it right

Once we've run our first round of tree tests, on 2 or 3 of the most promising trees we thought up, we analyze the results. Typically we find:

- **One of the trees performed the best**, but it still has some problems to solve.
- **Some of the trees just didn't work** for our participants. We can safely discard these trees and move on.
- Some of the lower-performing trees may have elements (certain groupings or terms) that actually did perform well, so **it may make sense to incorporate these elements into our best tree**.
- (In rare cases, we may find that none of the initial trees perform well enough to continue with. We'll have to come up with some new ideas quickly (perhaps ideas that were discarded earlier) or go back and do more basic research with users.)

After the first round of tests, we either have:

- 2 trees that performed reasonably well, but should perform better with the revisions we have in mind, or
- 1 tree that clearly out-performed the others, and should improve further with our revisions.

We can now run a second round of tests to see if our revisions did indeed make things better:

- If we tested 2 trees, we can then determine which performed better and which we're more comfortable going with as our actual site tree.
- If we tested 1 tree, we can look for any remaining areas or terms that need tweaking.

In a perfect world, we would keep testing until we had a perfect tree, but there is never time or budget enough for that. We typically only do more than 2 rounds of testing if there are important parts of the tree that are still not performing well enough.

### Keeping it cheap and fast

If we were expecting tree testing to be a one-off trick – build a tree, test it, and we're done – it may be alarming that **we recommend several rounds of testing**, with several trees, winnowing and refining them until we get a single high-performing tree.

What makes this approach feasible is that **we now have mature testing tools** that are both:

- **Cheap** (affordable for any development team), and
- **Fast** (able to deliver results in a week or two)

The combination of **cheap and fast changes how we should approach design**. Instead of doing a single round of deluxe in-person testing (or worse, no testing at all), we can do several cheap online tests in less time and for less money.

There are cases where in-person testing is the way to go, particularly for complex interactions, or for when there are only a small number of participants available. But for most projects, **several rounds of lightweight tests are a better bang for the buck**.

And that means we can go wide at the start, and go deep through to the end.

---

Next: [Putting it all together](#)

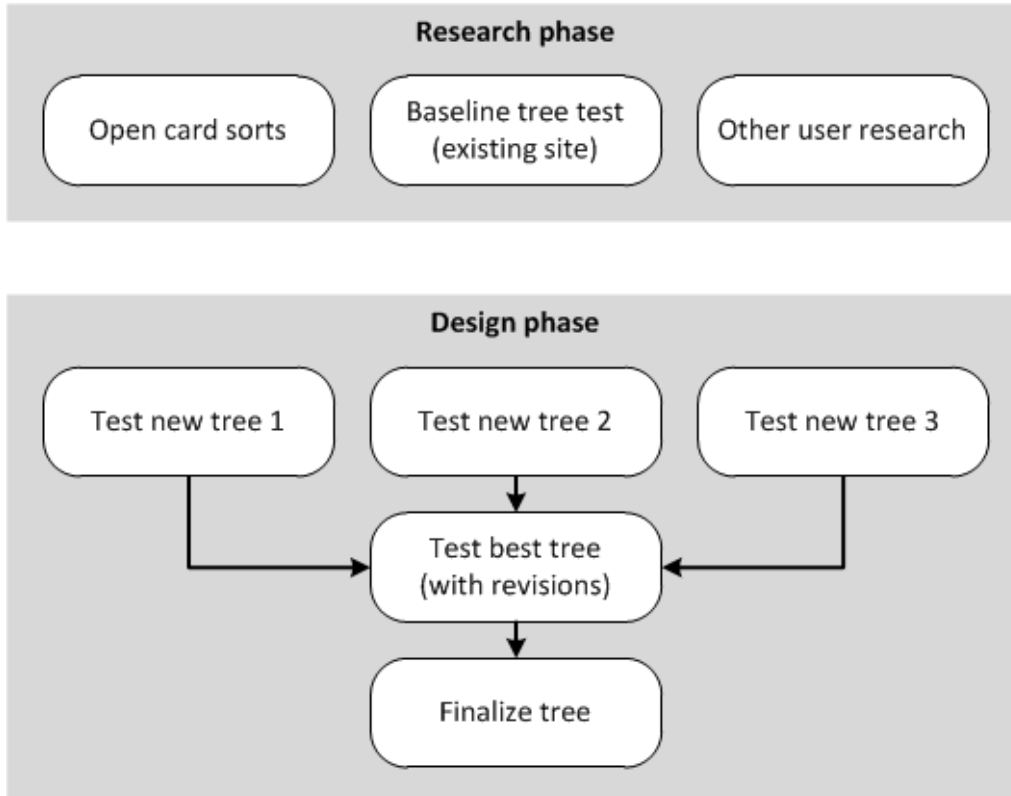
## Putting it all together

- The recommended approach
- The hurry-up approach

Let's take a step back now and look at how we can fit card sorting and tree testing into our overall design process.

### The recommended approach

If we have been given a reasonable amount of time to create a site tree, we do the following:



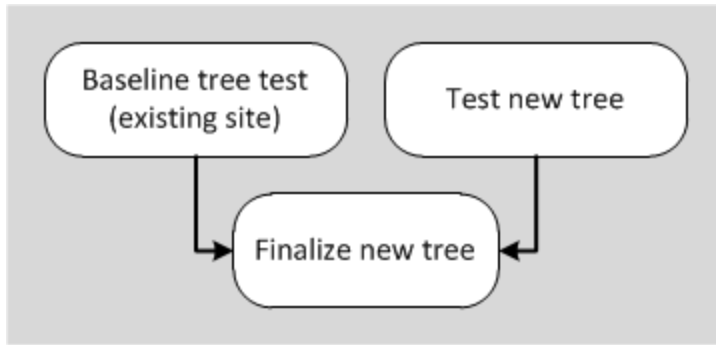
There are several advantages to this “full-fat” approach:

- **Card sorting generates ideas** for our new tree – ideas that come from actual users, not just the project team.
- **Baseline tree testing shows us what needs fixing** (and what doesn't), and gives us a score to measure against later.
- **Two rounds of testing for the new trees lets us experiment**, refine, and raise our chances of creating an optimal design.

### The hurry-up approach

**All of us have worked on a project where we were not given that “reasonable” amount of time to do what was needed.** Sometimes we have to cut corners.

Because we loath the idea of creating a site tree without some form of validation, we still run tree tests, but we abbreviate the process, using a single round that combines **before** and **after**:



---

Next: [Comparing to other IA methods](#)

## Comparing to other IA methods

- [Closed card sorting](#)
- [Usability testing](#)
- [Web analytics](#)

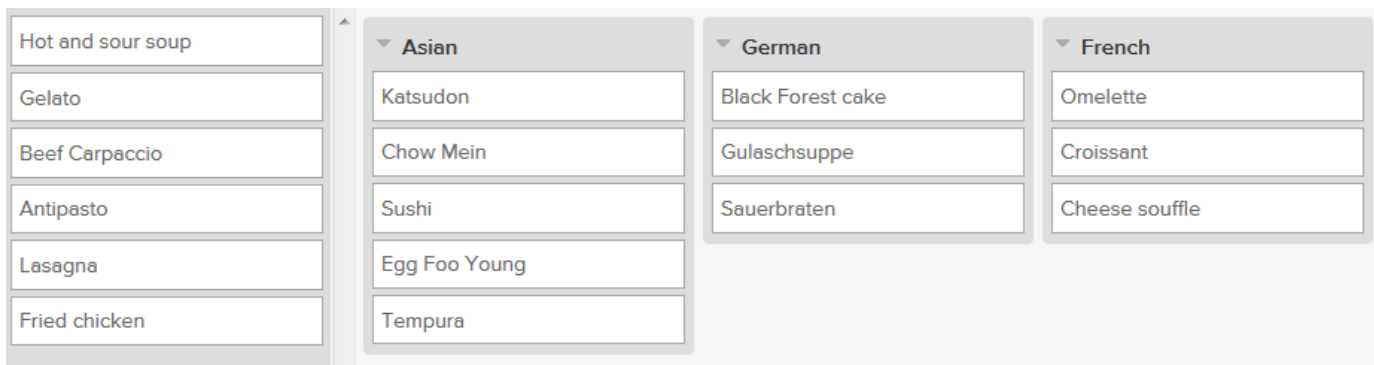
Tree testing, especially the online version of it, is a fairly recent technique, much newer than card sorting, for example. Before tree testing, designers used other methods to check the effectiveness of their site structures.

Here's our take on how does tree testing compares to a few of these methods.

### Closed card sorting

Before tree testing became established, **closed card sorts** were a popular way to evaluate a site structure.

In a closed sort, participants are given a "pile" of topic cards and asked to put them into pre-named groups. The group names are typically the top-level headings in the site tree:



Once enough participants have done the closed sort, we can inspect the results to see if they put the right cards into the right buckets and (if not) where they disagreed.

While this does help us judge the effectiveness of the top-level headings, **the results we get from a closed card sort are not as useful as a tree test**, because:

- Most closed sorts only test a single set of groups, which only represent a single level of the tree (usually the top level). Tree testing evaluates the full depth of the tree.
- Closed sorts are essentially a filing exercise ("Where would you put this topic?"). While this is similar to browsing a website, it doesn't mimic the act of browsing as well as tree testing does.
- Tree-test results are easier to analyze than closed-card-sort results (as we'll see in [Chapter 12 - Analyzing results](#)).

For more on comparing closed sorts to tree tests, see [David Juhlin's](#) well-considered article, [Is closed card sorting an outdated technique for IA?](#)

### Usability testing

Usability testing, whether in person or using remote tools, is a powerful technique, but **it differs in some major ways from tree testing**:

- Usability testing requires a website (or some kind of prototype – paper or clickable). Tree testing can be done earlier in the process because all it really requires is a text skeleton of a site tree.
- Tree testing isolates the organization of topics and their labeling. Usability testing, on the other hand, shows the effects of combining these factors with others such as navigation, search, actual content, visual design, and so on.

Early in the design process, it's good to simplify things by just working on a few factors at a time and getting them right. Tree testing lets us do that.

- Tree-testing tools largely automate the analysis of the results. Usability testing usually yields rich (but "analog") results that take more time and effort to interpret.

Tree testing certainly does not take the place of usability testing. **Think of tree testing rather as a complement to usability testing** - a early way of testing structural ideas before we even have a prototype.

It's also good to know that that **tree-testing results line up nicely with later usability testing**, as described in this [2014 academic paper](#).

## Web analytics

Analytics are great because they tell us what our total population of site visitors is actually doing on our site – where they click, where they don't, and so on.

The traditional weakness of analytics is that, while they can tell us (in detail) what actions our users are taking on the site, **they can't tell us why**. When we track visitors jumping from our home page to an intermediate landing page to a content page, we have no idea why they're going there; maybe they're looking for what we intended, maybe they're not. Maybe they leave because they found the answer they wanted; maybe they leave because they didn't. We have no context, so we don't know how well the site performed in that sense.

Like a usability test, tree testing gives us context by letting us set specific tasks for the user to do. When they go down a certain path in our tree, we know if that's a success or a failure.

Analytics are a great way to see where our users are going, and can reveal issues in our site structure, but on their own they're not enough.

---

Next: [Chapter 3 - key points](#)

## Chapter 3 - key points

We should **do tree testing early** in the design process.

In the initial research phase, **an open card sort and a baseline tree test** of the existing site will give us ideas for what needs changing in our site tree.

In the design phase, we should **"go wide"** by creating several alternative site trees incorporating the ideas from our research.

Online testing tools are cheap and fast, allowing us to **"go deep" (run several quick rounds of tests)**, winnowing and refining our ideas down to a single high-performing structure.

For evaluating site structures early in the design process, **tree testing is more effective than closed card sorts**.

---

Next: [Chapter 4 - Planning a tree test](#)

## 4 - Planning a tree test

*"Plans are nothing; planning is everything." - Dwight D. Eisenhower*

If we have a site structure to test and some tasks in mind, it's tempting just to dive right in – set up the test, email a bunch of users, and watch the results come in. Easy, right?

However, we'll get a lot more out of our testing if we take a step back and **ask ourselves some basic questions**, such as:

- Why am I running this test? What am I specifically trying to find out?
- What am I testing – the whole tree, or just the top levels? Or two completely different trees?
- Who should I test – existing customers, or prospective ones too?

How we answer these questions can change how we run our test and analyze our results.

---

### Why are we running this test?

Baselining, testing new/revised trees, comparing alternatives, etc.

### How many rounds of testing?

2 rounds is standard, but even 1 round will improve our site.

### Which trees will we test?

Existing tree vs. new tree(s)? The whole tree or just part of it?

### Who will we test?

User groups, recruiting, and incentives

### When will we test?

We can start as soon as we have a site tree roughed out

### Which tool will we use?

Treejack, UserZoom, paper, and other tools

### Where will we test?

Online using a browser vs. in person

### Who will do what?

Working with a team vs. going solo

### How will we handle problems?

Most problems can be prevented, but a safety net is always wise

### Documenting our plan

Some templates to keep us on track

### Key points

---

**Next:** [Chapter 5 - Creating trees](#)



## Why are we running this test?

- [Baselining an existing tree](#)
  - [Testing revised trees](#)
  - [Trying out new trees](#)
  - [Comparing alternatives](#)
  - [Testing groupings](#)
  - [Testing labeling](#)
  - [Sharing and documenting issues and goals](#)
- 

Why run a tree test? As we saw in Chapter 2, there are two common motivations:

- **To baseline an existing tree**, discovering where the problems are and establishing a base score.
- **To try out some new trees** that we've come up with, looking for problems and comparing to each other (and the baseline tree, if any).

### Baselining an existing tree

If we are testing an existing IA (e.g. the structure of a current website that we're about to revise), we're obviously interested in finding out which parts of the current structure work well and which don't.

Most of the time, **we will already have an idea of where some of the problems are**. It might be from other usability testing we've done, from web analytics, from user feedback, from our own gut feelings, or (most commonly) from some mixture of all of these.

When testing an existing IA, then, we're likely to be looking for:

- How well the **suspected** problem areas perform, and
- Which other (**unsuspected**) areas perform particularly well or poorly

### Testing revised trees

If we're revising a site structure, we will generally be looking for:

- How well the **revised** parts of the new structure perform, and
- Which other areas perform particularly well or poorly, especially **areas that may be indirectly affected** by the revisions we made.

### Trying out new trees

If we're creating structures for a new website, we may not have much existing research to inform our IA work. In this case, the main value of tree testing is being able to **evaluate one or more structures early in the design process**, before the website exists even in beta form.

### Comparing alternatives

Whether it's architecture or brand design or vacuum cleaners, the best designers agree on one thing – **generate lots of ideas early**, then cheerfully discard the ones that don't work out.

The same is true with site structures. **Early in the design phase, we should think up several different ways of structuring our site**. Yes, we will probably have a favorite, but our favorite may not be the best solution for our users. If we create some true alternatives and test them against each other, we're more likely to produce a better structure.

So, whether we're testing revised structures or new ones, a main goal of our testing should be to **compare multiple candidate structures and determine which performs best**.

In our experience, what often happens is that tree A performs best overall, but parts of tree B do better than their counterparts in A. The natural next step is to **create a hybrid** (tree C), which usually ends up testing better than either A or B. If we only created tree A, how would we ever get to C, the better structure?

### Testing groupings

For most designers, the main reason to run a tree test is **to determine if their main grouping scheme works well**.

For example, if we decide to go with a task-based scheme (e.g. installing a product, using it, getting support, uninstalling it, etc.), we want to know if our task-based headings help our users find the page they're looking for.

If we're testing several grouping schemes against each other (the recommended approach), **we want to find out if one scheme is clearly better than the others**. If several schemes work equally well, then we can choose based on other criteria (such as how much effort it will take to rework our content to fit a scheme).

We can also **flip our schemes and test which variation works better**. For example:

- We could create our first tree to use audiences as our top-level headings (e.g. teachers, students, parents, etc.) and tasks as our second level (choosing a school, enrolling, choosing courses, etc.).
- We could create a second tree that flips this scheme, so that our top level is tasks and our second level is audiences.

For more on grouping schemes, see [Chapter 5 - Creating trees](#).

## Testing labeling

Another big reason to run a tree test is to **test the terms we use**. Will our users understand what "contingency planning" means? Will they be able to distinguish between "products" and "solutions"?

Labeling is a critical element of information architecture, and it's often hard to get right. We must consider:

- **The terms we use internally (the organization's jargon) vs. the terms that users understand and prefer.**  
For example, we may say "Business Development", but our customers call it "Sales". Which term should we use on our website?
- **The conflicting terms that our various audiences use.**  
Doctors may be looking for "deep-vein thrombosis", but you and I would probably look for "blood clot".
- **The many synonyms we can choose from.**  
Languages are rich, and there are often several words that we could use. Which works best for our website visitors?

If we test alternative terms against each other, we'll usually find that one works substantially better than the others. That's a clear win.

However, we may also find that two terms work equally well, in which case we can decide based on other factors. For example, a consumer-review site in New Zealand considered renaming their "Electronics" section to "Technology", and this became the subject of prolonged internal arguments about whether users would understand the new term properly. When they ran tree tests, they made sure to include tasks that targeted these alternative terms in their trees. The result was a 51/49 split; both terms worked well, so they could use either depending on other factors.

## Sharing and documenting issues and goals

When we tree test, there are several problems we typically want to fix, but yours may not completely overlap with mine. To run a good study, we need to be clear about what we're trying to find out, which means discussing it and writing it down.

To do this, we typically run a short workshop (1 to 1.5 hours) to:

- make a list of the problems we're trying to solve, and rank them
- make a list of our specific goals for this study (some of which will spring directly from our issues list).

We record this list in a shared, public place (a project whiteboard, an online spreadsheet, etc.) so we can keep it handy when designing our tree tests.

---

Next: [How many rounds of testing?](#)

## How many rounds of testing?

---

The time, effort, money, and participants it will take to develop our site tree depends partly on how many rounds of testing we're intending to do. More rounds usually means a better result (as we would expect), but there are also diminishing returns to consider.

In [Putting it all together](#) in Chapter 3, we recommended a "full fat" process with 3 rounds of testing:

<b>Round 1</b>	Test the existing tree (baseline)
<b>Round 2</b>	Test 2-3 new tree candidates
<b>Round 3</b>	Revise/retest the best tree (often a hybrid)

Because of budget or time constraints, this is often cut down to **2 rounds**:

<b>Round 1</b>	Test the existing tree (baseline) and 2-3 new trees
<b>Round 2</b>	Revise/retest the best tree (often a hybrid)

The first round of testing shows us where our tree is doing well (yay!) and where it needs more work. So we make some thoughtful revisions. Careful, though, because even if the problems we found seem to have obvious solutions, we still need to **make sure our revisions actually work for users, and don't cause further problems**.

The good news is, **it's dead easy to run a second test**, because it's just a small revision of the first one. We already have the tasks and all the other bits worked out, so it's just a matter of making a copy of the test (in whatever tool we're using), pasting in our revised tree, and hooking up the correct answers. In an hour or two, we're ready to pilot it again (to err is human, remember) and then send it off to a fresh batch of participants.

There are two possible outcomes here:

- Our fixes are spot-on, the participants find the correct answers more frequently and easily, and our overall score climbs. We could have skipped this second test, but **confirming that our changes worked is both good practice and a good feeling**. It's also something concrete to show the boss.
- Some of our fixes didn't work, or (given the tangled nature of IA work) they worked for the problems you saw in round 1, but now they've caused more problems of their own. Bad news, for sure, but **better that we uncover them now in the design phase** (when it takes a few days to revise and retest) instead of further down the track when the IA has been signed off and changes become painful.

Note that Round 1 combines the "before" and "after" testing, because most of our clients have a good idea of where the weaknesses are in their existing tree. If we don't, the full 3-round approach described above is recommended; this can be combined with an open card sort to help generate ideas for the revised structure.

On some larger and more complex trees, additional revision rounds may be needed to confirm we have solved the major issues we uncover.

For planning, this means that we need to:

- add the desired # of rounds into our project schedule
- determine how we will get enough fresh participants for each round

---

Next: [Which trees will we test?](#)

## Which trees will we test?

- [How many trees?](#)
  - [Which part of the tree?](#)
- 

This usually comes down to 2 questions:

- How many trees are we testing at a time?
- Are we testing the whole tree, or just part of it?

### How many trees?

If we're testing an existing tree for problems, before starting our IA redesign, the answer here is simple – we're testing just the one tree.

If we're revising the IA for a site, and we haven't done a baseline test yet, **it's a good idea to test the "before" and "after" versions**. At minimum, this means testing two trees – the existing one (to get a baseline score) and our revised tree (to see what improved and what didn't).

As mentioned above, though, **we really should be testing more than one alternative**, so we can be sure our eventual new tree is as effective as possible. Typically, we'll test 2-3 proposed trees against each other (and against the existing baseline tree), then we'll test a "best of" hybrid of the two in a second round.

For more on testing alternative trees against each other, see [The design phase: creating new trees](#) in Chapter 3.

### Which part of the tree?

If we're testing a small or medium-sized tree (say, less than 500 items), we will normally test the whole tree – no major pruning required.

If our tree is larger (say, 500-1000) items, we have 2 options:

- **Test the whole tree** – Easy to prepare, but affects how many tasks we can ask each participant.
- **Test a "pruned" version of the tree** – Takes some effort on our part, but lets us concentrate on the parts we're really interested in.

Finally, if our tree is very large (more than 1000 items), testing the whole tree may be feasible, but in most cases we recommend pruning the tree to keep the participants' effort from becoming onerous.

For more on pruning trees, see [Which part of the tree?](#) in Chapter 6.

---

Next: [Who will we test?](#)

## Who will we test?

- Which user groups?
  - How to recruit?
  - Will we offer an incentive?
- 

Who indeed? Early on, we need to determine:

- Which group (or groups) of users to target (or to specifically exclude)
- How we will invite them to participate
- What incentive (if any) we'll offer them for helping us out

## Which user groups?

Most websites serve **several different types of users**. For example, a toy-store site may get a large number of visits from both children (the toy users) and from adults (the toy buyers). If we're designing the tree for this site, we'll naturally have to create a structure that works for both types of users.

This also means that we should test our tree with both types of users. The adults may be easy to recruit, but how will we get children to participate?

There may also be certain users who we don't want to participate. If we're designing a website only for overseas users, we don't want domestic users cluttering our results (and wasting their own time).

For more on user groups, see [Different user groups](#) in Chapter 9.

## How to recruit?

Anyone who has done user research knows that recruiting always takes a bit longer than expected, so we need to start planning this early.

The two classic ways to recruit for online studies are:

- **Email** – using lists of existing and/or prospective customers
- **Web ads** – invitations posted on our website and/or other related sites

Other methods include commercial research panels and crowd-sourcing sites like Amazon Mechanical Turk.

If we are targeting more than one user group, or if participants are hard to come by, we may need to try a variety of methods.

For more on recruiting methods, see [Chapter 9 - Recruiting participants](#).

## Will we offer an incentive?

**In most cases, yes.** We offer incentives in the vast majority of our studies. Even a modest incentive makes it much easier to get good numbers in a short time, which is a godsend to iterative testing.

We have conducted a few studies where we didn't offer an incentive, but those are special cases.

Because we're only asking for 5-10 minutes of a person's time, it's usually not worth it to reward each participant. Instead, we **offer them a chance to win something they value** (e.g. "5-minute survey – win \$300 in groceries!").

If we're working for a government agency or a large organization, we may need to decide on an incentive early on, because of the lead time needed to get it approved by management.

For more on incentives, see [Offering incentives](#) in Chapter 9.

---

Next: [When will we test?](#)

## When will we test?

---

The experts say that we should do usability testing early and often. Tree testing is no different – indeed, it was created to let us test very early (before we even have a website coded) and very often (because it's both cheap and easy to run a tree test).

In general, **we can start testing as soon as we have a structure to test** – either a text dump of our existing site's IA, or the new IA ideas we've been playing with. We'll also need time to create "find it" tasks that exercise our structure(s), and time for the overhead of setting up the tree tests.

Here's a typical high-level timeline for 3 rounds of tree testing (testing the existing tree, testing our new trees, then testing our even-better-with-revisions "final" tree):

Time required	Activity	Details
(varies)	<b>Earlier IA work</b>	<ul style="list-style-type: none"><li>• User research (surveys, contextual inquiry, etc.)</li><li>• Content inventory/auditing</li></ul>
1 week	<b>Round 1</b>	<ul style="list-style-type: none"><li>• Open card sort</li><li>• Baseline tree test (existing site)</li></ul>
3 days	<b>Create new trees</b>	<ul style="list-style-type: none"><li>• Try alternative groupings and terms</li></ul>
1 week	<b>Round 2</b>	<ul style="list-style-type: none"><li>• Test new trees against each other</li><li>• Compare to existing tree's results</li><li>• Pick best tree and revise</li></ul>
1 week	<b>Round 3</b>	<ul style="list-style-type: none"><li>• Test revised tree</li><li>• Revise and finalize based on results</li></ul>

If we're just planning 1 or 2 rounds of testing, it should be easy to take this and cut it down to what is needed.

---

Next: [Which tool will we use?](#)

## Which tool will we use?

- [Testing online with commercial tools](#)
- [Testing with paper cards](#)
- [Other tools](#)

While the vast majority of tree testing is done online using dedicated tree-testing software, we can also test on paper (the original medium) or by using quick-and-dirty (and free) methods.

## Testing online with commercial tools

As tree testing has matured as a recognized IA technique, online tools have sprung up to meet demand. All of these tools offer advantages over doing a tree test manually, such as:

- **Unmoderated testing**  
People only need a web browser to participate - anywhere and any time, without us having to be present.
- **Flexible testing options**  
As the test administrator, we can customize how the test is run.
- **Automated analysis**  
This is the big one. The software records the participants' actions, and automatically summarizes/visualizes the results.

Here's a list of the commercial tools we know of (listed alphabetically), with some basic information on each:

Product	Owner	Starting at...	Summary
<a href="#">C-Inspector</a>	<a href="#">Steffen Schilb</a>	\$99/study	Introduced in 2008, but discontinued in 2014.
<a href="#">Treejack</a>	<a href="#">Optimal Workshop</a>	\$109/month	Introduced in 2008 as a companion to the <a href="#">OptimalSort</a> card-sorting tool.
<a href="#">UserZoom tree-testing module</a>	<a href="#">UserZoom</a>	\$19,000/year	Aimed at larger enterprises, the UserZoom suite of tools includes tree testing.

Disclaimer: While most of our experience is with Treejack, this guide does not recommend a particular product, because the right product will depend on factors such as budget, compatibility with other tools we're already using, and the specific features needed.

## Testing with paper cards

For pointers on running a tree test using index cards (the original method), see [Tree testing on paper](#) in Chapter 15.

## Other tools

Besides paper, designers have come up with several other "home-grown" ways to test site structures. Most of these involve building an expandable tree in some existing tool, then manually tracking participants' clicks through that tree. Some examples include:

- **Folder clicking**  
In an OS file manager (e.g. Windows Explorer), we create a tree of folders that represent our headings. Then we present participants with tasks as usual, and jot down which path they click through the folder tree.
- **Site maps produced in web editors**  
Some HTML editors (such as Dreamweaver) allow us to create interactive site maps. As above, we can then give tasks to our participants and see where they click as they move down the tree.
- **Simple HTML prototypes using tree widgets**  
Several JavaScript libraries offer tree controls that present a clickable, expandable tree of text. This makes it possible to quickly create a site tree that we can test manually.
- **Custom tools**  
The folks at [Sense/Net](#) created their own tree-testing tool to help them redesign their website. Check out their articles on [deciding on tree](#)

testing, the tool itself, and the results they generated.

---

Next: *Where will we test?*



## Where will we test?

- [Testing online with commercial tools](#)
  - [Testing in person](#)
- 

### Testing online with commercial tools

Most tree testing is now done online, so the “where” is really the participant's PC or tablet or phone, wherever it may be. While their device is out of our control, we can check some things up front:

- Do we know of **any firewalls or network restrictions** that will hamper certain participant groups in our audience?  
This is most common when targeting users in a large organization.
- Are our participants running a **compatible browser**?  
Again, this is mostly a problem with large organizations who may be locked into older browser versions, or with users who have older mobile devices.

For more on dealing with this, see [Checking for technical problems](#) in Chapter 10.

### Testing in person

If we're testing on paper, or doing an online test face-to-face with a participant (see [Asking “why?” with in-person sessions](#) in Chapter 11 for reasons why we would want to do this), then the “where” becomes important.

While a full discussion of how to host participants is beyond the scope of this guide, we offer a few tips in Chapter 9's [Recruiting for in-person sessions](#).

---

Next: [Who will do what?](#)

## Who will do what?

---

We may be tree testing by ourselves, with a colleague, or we may have an entire project team at our disposal. Regardless, the following roles naturally emerge, so it's a good idea to be clear about who's covering what:

Role	Responsibilities
Sponsor	Pays for the testing and represents our interests among upper management
Project leader	Makes the final decisions on scheduling, goals, etc.
Information architect	Designs the trees and tasks, analyses the results, ultimately responsible for the final site structure
Recruiter	Finds the participants, either using customer lists, web ads, or customer panels
Worker bee	Handles the details (writing out cards, entering data in an online tool, etc.)

---

Next: [How will we handle problems?](#)

## How will we handle problems?

- [Testing online](#)
  - [Testing in person](#)
- 

Anyone who has done user research is familiar with [Murphy's Law](#) – the participant doesn't show up on time, the test equipment fails (it was working this morning!) – and the list goes on.

### Testing online

For online testing, the main things to cover are:

- **Piloting our test** to shake out mistakes in the tree and the task wording
- **Providing a contact** for participants who encounter problems
- **Alerting our support channels** that we're doing a study, in case participants call to check that it's legitimate.

For more on handling problems during a test, see [Checking for technical problems](#) in Chapter 10 and [Monitoring the test's progress](#) in Chapter 11.

### Testing in person

For tips on testing people in person, see [Recruiting for in-person sessions](#).

---

Next: [Documenting our plan](#)

## Documenting our plan

- [A planning questionnaire](#)
  - [A sample project checklist](#)
- 

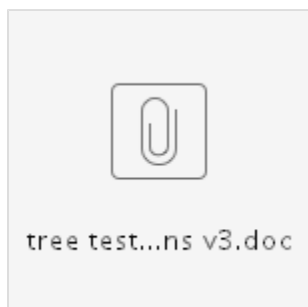
If all this sounds like a lot of work, don't worry – it's not. Most of this is very straightforward, and we'll be able to fill in most of the answers in a single sitting (or a single kick-off meeting, if we're working with a team).

Whether our tree-testing plan is simple or more complex, we do recommend:

- **Collaborating with the team on it**, so everyone's in the loop (and can be involved as they want to be)
- **Writing it down and sharing it**, so that everyone can keep updated as the project moves along (and sometimes changes as it goes). We typically use a project spreadsheet that we fill in as we go, shared in real-time using Google Drive, but the exact tools used matter less than the fact that it's documented and "lived in" by the team.
- **Reusing it next time**, adding or deleted steps as we go, until we get a process that is tailored to our situation.

## A planning questionnaire

When you're planning a tree test, you may find it helpful to use this simple questionnaire to fill in the answers to the questions posed throughout this chapter:

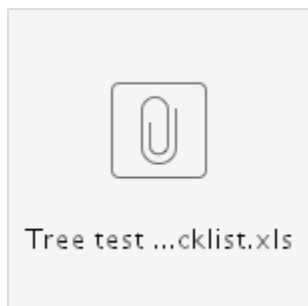


## A sample project checklist

Over the course of hundreds of tree tests, we've refined our work down to a detailed checklist of everything we do in a tree-testing project.

Note that this is a superset of everything that might apply; we normally start from this template and delete the items that don't apply to a specific project.

Note also that this is a very detailed checklist, with lots of items that may not make sense yet. Not to worry - they will be explained in the subsequent chapters of this guide.



---

Next: [Chapter 4 - key points](#)

## Chapter 4 - key points

**A bit of planning** up front will save us time, effort, and frustration later.

We can use tree testing to get a **baseline score** for an existing website, **test revisions** to an existing structure, or **make new structures compete against each other** and the original site.

We should plan to **test our entire structure** unless it's large (> 500 items), in which case we can test a pruned version of it.

Early on, we'll need to determine **which audiences to target**, how we will reach them, and what incentive (if any) we'll offer them.

We should **start testing as soon as we have a structure** written down.

Will we only **test online**, or will we need to supplement our results with some **in-person sessions**?

To run a good study, there are **several roles to fill**. Will we do them all ourselves, or do we need to involve other people?

**If something goes wrong** during testing, we need to plan how we will handle it.

**Documenting our plan** makes everything easier, especially if we have a template to start from.

---

Next: [Chapter 5 - Creating trees](#)

## 5 - Creating trees

*"A place for everything, and everything in its place" - Charles A. Goodrich*

Tree testing starts with...of course...a tree.

By "tree", we mean the hierarchical structure of our website or information space, represented as a multi-level list of headings:

7			
8	<b>BikeSite</b>		
9		For commuters	
0			Bike routes
1			Skills training
2			Cycling and the law
3			Tips for commuting
4		For families	
5			Places to cycle
6			Learning to ride
7			Safety tips
8		For roadies	
9			etc.
0		For mountain bikers	
1			etc.
2		For cycle tourers	
3		News & events	
4			

We may be testing an **existing tree** (e.g. the structure of our existing site) or trying out some **new trees** (revised, or completely rethought) to see how well they work.

While a full discussion of how to create site trees is beyond the scope of this guide, we'll cover the basics and provide links to other good resources.

### Basing new trees on research

Knowing our users, content, and what needs fixing is crucial

### Roughing out alternative trees

Jotting down just enough to triage

### Common schemes to organize sites

By audience, activity, topic, department, brand, etc.

### Picking candidates to test

Triaging ideas, fleshing out candidates, checking coverage

### Combining and flipping schemes

Trying permutations of level-1 and level-2 headings

### Posing questions about tree elements

Testing specific alternatives using specific tasks

### Wide/shallow vs. narrow/deep

Wider, shallower trees usually work best (but have their own problems)

### More on creating trees

Where to find more tips on creating site structures

### Labelling and terminology

### Key points

Speaking the user's language, keeping headings clear and distinguishable, etc.

## Team-sourcing ideas

Involving the team nets us more (and different) ideas

---

**Next:** [Chapter 6 - Preparing a tree for testing](#)

## Basing new trees on research

---

Designing the structure for a website is never a lockstep 1-2-3 process, and different designers will approach it in different ways.

It's always good, however, to start with some knowledge. In [The research phase](#) in Chapter 3, we described how some basic user research (such as contextual inquiry, open card sorts, and tree-testing the existing site) can give us a head start.

If we've done the research, **we are not starting with a blank slate**:

- 1. We know our users.**  
Our initial discussions with stakeholders defined who our target audiences are, and some basic research determined what they want and how they go about it.
- 2. We know our content.**  
A content audit has determined which topics we're keeping, updating, adding, and deleting.
- 3. We know what's right and wrong with the existing structure.**  
A baseline tree test has shown us which parts of the existing site tree work well (and should be reused) and which parts perform poorly (and need to be changed).
- 4. We have new ideas to try out.**  
An open card sort has given us ideas about how users mentally group our content and which terms they use.

We can now sketch out some new structures to test. Let's take a look at the most common ways sites are organized, and some common tactics that can help us create our own trees.

---

Next: [Common schemes to organize sites](#)



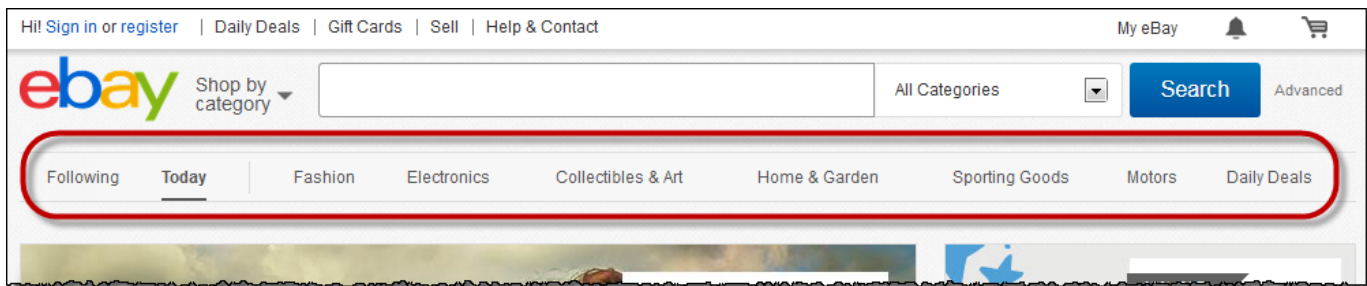
## Common schemes to organize sites

- By topic
- By activity or task
- By audience or role
- By brand
- By geography
- By internal department
- By format

The way we organize a site depends, obviously, on its content. But our site is probably similar to others already out there. Here are some of the most common ways that websites are structurally grouped:

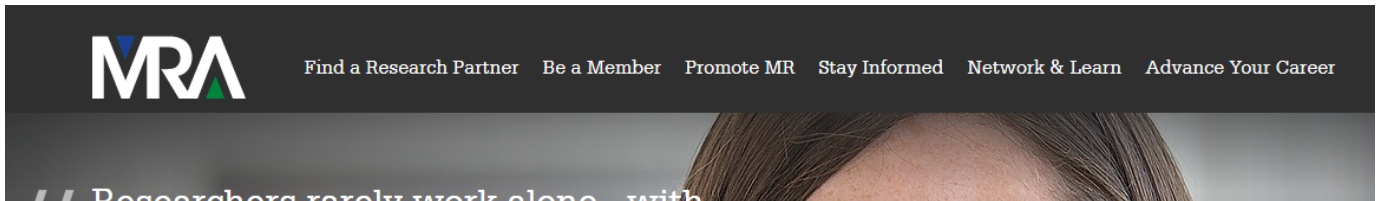
### By topic

Topic-based navigation is very common at level 1 and 2, where the content varies widely and it's critical to get the user to the right section before they can start their task.



### By activity or task

Grouping content by activities and tasks is common, especially on lower levels of a website. It takes advantage of the fact that most users arrive with a specific task in mind:

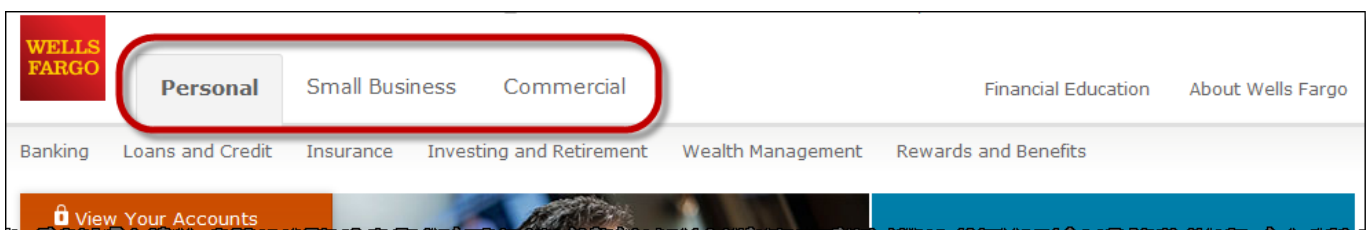


Note that activities (e.g. "Administering a school") often resemble audiences and roles ("School administrators"); sometimes it is just a question of phrasing.

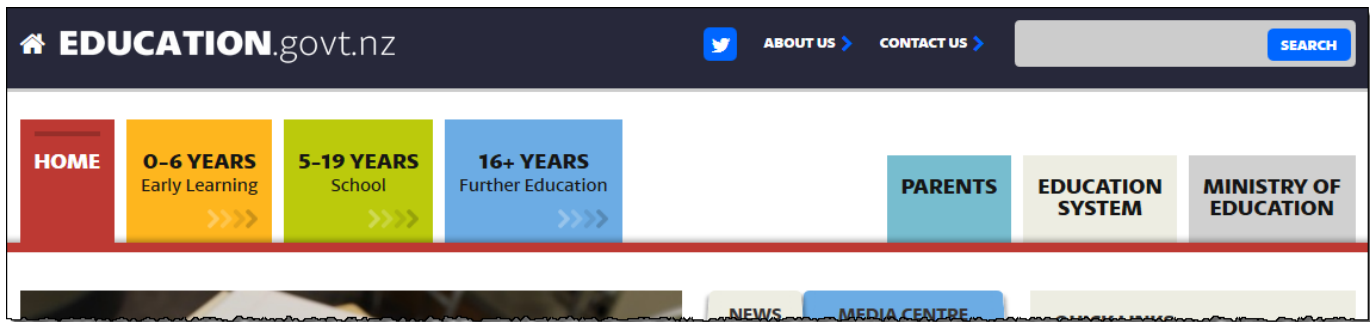
Note also that activities often mix well with topics (see above). Consistency is less important here than effectiveness.

### By audience or role

When a site caters to several audiences, and the content is different for each, this is a popular choice. For example, a bank may divide its site into sections for personal banking, small business, large corporations, and so on.



Sometimes the role is age-based, such as this example from the NZ Ministry of Education:



Usability guru Jakob Nielsen warns about the [problems with audience-based navigation](#), but if done properly, it can be effective.

## By brand

On shopping sites, dividing the content by brand is sometimes used as primary navigation, but more often used in a secondary role or as an alternate way to filter:



## By geography

Where content varies largely by region, it may be a good idea to divide content this way. However, this does assume the user knows the regions by name, and it also means grappling with geographical boundaries and content that does not fit into a specific region.

## By internal department

Grouping content by internal parts of an organization (e.g. Human Resources, Finance, IT, etc.) is common in intranets, but is usually not a good idea for public-facing websites (where site visitors may not understand (or care about) the organization's internal structure).

## By format

Some sites offer sections devoted to certain formats, such as documents (often PDFs), videos, or picture galleries, often using their own jargon (where, for example, "papers" are PDFs, "guides" are HTML, and "tutorials" are video).

This can make things hard to find, because a site visitor with a task in mind may not know which format the site has used for that information.

For example, if we're at the Home Depot site looking for help on patching a roof, do we try "Project guides", "How-to advice", or "How-to videos"?

---

Next: [Combining and flipping schemes](#)



## Combining and flipping schemes

- [Combining schemes](#)
- [Flipping schemes](#)

When first creating a site tree, we usually play with various kinds of top-level groups, using some of the variations described in [Common schemes to organize sites](#) earlier in this chapter.

### Combining schemes

A very common tactic is to combine some of these schemes as first- and second-level headings. For example, we may use *audiences* as our primary navigation, then *topics* within each of the audience sections:

<b>For parents</b>	
	Primary schooling
	Secondary schooling
	Post-secondary
<b>For teachers</b>	
	Primary schooling
	Secondary schooling
	Post-secondary

We may then decide to try replacing *topics* with, say, *activities*:

<b>For parents</b>	
	Choosing a school
	School transport
	Home schooling
<b>For teachers</b>	
	National curriculum
	Training & upskilling
	Employment & pay

### Flipping schemes

Another common tactic is to flip the primary and secondary navigation, to see if it fits the content better.

For example, the *audience/topic* tree that we tried earlier...

<b>For parents</b>	
	Primary schooling
	Secondary schooling
	Post-secondary

<b>For teachers</b>	
	Primary schooling
	Secondary schooling
	Post-secondary

...could be flipped to become an *topic>audience* tree:

<b>Primary schools</b>	
	For parents
	For teachers
	For administrators
<b>Secondary schools</b>	
	For parents
	For teachers
	For administrators

We may do the flip, think about it, and decide that it won't work for our purposes (perhaps the content doesn't fit as well, or we're sure that it will be confusing for users).

But, if it looks reasonable, and we're not sure how well it will work with users, it's probably worth testing both versions in side-by-side tree tests.

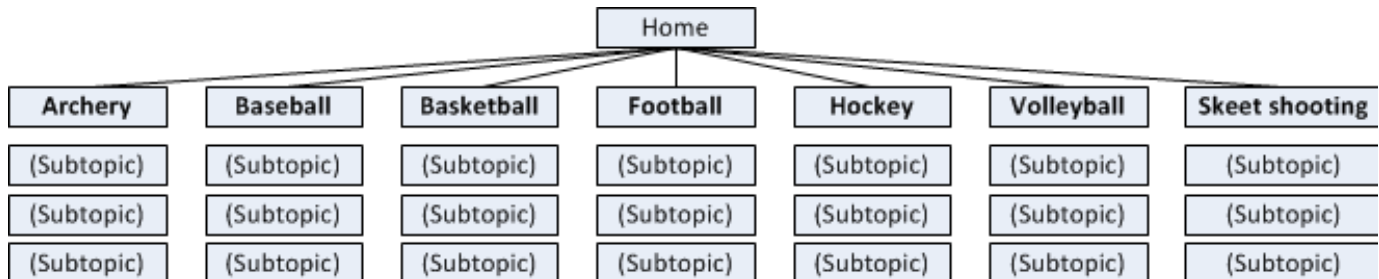
---

Next: [Wide/shallow vs. narrow/deep](#)

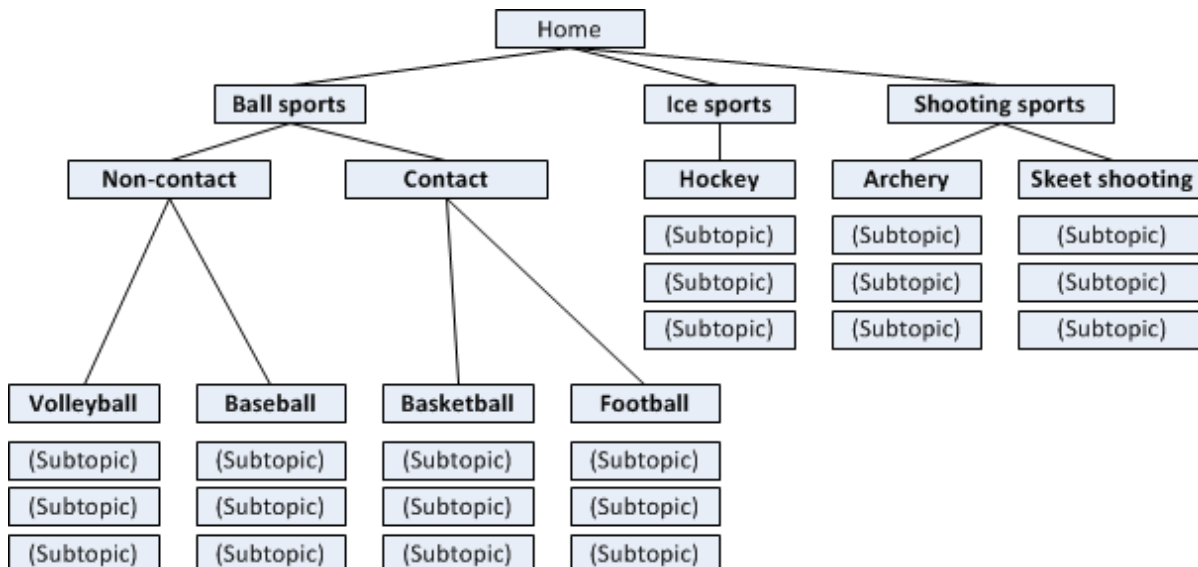
## Wide/shallow vs. narrow/deep

When creating site trees, a common question is “Should we create a shallow tree with lots of headings at the same level, or a deep tree with fewer choices at each level?”

For example, here’s a tree designed as wide and shallow:



The same topics could also be modelled as a narrow, deep tree:



In most cases, it’s best to go with a shallower, wider tree because:

- With many headings to choose from, the user may be able to spot the keyword they’re looking for immediately, without having to click down a level.
- In a narrow/deep tree, trying to keep the numbers of headings small at a given level sometimes means that we have to create headings that are abstract, and this makes it hard for users to understand what those subsections contain.
- If the tree is too deep, it’s more likely that users will get lost or discouraged, because they have to do more “navigation work” before they get to the actual content.

Note, though, that **wide/shallow trees can cause problems if they’re too wide:**

- A huge number of headings at a given level can make tiring for the user to skim and choose.
- Lots of headings can make it harder to keep them all clear and distinguishable.
- Too many headings may make it hard to fit them all in the site’s navigation bar.

For more on this, check out Kathryn Whinton's excellent article on [Flat vs. Deep Website Hierarchies](#).

---

Next: [Labelling and terminology](#)

## Labelling and terminology

- Speak the user's language
- Be wary of brand names
- Make headings unambiguous
- Make headings distinguishable
- Use specific, concrete terms
- Make headings quick to skim
- Balance brevity with clarity
- Combine entangled topics

Besides **organization**, the other IA element that tree tests focus on is **labelling** - the specific words we use in our headings.

When we run a tree test, we are seeing the interaction of these two factors:

<b>Organization</b>	If a user can't navigate down to the right heading, it doesn't matter how hard we worked to make that heading clear.
<b>Labelling</b>	If a user doesn't understand a certain heading, they're unlikely to click on it to see its subheadings

Some labels are dead easy to create, while others seem to get harder the more we tinker with them. What is it that makes one label better than another?

Below are some principles and tips to create effective headings.

### Speak the user's language

The most important thing we can do when phrasing headings (and content in general) is to **use the same terms that our audience uses themselves**.

For example, if we create a section called *Contingency planning*, and our audience generally has a high-school education, we should replace it with a more understandable term like *Emergencies*.

**Note that "speaking the user's language" is not the same thing as the common advice to "avoid jargon"**. If our audience regularly uses jargon themselves (for example, programmers who are comfortable with terms like *AJAX* and *hypervisor*), then we should consider using those terms in our headings and our content. While jargon is often opaque for outsiders, it is efficient and precise for insiders.

Consider the following example from a bus website:



- If we're new to this bus company, we probably don't know the different between *Flexi-Pass* and *TravelPass*, so we would have to visit both pages to find out.
- However, if we're regular customers, we are probably familiar with these terms, so navigation is easy.

If we're not sure which terms our audience uses, there are several ways to find out:

- If we run an [open card sort](#) to generate ideas for our site tree, we can examine the headings that our participants create.
- If we do [contextual inquiry](#) or other qualitative research with users, we can review our notes or recordings for verbatim terms that they use.



- Check our **search logs** to see what our site visitors are typing into the search box. This has the double value of showing what our users may not have been able to find by browsing, and what words they use during navigation.

## Be wary of brand names

Organizations love to come up with catchy or cute names for their products or services. And this is not limited to commercial products; government agencies also have a long history of creating programmes with catchy "marketing-speak" names, such as *StudyLink* (for student loans) and *Keeping Connected* (a transport portal).

The problem here is that while the organization knows what these labels mean, it often assumes (wrongly) that its users know too.

Brand names used by themselves in headings can cause the following issues:

- **Site visitors may not know what the brand name means.**  
"I have no idea what *LUCAS* is." (a university admin system)  
For more on heading clarity, see below.
- **They know what they want, but not its brand name.**  
"Where's the portal login?" (hidden behind a *Keeping Connected* link)
- **Brand names may be hard to distinguish from each other.**  
"What's the difference between the *At Home* package and *HomePlan*?"  
For more on distinguishable headings, see below.

## Make headings unambiguous

In addition to being understandable, effective headings are also **unambiguous** - users should not be asking which of several meanings might apply.

For example, this healthcare site has a section for medical professionals, which they label *Providers*...



...but tree testing and usability testing showed that other audiences (such as businesses) also considered themselves to be "providers", causing some hesitation in choosing the right section.

Changing this label to *Medical Providers* solves that problem, but introduces another, because *Medical Providers* might be interpreted by patients as offering a list of medical providers to contact.

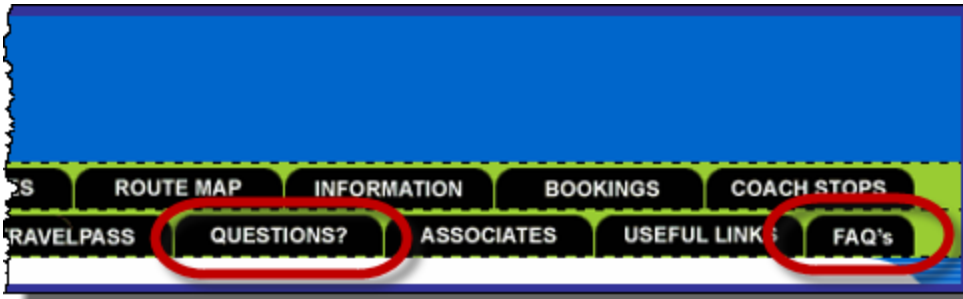
Using a term like *For Medical Providers* is clearer, but alas, this may be too long for the space available - see [Balance brevity with clarity](#) below.

## Make headings distinguishable

Users typically encounter several headings at a time, and **need to be able to successfully choose between them**.

That's why it's critical that we make our headings easily distinguishable from each other.

For example, if we have a FAQ heading, most audiences will know that it leads to a page of frequently asked questions. By itself, it's a clear heading. If, however, we group it with other headings like those shown below...

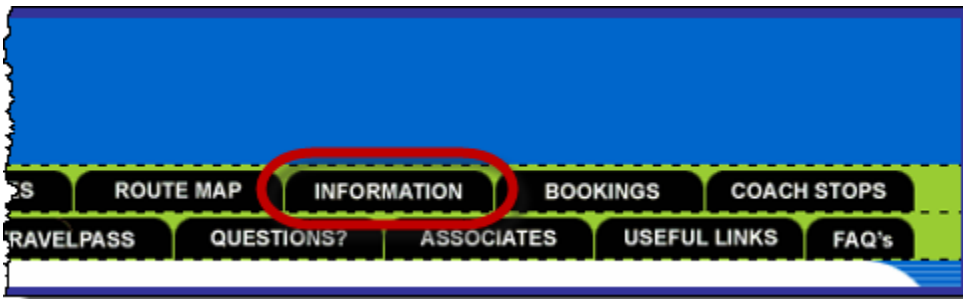


...now it becomes much harder to decide where to go if we have questions.

## Use specific, concrete terms

We should **be as specific as possible when writing a heading**. This helps prevent misunderstanding, and makes headings more distinguishable from each other (see above).

For example, the bus website mentioned above has an *Information* tab. If we asked 5 different people what that tab contained, we would likely get 5 different answers:



Writing a vague heading can also create an **evil attractor** - a link that lures clicks when we don't want it to.

In [Discovering evil attractors](#) in Chapter 12, we see that a consumer-review website tried a *Personal* subheading in their *Appliance* section. In tree testing, participants mistakenly chose it as the answer for a wide variety of unrelated tasks. When they made it more specific (changing *Personal* to *Persona Care*), the problem went away.

## Make headings quick to skim

It also helps to make headings easy to skim, by **putting the most important differentiating words up front**.

For example, this list of headings is typical, but not very scannable:

- Using your debit card
- How to apply for a credit card
- Understanding credit-card limits
- How you can budget for credit cards

Usability guru Jakob Nielsen calls this "Starting with blah-blah and deferring the information-carrying text to the end".

We can easily make this list more scannable by front-loading the headings, like this:

- Debit cards - how to use
- Credit cards - apply online
- Credit-card limits - how they work
- Budgeting for credit cards

For more on front-loading headings and links, see [Nielsen's oft-cited article](#).

## Balance brevity with clarity

One conundrum that we encounter when writing headings is the tug of war between:

- **brevity** (these headings may need to fit in a column or along a nav bar), and
- **clarity** (it's usually easier to be specific and unambiguous if we can add a few extra qualifying words)

There's no magic answer here. Some headings are very clear using a single short word (e.g. *Clearance* in a shopping site), while others require a lengthier phrase (e.g. *For Medical Providers* in a healthcare site), and may still not be as clear as we would like.

When writing headings that will be space-constrained (such as the headings in a horizontal nav bar), **it's a good idea to lay them out roughly before we tree-test them**. The layout can be quick and dirty, using paper and pen, or a mock layout in a drawing app like PowerPoint. We can then determine if we need to shorten the headings. This prevents us from tree-testing a verbose tree and later having to trim it down for the actual website, because shortening our carefully crafted headings is likely to change their effectiveness.

## Combine entangled topics

We sometimes encounter two topics that are intertwined, to the extent that it doesn't make sense to separate them, but it's also hard to think of an overall word for them.

For example, a city-council website may have content about city parks, sports fields, and play areas, that they want to put in a single section of the site:

- *Parks* by itself may not be general enough to grab people who want to know about sports.
- *Recreation* may not be where people look for bike-commuter trails.
- There may not be an obvious word that covers both of these.
- The solution, of course, has now become commonplace - *Parks & Recreation*.

This tactic - taking the best descriptors of each topic and combining them - can be a very effective way of combining overlapping content without having to "invent" some abstract word that hurts findability.

Consider an organization that publishes position papers and monthly bulletins. They want to combine these in a single section, but what to call this section?

- *Resources* is a common solution, but usually a poor one, because it means too many things to too many people, and is likely to become an *evil attractor*.
- *Publications* is better, but is still a bit vague, and may not be a likely target for those looking for news articles (which are published in their bulletins).
- *Papers & Bulletins*, while longer-winded, actually tests better than the others, because it's specific about the two major types of content in that section.

---

Next: [Team-sourcing ideas](#)

## Team-sourcing ideas

If we're working with a project team, we have an advantage in generating ideas for site trees – **other people's minds**.

Different people have different backgrounds, different knowledge, and different biases, and if we share our IA research with them, they are likely to come up with new and useful ideas for our site structure.

When we're working with a project team, we typically do the following:

1. Distribute the results of our research (e.g. findings from our card sort and other related studies).
2. Meet to review the research and show everyone how to “rough out” a site tree (see below).
3. Ask each person to take an hour at their desk to rough out 1 or 2 tree ideas.
4. Meet again to review everyone's trees.  
We start with “show and tell”, then we discuss the various approaches and pick 2-3 trees that we want to pursue.
5. Assign a person to each candidate tree to flesh it out for testing.

For easy collaboration, we use a shared spreadsheet (e.g. Google Sheets or equivalent) with:

- A tab for the existing site structure (if any)
- A tab with a list of the content areas that the site needs to cover (from our content audit)
- A tab named after each person
- Later, tabs for the candidate trees that we will flesh out

The screenshot shows a Google Sheet with five columns labeled A through E. The content is organized as follows:

	A	B	C	D	E
	Home				
		Products			
			SuperWidget		
			MegaGadget		
			UberThingie		
		Case studies			
		Support			
			SuperWidget		
			MegaGadget		
			UberThingie		
			Billing		
		FAQ			

At the bottom of the spreadsheet, there are several tabs: '+', a menu icon, 'existing site', 'content areas', 'Bob', 'Carol', 'Ted', and 'Alice'. The 'existing site' tab is currently selected.

Using a shared spreadsheet means that we don't have to email spreadsheets to each other, and each person can “peek” at what others are doing to help them get going.

---

Next: [Roughing out alternative trees](#)

## Roughing out alternative trees

When we “rough out” a site tree, it means that we **create a quick-and-dirty site structure in an hour or less**.

The point of this is to do the **minimum work to convey an idea to others**, so we can then decide whether it’s worth fleshing out enough to test. We don’t want someone to spend an entire day creating a detailed site tree, only to have the team decide that they’re going a different direction.

To rough out a site tree:

- We **only create 2 to 3 levels** of the tree, even if we know there will be more sublevels. We can create the sublevels later if the tree survives our triage process.
- We **don’t get stuck on wording**, especially for the lower levels. At this point we’re looking more for variety than exact terminology.
- We **do a quick coverage pass** to make sure we’ve accounted for all the content the site will include, but again, if we miss something, we can fix it later.

level 1	level 2	level 3	
Feature films			
	News & events		
	Funding		
	Film-making		
	Marketing		
	Contacts		
	Features - past, present, future		
Short films			
	News & events		
	Funding		
	Film-making		
	Marketing		
	Contacts		
	Shorts - past, present, future		
Documentaries			
	News & events		
	Funding		
	Film-making		
	Marketing		
	Contacts		
	Documentaries - past, present, future		
About us			
etc.			

Next: [Picking candidates to test](#)

## Picking candidates to test

- [Triaging the draft trees](#)
  - [Fleshing out the candidates](#)
  - [Checking coverage of content](#)
- 

Drafting a bunch of possible trees is a great thing to do, because it makes us entertain new (and sometimes unconventional) ideas, and because it makes us think hard about what users will do.

We could test all of these draft trees, and if we only have 2 or 3 of them, that's probably the right thing to do at this early stage.

However, if we've been creative and come up with a larger number of trees, **it's best at this stage to do some triage** – review the trees and decide which are the most promising (and feasible, given our content).

### Triaging the draft trees

We typically do this “tree triage” by:

- Circulating the various draft trees to our project team, so they have time to review them
- Running a short workshop to discuss the various ideas and pick the trees to pursue

In the triage workshop (which is 60-90 minutes long, depending on how many trees we need to review), we:

- Briefly review the draft trees and their main ideas about grouping and labeling
- If necessary, combine ideas from some of the trees into a new tree that seems more promising than the originals.
- **Pick 2 or 3 trees that look like the best candidates** to test with users

At this point, we usually give **standardized names to our candidate trees** so we can easily identify them later. For example:

- We pick one of Michael's trees – the one that features user groups at the top level – and rename it from “Michael users” to “tree 1”.
- Sarah's tree, which uses broad activities as the top level, is renamed from “Sarah” to “tree 2”.
- And so on.

### Fleshing out the candidates

Now that we've picked the trees we want to test, we need to flesh them out.

Earlier, we recommended that when “roughing out” a tree, create the top 2 or 3 levels, and don't get stuck on wording. This is time saved for the trees that we discarded during our triage.

For the trees that made the cut, we now need to think through the whole tree. This means:

- **Expanding the tree to include all the sublevels we want to test.**  
For most trees, this will be all the sublevels that the eventual website will have. For some trees, however (particularly shopping sites that may have thousands of products), we may decide to test a representative sample of the lower levels.  
For more on how many levels to test, see [Which part of the tree?](#) in Chapter 6.
- **Making our labels consistent.**  
This is the time to clean up our wording so that we use **parallel phrasing** (e.g. “For patients”, “For doctors”, etc.) where possible, and stricter terminology (e.g. Are we calling them “cars” or “vehicles”?).

### Checking coverage of content

Once we have decided on which trees to test, we also need to **make sure that they represent all the content** that we plan to include in our website.

It's likely that we've covered all the major topics, but how can we be sure we've covered the minor ones too? Remember, if we miss something here, then discover it later (after testing, or even worse, during development), it may be painful to retrofit. Better to take the time now to make sure.

There is no magical way that we've discovered to check coverage of content. It's simply a matter of **comparing, line by line, our list of planned content to our candidate trees**, and making sure that each tree accounts for all the content.

---

Next: [Posing questions about tree elements](#)



## Posing questions about tree elements

When we are roughing out draft trees, and later when we fill out the candidate trees we've chosen to test, **questions and ideas naturally pop up**. We may encounter thoughts like these:

- Should we combine "Doctors" and "Nurses" into a single section? Could we call that "Medical"?
- Should we list all 12 topics, or just the 6 main ones plus a 'More' link?
- Should we call that section "Electronics" or "Technology"?
- Will our audiences know what "Contingency planning" means?
- ...and dozens more like these.

**Some of these thoughts may make us stop and rough out a new tree** immediately to see if it's worth considering. That's good, because it's still early days and we should keep entertaining fresh ideas as they happen.

Some questions will persuade us to **test specific alternatives using our candidate trees**. For example, we might decide to name a section "Electronics" in tree 1, and "Technology" in tree 2, and see which performs better in our testing.

Finally, **some questions will suggest tasks** that we should pose to users in our tree tests. We wondered if users would understand "Contingency planning", so we should write a task that asks them to find a topic in that section (e.g. how to prepare for an earthquake).

As these questions and ideas surface, **it's a good idea to keep a running list** of them so we can address them (either by discussing and discarding them, or by incorporating them into our trees or the tasks we will pose to users during testing).

note	decision
goal - get everyone into the right buckets first time	
goal - keep strays away from "About ACC". Will clarifying the other sections solve this?	Change to "About Us", and rename other sections.
keep current top-level categories (client, business, providers, IP), but can rename. Everything else is fair game	done
Claims & care - needs rewording because of generic "claims" meaning? Does it still attract unwanted visits from business and providers?	test "Making a claim"
Change "For Providers" >> "For Healthcare Providers"?	test "For Providers", if not clear then change to "For Healthcare Providers".
rename IP to Safety? Or some combination?	try "Preventing Injuries"
Should Publications be a global utility link, a main tab, a quick link, or what?	Try keeping it as a header link
make News more prominent?	added top-level news, and news in each customer section
expose one primary IA (as opposed to the various navigation lists shown now)	
standardize wording - "make a claim" vs. "making a claim" vs. "How do I make a claim" vs. etc.	
test entire tree for all 3 customer groups (requires us to have all IA done	

**Next:** More on creating trees

## More on creating trees

---

We've covered some basics of creating site structures here, but it's a bigger topic that others have written extensively (and well) about. Here are some suggested readings:

	<a href="#">Information Architecture For the Web and Beyond (4th edition)</a> by Louis Rosenfeld, Peter Morville, and Jorge Arango
	<a href="#">A Practical Guide to Information Architecture (2nd edition)</a> by Donna Spencer

---

**Next:** [Chapter 5 - key points](#)

## Chapter 5 - key points

Tree creation should be based on not only our content, but our **knowledge of our users**, our **research of their mental models**, and our **testing of the existing site structure**.

We should consider whether the **classic grouping schemes** (by task, by audience, by topic, and so on) suit our needs for the top levels of our site.

We should consider whether our structure is best designed as **wide/shallow or narrow/deep**.

When writing headings, speak the user's language, be specific, and ensure that headings are clear and distinguishable.

We should **create several candidate trees** (ideally team-sourced) so we can pick the strongest candidates to test against each other (and the existing structure).

We should make sure that our tasks (described in Chapter ~) **target the differences between our trees**, so our results can help us pick the winning design.

---

Next: [Chapter 6 - Preparing a tree for testing](#)

## 6 - Preparing a tree for testing

*"Give me six hours to chop down a tree and I will spend the first four sharpening the axe." - Abraham Lincoln*

We may be testing an **existing tree** (e.g. the structure of our existing site) or trying out some **new trees** (revised, or completely rethought) to see how well they work.

In either case, our tree will need to be prepared for testing. In many cases, this is just a matter of **minor housekeeping** (importing the tree into a spreadsheet, excluding certain headings like "Search", and so on), but sometimes we need to make some **bigger decisions** about our tree (e.g. whether to prune a large tree).

In this chapter, we'll cover everything we need to consider when preparing a tree for testing.

---

### Working in an electronic format

Using a spreadsheet or text file, sourcing the content, using the right format

### Which part of the tree?

How to cut big trees down to size for testing

### Which headings to include/exclude?

Global nav, footer links, utility links, etc.

### Spotting missing content

Content that doesn't have a page of its own might be missed

### Dealing with shortcuts and duplicated content

Handling topics that appear in more than one place

### Breaking up double-level topics

Handling multi-level navigation (e.g. mega menus)

### Using link names instead of page titles

Link names represent the nav better than page titles do

### What to call "Home"

In certain cases, "Home" may cause problems

### Transferring the tree to a testing app

Sanitizing text, importing to a tool, and randomizing subtopics

### Key points

---

**Next:** [Chapter 7 - Writing Tasks](#)

## Working in an electronic format

- Using a text file
- Using a spreadsheet
- Sourcing the tree
- Using the right format

First, we need our tree in a form where we can easily work with it – typically, a simple text-document format on a computer. If we're working with an online tree-testing tool, this will also let us easily copy our finalized tree into the tool later.

(While we can prepare our tree directly in the tree-testing tool, this is usually slower and fussier than working in a dedicated document app first.)

### Using a text file

Some people use plain ASCII text files (e.g. **tree1.txt**), while others use a word processor (e.g. **tree1.doc**).

Regardless of the file format we use, **each heading must be a separate line of text**, and its level in the tree is shown by indenting with spaces or (even better) tabs:

```
BikeSite
  For commuters
    Bike routes
    Skills training
    Cycling and the law
    Tips for commuting
  For families
    Places to cycle
    Learning to ride
    Safety tips
  For roadies
    etc.
  For mountain bikers
    etc.
  For cycle tourers
  News & events
```

### Using a spreadsheet

While either of these will work, **we recommend using a spreadsheet** instead, because:

- The separate columns make it easier to see which level a given heading is on, and to make sure we haven't accidentally skipped a level.
- We can include several trees (e.g. the baseline tree and some revised versions) in a single spreadsheet by using the page tabs.
- Most importantly, we can add columns to annotate the tree. In the screenshot below, we've added columns for *Tasks* and *Notes*.

	A	B	C	D	E
1	Level 0	Level 1	Level 2	Tasks	Notes
2					
3	BikeSite				
4		Commuting by bike			
5			Bike routes		
6			Skills training		
7			Cycling and the law		
8			Tips for commuting		
9		Recreational/family			
10			Places to cycle		
11			Learning to ride		
12			Safety tips		
13		Road training/racing			
14			etc.		
15		Mountain biking			
16			etc.		
17		Touring & camping			
18		News & events			
19					

If we're working alone, we usually just use our favorite spreadsheet program (e.g. Microsoft Excel, Apple Numbers, etc.).

However, **if we're collaborating with others, we highly recommend using an online spreadsheet** (such as Google Sheets) that provides multi-user editing. This lets everyone stay current and contribute without having to email files around and manually consolidate changes later.

Here's an Excel template to get started:



A few notes about our spreadsheet template:

- The first column is reserved for the **root node** of the tree, so it only has a single entry, typically labeled "Home" or "Top", but we can rename it to whatever suits our site.
- We've included 2 levels, but add or remove columns as required.
- **Each heading is on its own row.** A common error is to start subheadings on the same row as the heading, but some tree-testing tools will get confused by this.
- The *Task* column lets us **pencil in ideas for tasks** we'd like to use to exercise a particular part of the tree. By writing tasks here in the tree, it also provides a quick way to check which parts of the tree we're testing. For more on this, see [Chapter 7 - Writing tasks](#).
- The *Notes* column gives us a place to **jot down comments, issues, or to-do's** for later work. If a note is particularly important (e.g. we think it should be discussed at our next meeting), we color-code its cell to make it stand out.
- The *Assigned to* column tracks which team member is responsible for each section of the tree.
- We can add more columns as needed. For example, if some tasks may only apply to some user groups, we may want to add an *Audiences* column to track this.

For the remainder of this guide, we'll use the spreadsheet method, although these ideas should be easy to adapt to other methods as well.

## Sourcing the tree

If we're dealing with a small tree, and we don't already have it in a form that can easily be converted to a spreadsheet or text file, it's sometimes easiest to just type it manually into our spreadsheet.

However, most tree testing is done on not-small sites, where findability becomes a real problem to solve. For these sites, typing the site structure manually is tedious. It's usually better to **find an existing map of the site** and figure out how to convert it to the format we need.

To the best of our knowledge, there is no standard file format or schema for mapping site structures. (Even Google's **sitemap** file format does not include hierarchy information.) Because of this, importing our existing tree depends on where it's coming from:

- If we're using a **Content Management System (CMS)**, there is usually a way to dump the site structure to a file – ideally to a spreadsheet or CSV file, but even a plain text file is better than starting from scratch. From there, we can then copy/paste the content into our spreadsheet and massage it into shape.
  - If the CMS exports a tree structure, check that it matches the actual hierarchical navigation of the site. (In some CMS's, the menu structure and the CMS structure can differ.)
  - If the CMS provides an option to export page titles or link names, choose the latter because they are what users see when they're deciding where to click.
- Many designers model their site trees using **mind-mapping software** (Mindjet, FreeMind, etc.) because of its ability to quickly build, revise, and visualize a hierarchy. Like CMSs, these programs typically provide a way to export the map to a text file using indentation, which we can easily import into a spreadsheet.

## Using the right format

Whichever tool we use to create and edit our site tree, we want to make sure that it can eventually be **imported easily and reliably** into our tree-testing tool.

Luckily, the format itself only has 1 major rule: **One heading per line**

Right			Wrong		
Level 0	Level 1	Level 2	Level 0	Level 1	Level 2
BikeSite			BikeSite	Commuting by bike	Bike routes
	Commuting by bike				Skills training
		Bike routes			Cycling and the l
		Skills training			Tips for commuti
		Cycling and the law		Recreational/family	Places to cycle
		Tips for commuting			Learning to ride
	Recreational/family				Safety tips
		Places to cycle		Road training/racing	etc.
		Learning to ride		Mountain biking	etc.
		Safety tips		Touring & camping	
	Road training/racing	etc.		News & events	
	Mountain biking	etc.			
	Touring & camping				
	News & events				

Many first-timers make the mistake of putting the first subheading of a section on the same line as the section's heading. This makes it harder to move around subheadings while we're revising, and may cause problems when importing to our tree-testing app.

Importing the tree can save a lot of typing, but it's not a silver bullet. **We'll still need to check our tree** for link-name discrepancies and for

problematic or missing content, which we describe in the remainder of this chapter.

---

Next: [Which part of the tree?](#)



## Which part of the tree?

- Size does matter
  - How big is the tree?
  - Pruning levels
  - Pruning unnecessary branches
  - Testing subtrees
- 

### Size does matter

If we're tree-testing on paper, the size of our tree obviously matters because of the physical effort involved. Most designers would balk at the idea of writing index cards for a tree 7 levels deep with 1000 items or more. They would immediately look for ways to cut their tree down to something more manageable for testing purposes.

We might think that online testing gets rid of this problem. After all, web apps can now handle vast amounts of data without bogging down, so why not test the whole tree?

**The main problem with tree-testing very large structures is not technical; it's about the participant's experience of the test.** If they have to search through a huge tree (either wide, or deep, or both), it's going to take them longer for each task, and they're more likely to get tired of the test sooner. That means more abandoned sessions ("Ack, I'm only halfway through this") and less chance that they'll participate in future online studies for us ("I finished, finally, but I'm not doing one of those again").

We can counter this by asking each participant fewer questions, but that requires us to have more participants in total, which can be a real problem if we're targeting very specific types of users.

If we prune large trees for testing, we can provide a better participant experience without giving up much in the way of useful findings. Ideally what we're looking for is a study that is easy to pitch to users ("Do our 5-minute survey – win an iPad!"), quick for them to complete, and leaves them in a mood to say yes to the next study we cook up.

The other reason to consider pruning large trees is the **effort needed to prepare the tree**. If we can cut a few hundred entries out of our tree, those are entries that don't have to be reviewed and cleaned up. If we're testing several trees, the saved time really adds up.

### How big is the tree?

If we're testing a **small or medium-sized tree** (typically less than 500 items), we will normally test the whole tree; it's not big enough to cause any unwanted test effects. We still need to check the tree for specific headings to include/exclude/message (see the rest of this chapter), but we can safely skim or skip the rest of this particular section.

**If the tree is large** (in our experience, that means 500-1000 items), we have two options:

- **Test the whole tree** – This is usually easiest from our point of view (no pruning required), but can degrade the participant's experience as described earlier.
- **Test a "pruned" version of the tree** – We may be most interested in how well the top few levels of our site work, in which case we can limit the test tree to the top 3 or 4 levels. Participants should then have time to do a larger number of tasks in a shorter total time.

We can also cut the tree down to size by selectively pruning branches that are not important to our test.

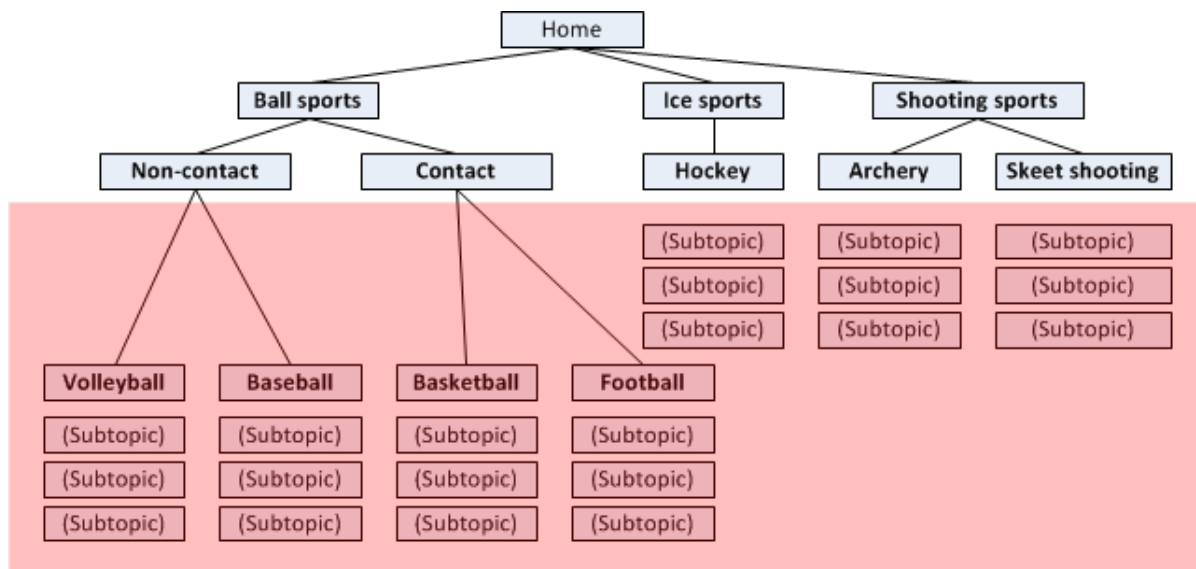
Lastly, we can opt to test only a particular subtree of the whole tree – e.g., the Sports section of a newspaper site. Here again, there are trade-offs, so it depends on what we're really looking for.

Each of these methods is explained below.

Finally, **if our tree is very large** (more than 1000 items), it's almost guaranteed that we'll need to prune it to keep our participants' efforts from becoming onerous.

### Pruning levels

The simplest way to cut a large tree down to a more manageable size is to **cut everything below a certain level**. That is, if we have a tree 5 or 6 levels deep, cut it off at 3 or 4 levels instead (in this example, the topics above the red area):



This is easy, but we must be careful – we’re giving up something by axing the lower levels. In the full tree, we would have been able to see if participants could follow the full path down to the exact answer we intended (and if not, where they strayed off the path). **With a “short” tree, we’ll only find out if they were able to get to the right general area.**

This sounds like a big trade-off, but it really depends on the goals of our particular test. If we’re mainly concerned with the high-level organization of the site, losing the paths through the lower levels may not matter so much, and this type of horizontal pruning may be the best return for our effort.

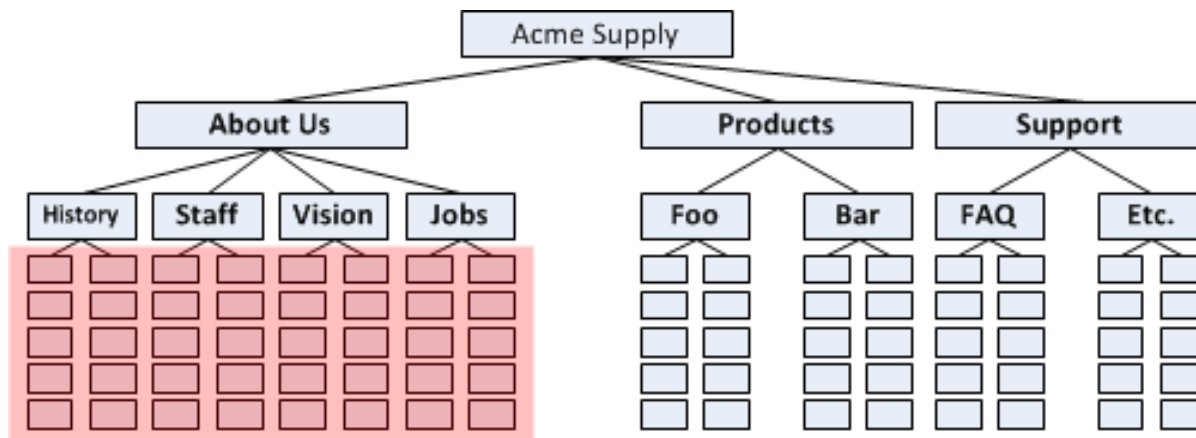
If, however, we do want to know how well our lower levels perform too, then we should consider the other methods described below.

### Pruning unnecessary branches

Instead of pruning straight across a given level of the tree, we may want to be more selective about what we cut. There’s no hard-and-fast rule that says we need to have an equal number of levels across all sections of our site. If there’s a **section that we don’t intend to test** (either because it gets very little traffic, or we already know that it works well), we can chop it off short, while leaving other sections (the ones we really want to test) at their full depth of topics.

For example, suppose we’re testing a new tree for a large corporate site, and we’ve decided to focus our testing on the sections that handle selling products and supporting them. Those sections might go 4, 5, or even 6 levels deep, which will give us some detailed results.

Suppose we also have an *About Us* section that’s equally deep. If we’re not really interested in testing it (perhaps because that section gets infrequent traffic), we could cut it off at, say, level 2 (at topics like *Company History*, *Staff*, *Our Vision*, *Jobs*, etc.).

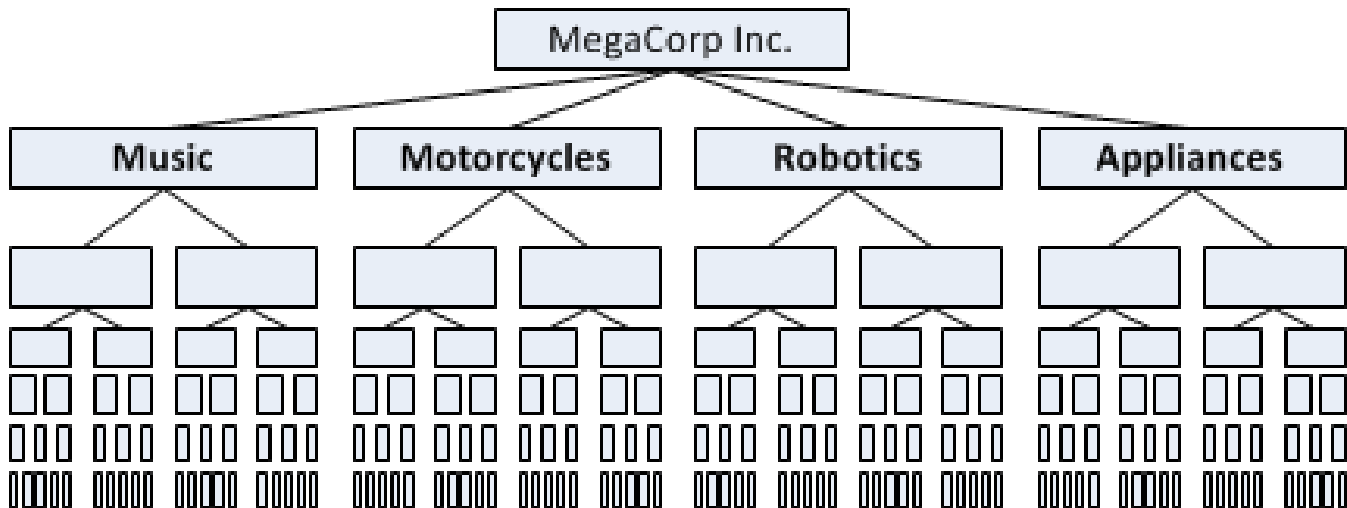


We may also want to do some selective pruning after we’ve [written our tasks](#) (covered in Chapter 7). If there’s a section of our site that none of our tasks come anywhere near, we can probably chop its lower sections without affecting the results. While it won’t save users time during the test (we don’t expect them to visit that section anyway), it will save us time preparing the tree.

## Testing subtrees

If we have a very large tree (perhaps thousands of items), a more drastic way to cut it down is to only test a single section of it.

Suppose again that we're testing a large corporate site like Sony or IBM, with many different business divisions, or perhaps a broad government agency with many different departments:

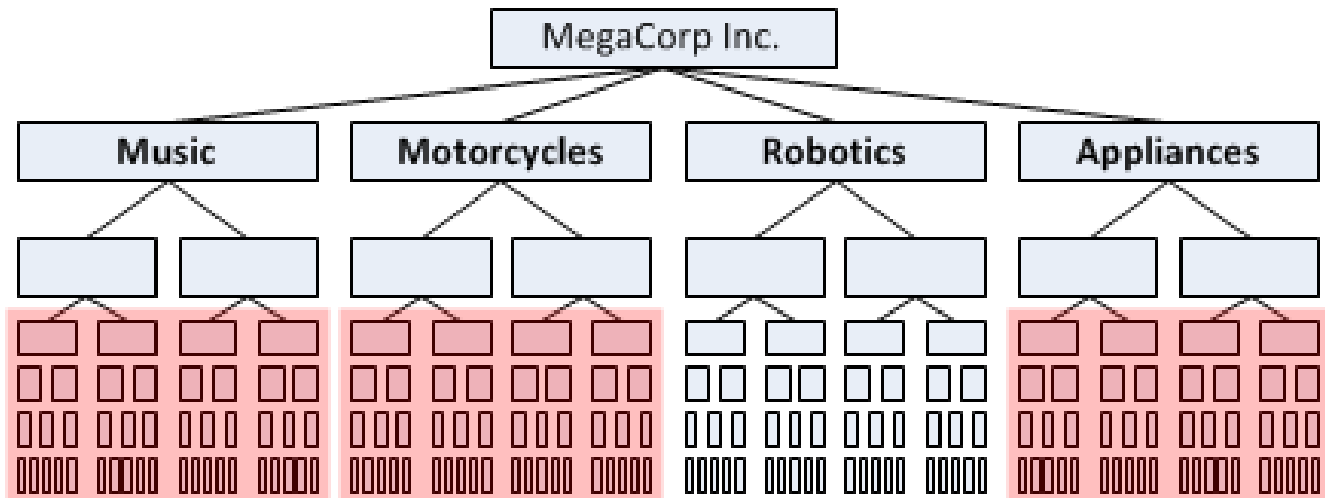


Do we test the entire tree, or just part of it?

Let's take the easy case first. If we want to test several parts of the big tree, we're going to create tasks that use those sections, so we'll clearly need to keep all the sections, at least down to a certain level. If we want to cut down the tree, we'll need to prune levels across the board or selectively, as described above.

If we're really just testing a single section of the big tree, it may still be a good idea to include all the top-level sections anyway, to **see if participants can find their content amid the "clutter" of other divisions/departments**. In this case, we can safely chop off the sections we're not interested in at level 2 or 3, because we don't expect much traffic there (hopefully none at all). Note: don't cut those sections off at level 1, because participants will quickly figure out that those are just stubs and not where the real action is supposed to be.

For example, if we want to test *Robotics*, we could abbreviate the other divisions:



In the end, we should only test a subtree by itself if:

- We don't intend to test content in other parts of the big tree, and
- We're very sure that participants would choose our subtree every time for each of the tasks we give them, or that they would go to our subtree directly in the real site, bypassing the parent site entirely.

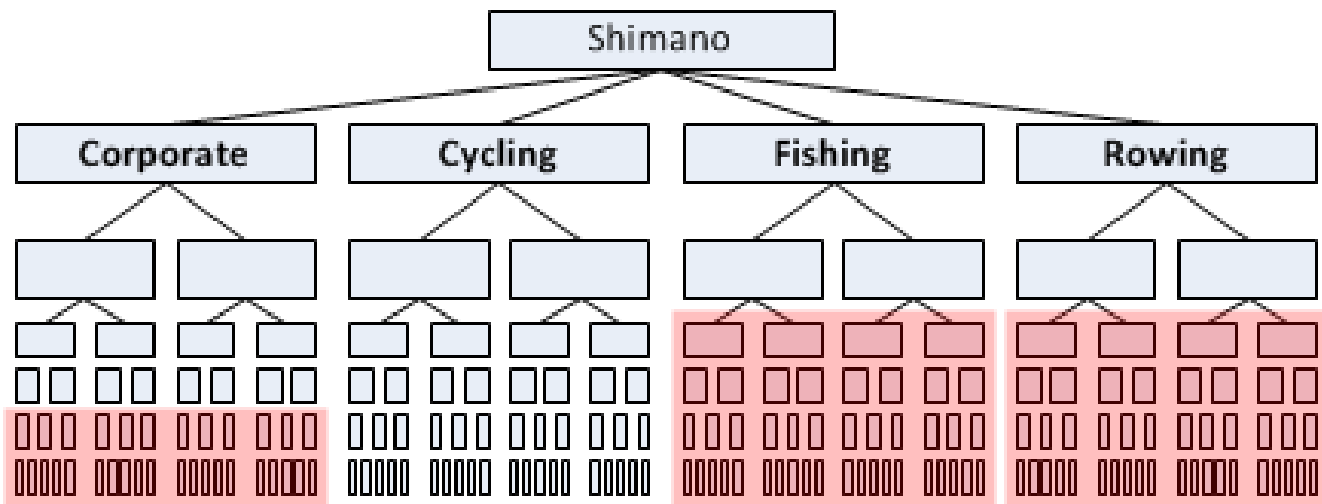
A real-life example makes this clearer. Suppose we're testing the Shimano website. Shimano is best known for cycling products (bike pedals, gears, etc.), but they also make equipment for fishing and rowing.

Here is their top-level navigation:



If we were testing pure cycling tasks (look for bike shoes, check a warranty, read a product FAQ), we could just test the *Cycling* subtree (with *Shimano Cycling* as the root of the tree), because we'd be 99% sure that no one would choose the other sections shown here – they clearly have nothing to do with cycling.

If, however, one of our tasks involved finding the story of how Shimano got started in the bike industry, now it's not so cut and dried. We could imagine some participants going to the *Corporate* section to find the company's history. In that case, we might be best to include these top-level sections, include the full *Cycling* section (obviously), most or all of the *Corporate* section, and the top level or two of the *Fishing* and *Rowing* sections.



If we were writing tasks that involved both cycling and fishing, then clearly we would have include all these top-level sections, but we could safely cut the *Rowing* section short. The bigger problem in this case would be that we're asking each participant to try both cycling and fishing tasks, and a cyclist (for example) may not know anything about fishing, so their fishing results would be suspect. For more on setting different tasks for different user groups, see [Different tasks for different user groups](#) in Chapter 7.

Next: [Which headings to include/exclude?](#)

## Which headings to include/exclude?

- Including global navigation
- What about footer links?
- Excluding non-global links
- Excluding “Overview” headings
- Excluding Search and Site Map
- Excluding Contact Us, Help, etc.
- Dealing with faceted browsing
- Dealing with complex navigation

When we’re preparing a site structure for tree testing, one of the most important things to decide is which headings to include and which to exclude.

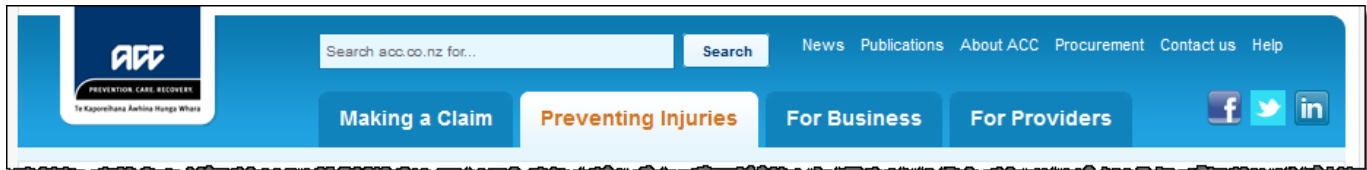
In general, we recommend to:

- **Include global navigation items**, unless they’re very minor
- **Exclude links or shortcuts that are not in the global navigation**
- **Exclude “escape” headings** like *Search*, *Contact Us*, *Help*, etc.

## Including global navigation

In [What is tree testing?](#) in Chapter 2, we saw that the purpose of tree testing is to evaluate the top-down structure of a site – that is, whether users can find what they’re looking for by browsing the site using its top-down navigation.

In most sites, **this top-down tree structure is represented by the global navigation**. Most often this is a row of tabs or headings across the top (or down the left side) of each site page:



Using the example above, our tree would clearly include the main tabs (*Making a Claim*, *Preventing Injuries*, etc.) and the subtopics under each of them.

But what about the other global navigation links – the ones that are in the header, but are not major tabs? In the example above, should we include *News* and *Publications* in our tree?

In general, the answer is yes – **we should include most global navigation items in our tree**, because they are top-down navigation links **available on every page of the site**.

Notice that we said “most”, not “all”. There are often other items that are so minor that they would just add clutter to a tree test. In the example above, we might choose to exclude the social-media buttons, unless we intended to write tasks that involved them.

## What about footer links?

In most sites, it isn’t just the header that provides global navigation. The footer usually does a little (or a lot) of this as well.

Sometimes the footer just repeats the major navigation from the header, or even expands it to show the second level as well (the popular “mega footer” pattern). In this case, the answer is easy – omit the footer links, because we already have them in our tree (from the header).

What about additional links in the footer? For example, here’s the footer from the example site above:



Should we include *Glossary*, *Privacy*, and other links like them?

The answer here is the same as for minor links in the header: **we should include them if we think they'll affect the test**, but we can exclude them if we're sure they would just be clutter.

In the example above:

- We would probably include *Glossary*, because we could imagine some participants going there for a given task.
- We would probably **exclude** *Privacy*, because it's rarely visited for everyday tasks. For the purposes of our study, it's just clutter.

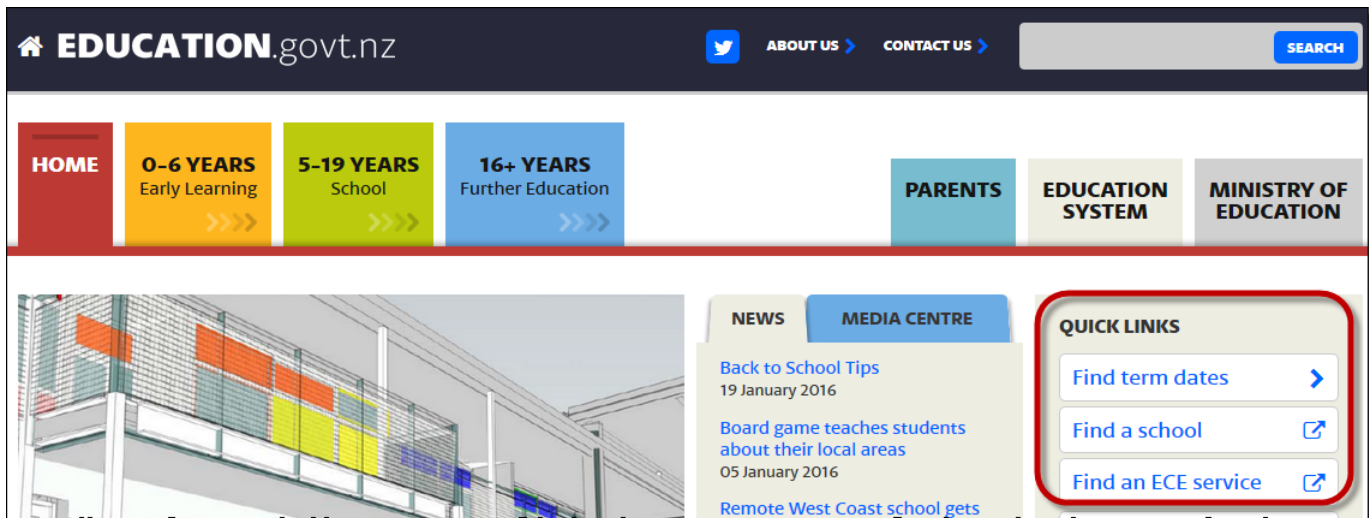
### Excluding non-global links

When we're deciding which headings to include, we must be careful to distinguish between:

- global-navigation links
- "featured" links on a page

As discussed above, **we are looking for links that in the global navigation scheme** (usually a menu system), to provide users with a way to find content by browsing.

Here's an example from a government education site:



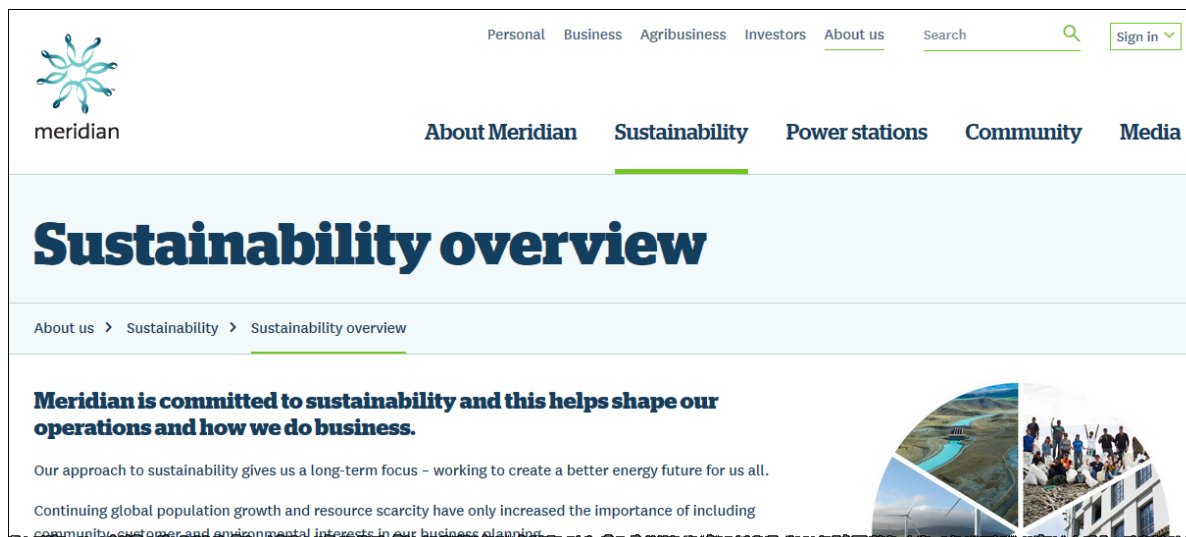
Should we include *Find term dates* and *Find a school* in the top level of our tree? After all, they're going to be used by a lot of home-page visitors to navigate to important content.

The answer in this case is a clear “no”. While these links are prominent on the home page, they do not appear on most pages of the site (presumably), so **they're not part of the global navigation**. Remember that we're not just testing navigation starting from the home page; we're testing navigation from anywhere in the site, especially because of how often visitors arrive at subpages via deep links or search engines. These links should only appear in our tree if they have a place in the site's menu hierarchy.

This also reminds us that tree testing is not enough on its own. While it will test top-down browsing, **it will not test the effectiveness of surfacing important links on a given page**. In the example above, featuring *Find term dates* will definitely affect how users navigate from the home page itself, but that needs to be user-tested with a tool that can show actual page layouts, visual design, and so on.

## Excluding “Overview” headings

In many content sites, each major section has an introduction that describes what the section is about, and which topics will be covered. Often this is labeled “Introduction”, “Overview”, or something similar. Here's an example from a power company:



In general, we **exclude the Overview topic** from our tree when we test it. Why? Because while it often mentions content that is covered in more detail later in the section (and could therefore be considered a “correct” answer for a task that asked about that content), it's not really where the “real” answer is. And because an overview may mention a lot of different topics, we don't really find out if the user was looking for the right thing.

It's also important to watch for the opposite problem – **content hidden under a main heading because it doesn't have a subheading of its own**. For more on this, see [Spotting missing content](#) later in this chapter.

## Excluding Search and Site Map

**We exclude Search because we're testing how users browse.**

If we include *Search*, many participants will encounter a task and say to themselves, “Hmm, I'd probably just search for that”, click *Search* in our tree, and the task would end. While that's a useful result in a normal usability test (observing that the user chose to search instead of browsing), it doesn't tell us much about how effective browsing is.

By excluding *Search*, we're effectively saying to them, “Yes, you might search on the real site, but if you had to browse, where would you go?”. This way, each participant gives us useful results on every task.

As for *Site Map* links, we exclude them because they're redundant; the textual tree we're presenting is a site map itself.

## Excluding Contact Us, Help, etc.

The reasoning we use for excluding *Search* also applies (usually) to *Contact Us*, *Help*, and similar headings. These are often used as “escape” choices, where the participant scans the list of headings, doesn't find anything promising, and says to themselves, “Ah, I'd probably just call them and ask.”

While that's a useful finding (none of the other headings seemed to match), we've found that providing a *Contact Us* topic or *Help* topic (especially

at the top level) **makes it a bit too easy to bail out on a task**. We might learn a little, but overall it's better to make them either pick a "real" heading or skip the task entirely.

The only time we want to include *Contact Us* or *Help* is if **we're actually testing subtrees under those headings**. For example, we may want to know if users can actually find items in our Help system, so we include a subtree of topics under Help. The trade-off here is that we will almost certainly get less feedback from other tasks when some participants use the *Help* topic to bail out of browsing the main tree.

## Dealing with faceted browsing

Tree testing is well suited to static content sites, where the grouping of topics and their subtopics doesn't change on the fly.

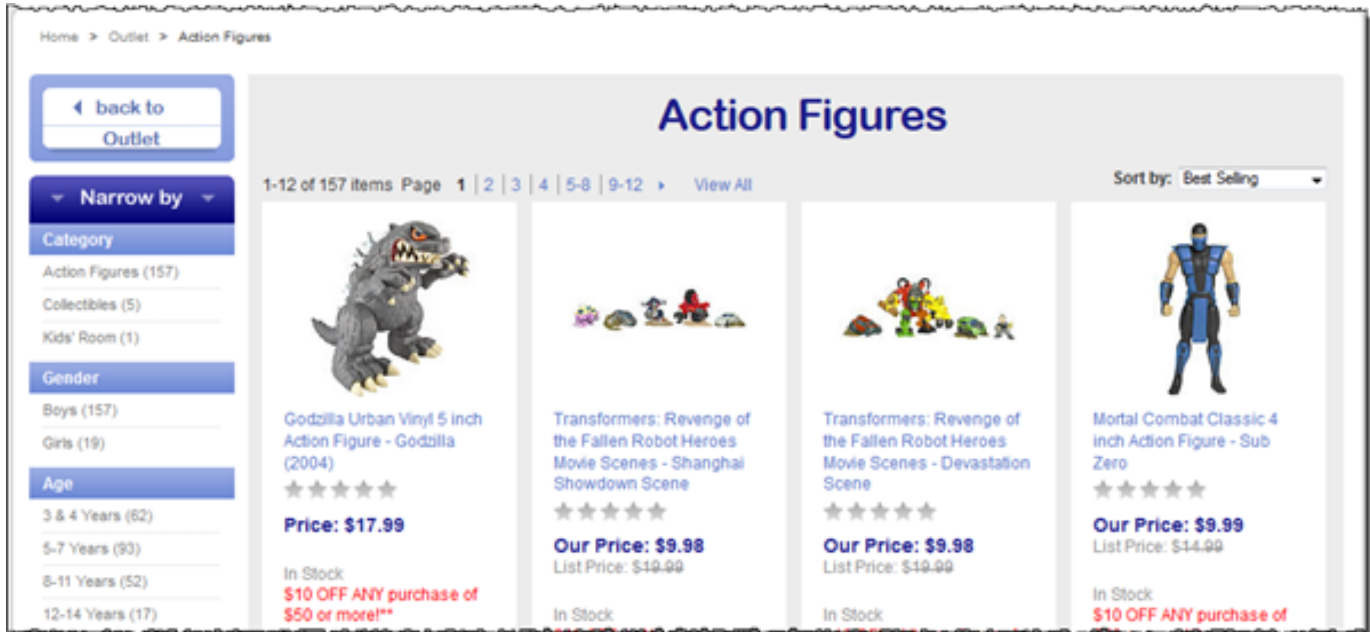
Occasionally, though, we may need to model a tree that includes **faceted browsing/search**, where the subgroups change dynamically.

Here's an example from [eToys.com](http://eToys.com). In their *Outlet* section, we can browse topics such as *Action Figures*...

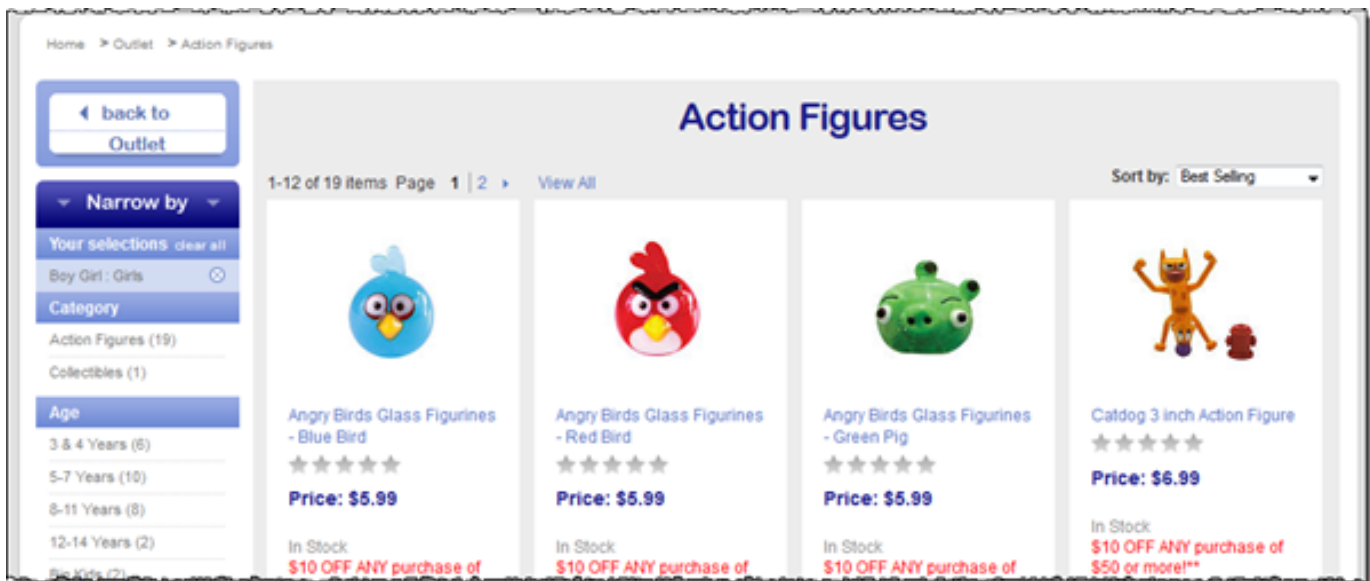


...then browse by lots of subtopics:





The problem here is that we've wandered into a **faceted search**, where choosing a subtopic like *Gender > Girls* filters the search results, but still offers all the other subtopics to continue filtering by:



If we wanted to model this in a tree test, we would need to offer every subtopic under every other subtopic, and then every remaining subtopic under that subtopic, and so on all the way down (until we ran out of topics). Trying to create a static tree of subtopics from a dynamic set of facets is a lot of work, and probably not worth it.

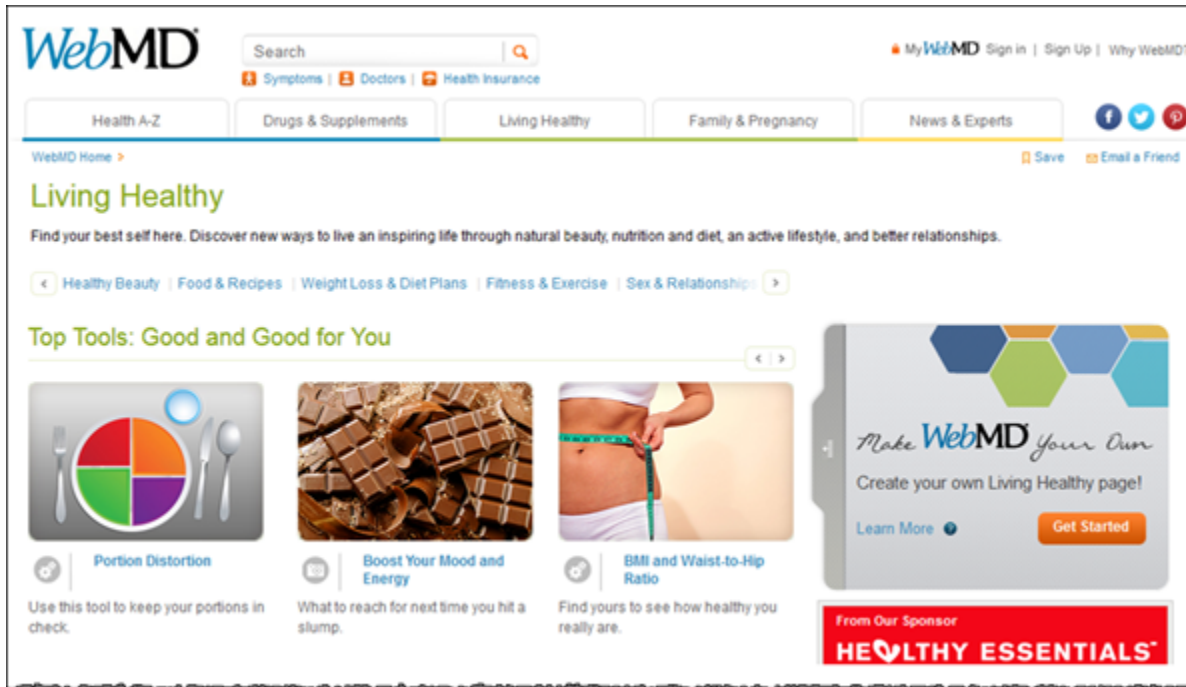
We have two alternatives here:

- If this was just part of a larger site, we might just model the faceted section one or two levels deep, to see if our facets betray any glaring problems.
- If we really want to find out how well the faceted browse/search itself works, we'll need to develop a working faceted search UI and do usability testing on it. Tree testing is just not the right method for this.

## Dealing with complex navigation

Some sites are hard to model in a tree test because they have **inconsistent global navigation** – that is, their top-down browsing structure is hard to determine based on the headings they present.

WebMD, the popular healthcare site, is a good example of this. At first glance, it presents what looks like a fairly standard global navigation scheme. Here is the *Living Healthy* section, with a strip of secondary navigation (*Health Beauty*, *Food & Recipes*, and so on):



If, however, we use the mega menu that appears when we hover over the *Living Healthy* tab, we get a **different set of topics and subtopics**:



Clicking through the WebMD site further, we actually find **3 or 4 different ways that they organize topics**, some of them overlapping.

Because tree testing insists on a consistently “top-down” browsing structure, it would hard to run a fair tree test on sites like this. A conventional usability test of the actual site would be a much better choice.

Next: Spotting missing content

## Spotting missing content

- Topics with only 1 subtopic

Perhaps **the most common mistake in tree preparation is omitting “hidden” content**. This is caused by following the page titles of the site, instead of looking for logical chunks of content.

By “hidden”, we mean **content that is on a site page, but doesn’t have a page of its own**, so to speak. This most often happens on landing pages, where we talk about topic A on the page, then link to topics B and C a level below. Because A doesn’t have a page title of its own, it “disappears” from our tree, because we’ve been focusing on page titles.

For example, suppose we have a *Contact Us* section (and suppose we’ve decided to actually include it because its organization needs testing). On the website, the *Contact Us* page has the company’s main contact info (in New York City), and then links to subpages for *Asia* and *Europe*:



When we put this into a text tree, we get the following:

- Products
- Services
- Contact Us
  - Asia
  - Europe

Now, when the participant gets a task like “Call up the company at their headquarters”, they click *Contact Us* and see two choices – *Asia* or *Europe*.

What happened is that the “USA” content got lost under the *Contact Us* topic. Many tree-testing tools only let us choose an answer at the end of the tree (a so-called “leaf node”), so *Contact Us* would not be selectable. Even in tools that let us select “midway” topics, participants tend to focus on the “next” choices that appear – in this case, *Asia* and *Europe*.

Once we’ve spotted this type of implicit content, it’s usually easy to fix. In our example, we would add an explicit *USA* topic:

- Products
- Services
- Contact Us
  - **USA**
  - Asia
  - Europe

Now the tree represents **the logical chunks of content on the site, not just a list of page titles**.

This is more of a problem with testing existing trees than with new trees that we create ourselves, because the existing trees are often extracted from a CMS, and they’re focused on pages and their titles. **When we extract an existing tree from a CMS, we should take some time to compare the “page” tree with the actual content, especially the “hidden” content of landing pages.**

## Topics with only 1 subtopic

When we're preparing our tree for testing, **we must watch for topics that have a single subtopic**. In most cases, this is a clue that some content is hiding there.

Suppose we have the following headings, taken from a power-company site:

- Understanding your bill
- Reading your meter
  - Sending us your meter reading
- Moving house
- etc.

Clearly, there are two meter-related topics here – *Reading your meter* and *Sending us your meter reading*. The *Reading your meter* page contains text on how to read a meter, and adds a link to a subpage, *Sending us your meter reading*.

However, during tree testing, when the participant clicks *Reading your meter*, there is only a single subtopic to select, *Sending us your meter reading*. We need to add the “reading” content as a selectable item, like this:

- Understanding your bill
- Reading your meter
  - **How to read your meter**
  - Sending us your meter reading
- Moving house
- etc.

Sometimes, though, there really **is** only one piece of content in a section. For example, if a company only sells one product, *their Products* section may only have the one subtopic, so it makes sense that it's the only selectable one in the *Products* subtree.

---

Next: [Dealing with shortcuts and duplicated content](#)

## Dealing with shortcuts and duplicated content

- Including duplicated subtrees

How should we represent **content that appears in more than one place** in our tree?

For example, suppose we have a *Frequently Asked Questions* (FAQ) page for a product, and that *FAQ* page is available from both the *Products* section and the *Support* section, like this:

- Products
  - Wizzo-matic
    - Features
    - **FAQ**
- Support
  - Downloads
  - FAQs
    - **Wizzo-matic FAQ**

It could be that this *FAQ* page is actually duplicated – each section has a separate page with the same content. Or (and this seems to be more common), there is only one actual *FAQ* page (say, the one in the *Support* section), and the one in the *Products* section is just a link to it.

In the tree we're going to test, do we include the *FAQ* topic in both sections, or do we remove the one in *Products* because it's not a "real" content page?

In general, **we recommend including both entries in the tree**, because even if the *Product-FAQ* link just links to the *Support* section, it's still a choice that we give the user when they go to the *Wizzo-matic* product page. If we didn't include the *Product-FAQ* topic in our tree, we would be withholding information from the test participant that a visitor to the real site would have.

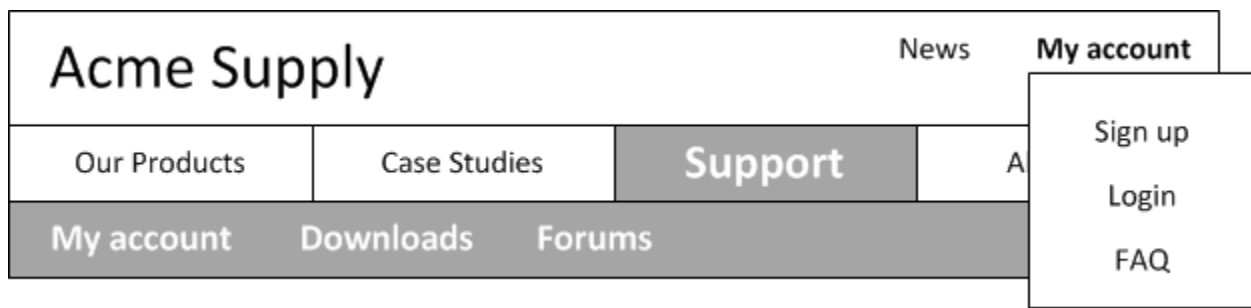
Later, when we analyze the results, we will definitely want to see if participants took the *Products-FAQ* path or the *Support-FAQ* path. If we see that 90% of them go first to the *Products* section for the *FAQ*, that suggests that the *FAQ* should move to live there, and the *Support-FAQ* topic should just be a link to it.

If we choose not to include both entries in our tree, we can still check the results to see which path the participants took. If a sizable fraction of them went to the section without the *FAQ* topic, that suggests that we should at least include a cross-link there when we design the real site.

### Including duplicated subtrees

What if the duplicated content is not just a single topic (like the *FAQ* page above), but an entire subtree of topics? Should we duplicate the entire subtree for testing?

For example, suppose we have a *My Account* section in our site, which lets users create an account, log into that account, and see a my-account *FAQ*. It "lives" in the *Support* section, but we've also added a shortcut in the main navigation bar (a utility link and pop-up menu in the upper right of the header):



Here's how we might represent this in a text tree:

- Products
- Support
  - **My account**
    - Sign up
    - Login
    - FAQ
  - Downloads
  - Forums

- About us
- **My account**
  - Sign up
  - Login
  - FAQ

Notice that *My Account* appears twice in the tree, because it appears in two places in the global navigation.

Again, in general, **we recommend including both subtrees in the test**, because that's what the user would see in the real site.

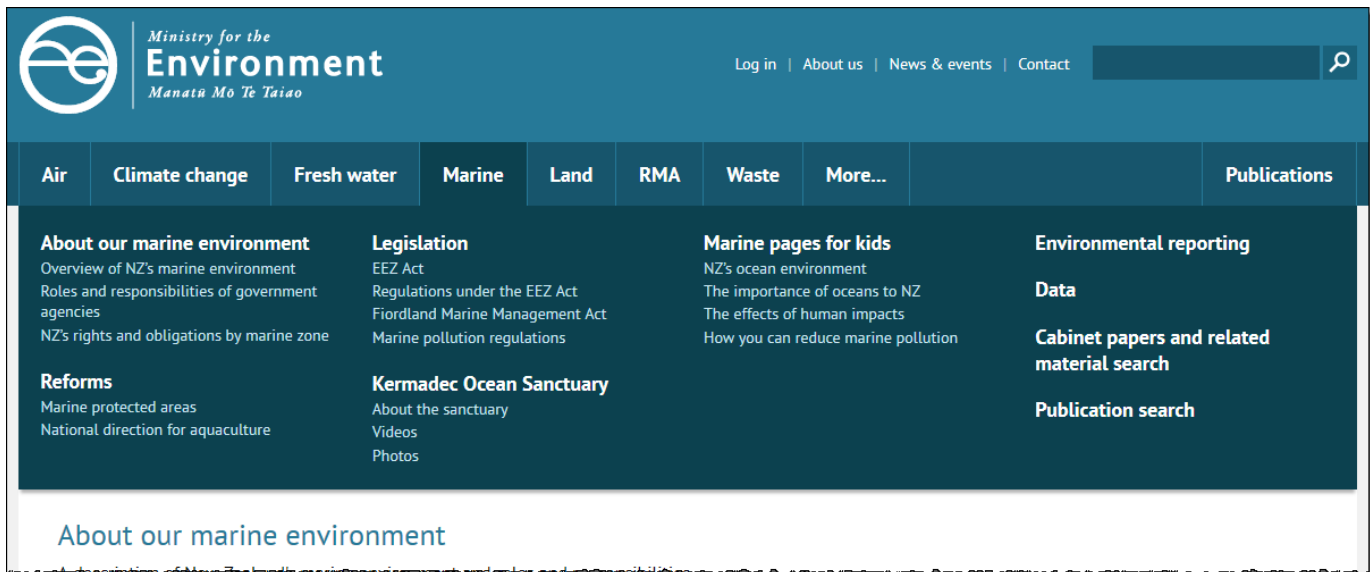
The only time we may not want to duplicate subtrees is if the subtree is very large, and it becomes tedious to keep both instances in sync as we develop the tree. In that case, we may want to prune the duplicate that we think is going to get less traffic during the test.

---

Next: [Breaking up double-level topics](#)

## Breaking up double-level topics

Many websites show users several levels of navigation at once, as a way of previewing what's in each section. Mega menus are a popular way of doing this:



But most tree-testing tools insist on showing only one level of navigation at a time. How should we handle this in our tree test?

There are three common approaches that we've seen:

1) **Show one level at a time** – This ignores the previewing of subtopics and makes the user click through each level manually:

- Ministry for the Environment
  - Air
  - Climate change
  - Fresh water
  - Marine
    - About our marine environment
      - Overview of NZ's marine environment
      - etc.
    - Reforms
    - etc.
  - etc.

2) **Combine levels using a prefix** – The subtopics are brought up to the parent's level, and the parent topic becomes their prefix:

- Ministry for the Environment
  - etc.
  - **Marine - About our marine environment**
    - Overview of NZ's marine environment
    - etc.
  - **Marine - Reforms**
  - Marine - etc.
  - Land - etc.

3) **Indent using spacer characters** – Subtopics are moved up to the parent level, but they are moved to the right to look like they are under the parent topic. In the example below, underscores show the spaces.

- Ministry for the Environment
  - Air

- Climate change
- Fresh water
- Marine
  - \_\_About our marine environment
    - Overview of NZ's marine environment
    - etc.
  - \_\_Reforms
  - \_\_etc.

While all three methods work, **we generally go with method 1**. While it is less "realistic" than the others (in terms of visual presentation), it does test the clarity of each topic separately. **Ideally, parent topics should be clear and distinguishable on their own**, even without previewing their subtopics.

---

Next: [Using link names instead of page titles](#)



## Using link names instead of page titles

---

Once we decide a heading is going to be included in our tree, what do we call it?

In some cases, the global-navigation link to a page **uses different text** than the actual title of the page itself.

Here's an example. In the global navigation, we have a link called *Jobs*, but the destination page is titled *Job vacancies*:



Which text should you use? **We recommend using the link text, not the page title**, because the link text is what users look at when deciding which path to follow.

Note that if our tree started as a dump from a Content Management System (CMS), it probably used the page titles. We should make sure to review the global navigation to see if there are any discrepancies between those titles and the link names that point to them.

---

Next: [What to call "Home"](#)

## What to call “Home”

---

When we speak of “level 1” navigation, we mean the main sections (usually the top-level tabs) of a website. On a typical company site, this might be headings like *Products*, *Support*, *About Us*, and so on.

What about “level 0”? What do we call the root of the tree?

By default, most tree-testing tools name this root node either *Home* or *Top* or the name of the organization. In most cases, this should be clear enough to participants.

**Watch, though, for confusions between the root label and the site headings.**

For example, suppose we’re working on a phone-company site that sells services to both residential and business customers. When we put our site structure into the testing tool, it comes out as:

- Home
  - For home
  - For business
  - Support
  - etc.

Participants are likely to confuse the root node *Home* with the *For home* section, especially when they’re backing up the tree during a task. In this case, renaming the root node to *Top* solves the problem.

Another solution is to be more explicit with the root name. In the example above, we could also rename the root to *Acme home page*.

---

Next: [Transferring the tree to a testing app](#)

## Transferring the tree to a testing app

- [Sanitizing the tree text](#)
  - [Importing into an online tool](#)
  - [Checking the imported tree](#)
  - [Randomizing subtopics](#)
- 

Once we've reviewed our tree, tweaked it, and decided that it's ready for testing, it's time to copy it from our spreadsheet into a tree-testing tool.

### Sanitizing the tree text

Just before we import the tree, it's prudent to check for certain characters that may foul things up, namely:

- **Leading spaces** – These often creep in when pasting text from other software. Spreadsheets hide these, but they can cause unintended indenting (or even be pushed down a level) when imported.
- **Line breaks** – If we dumped a site structure from a CMS, it may have included hard link breaks or carriage returns. Or, we may have added these in the spreadsheet ourselves. Either way, these are sometimes interpreted by the importing tool as 2 separate topics, not a single topic.

Before importing, we remove these offenders using the search/replace function of our spreadsheet app, searching for the leading blank or the line break (we may need to look up the character code for the latter) and replacing it with nothing.

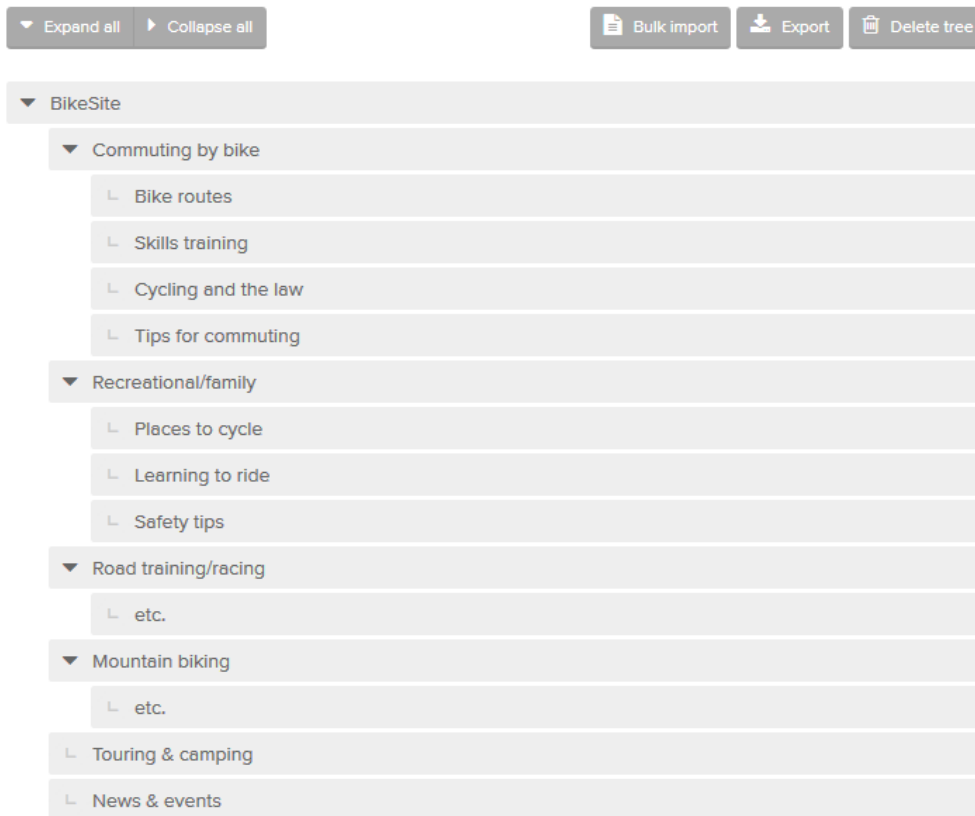
Having trouble finding invisible special characters? Try loading the initial text into a text editor that shows them as visible symbols.

### Importing into an online tool

Most online tree-testing tools make it easy to import a tree, either by:

- copying the cells from a spreadsheet into a text box in the tool, or
- importing a file (which may be a text or Word file with indenting, or a spreadsheet file with columns of headings).

In either case, the tool then parses the text into a tree of headings and subheadings.



## Checking the imported tree

Regardless of how we import our tree into the tool, we must be sure to check that nothing got lost in translation. In particular, we watch for:

- **Delimiters** – In addition to the line break mentioned above, there are additional characters used to enclose text, such as quotes, angle brackets, punctuation, etc. Check that these haven't caused splits in the middle of topic text.
- **Special characters** – These are usually non-English characters that incorporate various accent marks (e.g. the French **é**). Most should be fine, but check that they have survived the import without being changed.

## Randomizing subtopics

Some tree-testing tools offer the option to randomize the subtopics under a given parent topic.

For example, suppose that part of our tree looked like this:

- FAQs
  - Baseball
  - Basketball
  - Football
  - Rugby
  - Golf
  - Hockey
  - Rowing

If we chose the option to randomize subtopics, then during the tree test, each participant would see these subtopics in a random order:

Participant 1 sees... Participant 2 sees... Participant 3 sees...

<ul style="list-style-type: none"><li>• FAQs<ul style="list-style-type: none"><li>• Football</li><li>• Rowing</li><li>• Baseball</li><li>• Golf</li><li>• Basketball</li><li>• Rugby</li><li>• Hockey</li></ul></li></ul>	<ul style="list-style-type: none"><li>• FAQs<ul style="list-style-type: none"><li>• Rugby</li><li>• Rowing</li><li>• Basketball</li><li>• Baseball</li><li>• Football</li><li>• Hockey</li><li>• Golf</li></ul></li></ul>	<ul style="list-style-type: none"><li>• FAQs<ul style="list-style-type: none"><li>• Hockey</li><li>• Baseball</li><li>• Rugby</li><li>• Golf</li><li>• Basketball</li><li>• Football</li><li>• Rowing</li></ul></li></ul>
---	---	---

The purpose of this option is to compensate for participants who tend to choose items near the top of the list without looking at items near the bottom. This tends to happen when the list is long (10 subtopics or more).

However, we should only randomize subtopics like this if **both** of the following are true:

- Our tree has **many sections with long lists of subtopics**, *and*
- **We haven't ordered the subtopics in a particular way** (e.g. logically, chronologically, alphabetically, etc.).

These are unlikely criteria for most site structures, so **we don't recommend using the "randomize subtopics" option in most cases.**

Don't confuse "randomizing **subtopics**" (described above, and rarely used) with the "randomize **tasks**" option (described in [How many tasks?](#) in Chapter 7, and often used).

---

Next: [Chapter 6 - key points](#)

## Chapter 6 - key points

Normally we test the entire tree, but **if it's too big to feasibly test**, we can either prune levels, prune unnecessary branches, or test specific subtrees instead.

We need to decide **which headings to exclude** – non-global links, overview headings, search, “contact us”, and so on.

Be careful not to miss content that doesn't have its own page (usually because it's “hidden” on a landing page).

We must decide how we're going to handle **duplicated content and shortcuts**, as well as double-level headings.

If our page titles are different from their corresponding link names, **we should use the link names** in our tree.

**We should name our tree's root** (e.g. Home) so that it doesn't get confused with our top-level headings.

When importing our structure into a tree-testing tool, **watch for problematic characters** that may alter the topic levels or the text itself.

In most cases, **we should not use the tool's “randomize subtopics” feature** (if it offers one).

---

**Next:** [Chapter 7 - Writing tasks](#)

## 7 - Writing tasks

*"But I still haven't found what I'm looking for" - U2*

In conventional usability testing, we test a UI by **having participants perform tasks** - things that we want them to find or to do, just as they would when using that UI for real.

Tree testing is no different. We don't want the participant to just wander through the tree, giving us their opinions on how easy it would be to find things. **We want to simulate what it's really like to look for something** – something specific – using the top-down hierarchy of the site.

So, we ask our participants to start at the top of the tree, and we give them a definite item to find. In fact, we give them **a series of tasks** – enough to test several parts of the tree in several different contexts, but not so many that they get tired or grumpy, and not so many that they learn the tree more than a real site visitor would.

And our job, for each task, is to make sure that it's concise and unambiguous. We need each participant to understand the task quickly, and decide it means the same thing that we meant when we wrote it.

In this chapter, we'll cover how to decide which tasks to include in our tree test, and how to make sure each one is clear to our participants.

---

### Which tasks to include?

Common and critical tasks, problem areas, and borrowing

### How many tasks?

As many as needed to cover major areas, but <10 per participant

### Mapping tasks to the tree

Collecting task ideas, refining them, and checking coverage

### Different tasks for different user groups

Reasonable pretending, mixing and splitting groups

### Collaborating on tasks

Divide and conquer with multi-user editing

### Writing a good task

8 tips on creating effective and unambiguous tasks

### Identifying correct answers

Multiple answers, intermediate nodes, and judging correctness

### Entering tasks and their answers

Copying from a spreadsheet to an online tool

### Randomizing the order of tasks

Do this to reduce the learning effect on your results

### Letting participants skip tasks

Almost always a good idea

### Asking questions after a task

[summary text here](#)

### Key points

---

**Next:** [Chapter 8 - Setting up a test](#)

## Which tasks to include?

- Common tasks
  - Critical tasks
  - Tasks that exercise “problem” areas
  - Borrowing from usability tests
  - Borrowing from card sorts
- 

With any decent-size tree, the problem is not coming up with enough tasks to ask; it’s **cutting down a large number of initial task ideas to a manageable set** that answers our main IA questions.

Just as in conventional usability testing, we **concentrate on the most common and critical tasks**, plus those that test **parts of the site that we’re most unsure about**. But we make sure we’re not hitting those too hard, lest we miss a problem elsewhere or “train” our participants too well.

### Common tasks

One of the basic tenets of good design is to make common functions easy. In IA terms, this means **making frequently looked-for things easy to find**.

So we should start our task list with the common things our users would be looking for or doing on our site.

If we have existing user research at hand (surveys, usability-test results, analytics, search logs), then coming up with a list of ideas for common tasks should be easy.

### Critical tasks

We also want to include tasks that cover **things that users don’t need often, but need fast if the situation arises**.

For example, on a mobile phone, we may not ever dial our country’s emergency number, but if *we do* need to, it has to be *really* easy to find.

On a website, an equivalent case might be resetting a password. Users shouldn’t have to do it very often, but if they do, it needs to be easy to find.

### Tasks that exercise “problem” areas

We definitely want to include tasks that test areas of the site that:

- Have been **substantially reorganized or renamed**
- **Have been argued or worried over** by the project team
- **Have several ways they could be structured**

This is one reason why we recommend working in a spreadsheet format. As we ponder, discuss, and argue about various parts of the tree (or compare various tree ideas), we can remind ourselves to test those parts by adding a note in the *Task* column:



	A	B	C	D	
	Level 0	Level 1	Level 2	Tasks	Notes
	BikeSite				
		Commuting by bike			
			Bike routes	add a "cycle path" question	
			Skills training		
			Cycling and the law		
			Tips for commuting		
		Recreational/family			
			Places to cycle		
			Learning to ride	Add a task for this or "skills training"?	
			Safety tips		
		Road training/racing		No tasks for roadies in this round	
			etc.		
		Mountain biking			
			etc.		
		Touring & camping			
		News & events			

### Borrowing from usability tests

If this mantra of “common, critical, and suspect” tasks sounds familiar, that’s because it is. **These are often the same tasks that we focus on in traditional usability tests.**

The good news here is that, if we have run usability tests on our site, we can **recycle some of those user-test scenarios into tasks for tree testing**, especially any scenario that involved finding something on the site.

### Borrowing from card sorts

If we’ve done a card sort as part of our IA work (as described in [The research phase](#) in Chapter 3), we’re already halfway to creating a list of tasks for tree testing.

When we prepared the card sort, we picked out representative content from our site and created each item as a card. Now that we’ve created a tree where all that content is going to live, **a subset of those cards can be reworded into tasks for the tree test.**

tips on buying a car
accessible parking spaces
pay a speeding fine
traffic webcams
road tax for scooters
pay toll road fee
rebuilding permits for Christchurch
claims for earthquake damage
adult literacy courses
veterinary schools in NZ
support for gifted children
student scholarships

Note that in card sorting, we typically create 30-50 cards, whereas in tree testing, we’re normally dealing with far fewer tasks. In this case, it’s mostly a matter of choosing which of our cards answers our tree-test questions best, and gives us the coverage we want for the tree. And we’ll

also probably need to reword those cards into proper tasks, as covered in [Writing a good task](#) later in this chapter.

---

Next: [How many tasks?](#)

## How many tasks?

- How many tasks overall?
- How many tasks per participant?

First-time tree testers often ask “How many tasks should I include?”.

More precisely, we need to consider:

- How many tasks should the test include **overall**?
- How many tasks should **each participant** see?

## How many tasks overall?

There is no single definitive answer to how many tasks we should include in a tree test. The number will vary depending on several factors, including:

- **The size of our tree**  
Larger trees typically need more tasks to ensure adequate coverage.
- **The complexity and variability of the tree**  
If we have many sections that reuse the same general structure (e.g. several product subtrees that have very similar layouts), then we may not need to test each one.
- **Our confidence in the tree**  
If we're just making minor changes to a tree that has tested well before, we may not need to present many tasks. On the other hand, if it's a completely redesigned, unconventional, or contentious tree design, we'll definitely want to “put it through the ringer” by presenting a larger number of tasks.

As a starting point, though, the following numbers are typical of the tree tests that we run:

Size	# of items	# of tasks (overall)
Small tree	1-250	8-12
Medium tree	250-500	10-20
Large tree	500+	15-25

If we have **FEWER** tasks than this, we need to check coverage of the tree (described in [Mapping tasks to the tree](#) later in this chapter). There may be important parts of the tree that we're missing.

If we have **MORE** tasks than this:

- We should **check that each task is answering a specific question** we have about the site structure. If it isn't pulling its weight, or if other tasks already address that question, consider removing it.
- Remember that each task takes time to write (up front) and time to analyze (later), and that we'll need a bigger participant pool to get all those questions answered (see below).

## How many tasks per participant?

**Give each participant 8-10 tasks. More is risky.**

No matter how many tasks we create overall, **there is a (relatively low) limit to how many tasks we should ask each participant to do**. This limit stems from two factors:

- The time and effort required by each participant
- The learning effect of browsing the same tree repeatedly

## Participant effort

If we ask a participant to do **8–10 tasks in a tree test, that typically works very well** – they try a small number of tasks (each one a bit different), they see the tree a few times (but not too often), and they finish in 5 minutes or so, so they're not tired, bored, or grumpy at the end. They got a short, challenging exercise that was different (and more fun) than a traditional survey, and they leave happy (and probably willing to say yes to the next study we ask them to do).

That's fine if we're testing a small tree, where it's easy to whittle our task list down to 10 or so. But suppose we're testing a big tree and we've decided that we need 25 tasks to get adequate coverage of the parts that need validating.

**If we make each participant do 25 tasks, neither we nor they will be happy**, for several reasons:

- When we invite them to the tree test, we'll have to tell them it will take 15-20 minutes, not 5. That may drastically cut down the number of respondents we get.
- After about 10 tasks, they may start to get tired of the exercise, and not try as hard, perhaps skipping more tasks, perhaps picking answers that will just get the test over with sooner. In any case, task fatigue will likely change their behavior, which puts our results in question.
- Fatigue and boredom also means they may be less likely to take part in future studies. This is a particularly important consideration if we have a small pool of participants to begin with.
- The learning effect becomes a real problem (see below).

## The learning effect

The other problem with giving each participant a lot of tasks is the "learning effect"; as they browse the tree for each successive task, **they start learning the structure**.

This is not a bad thing in itself. After all, users visiting a website for more than a few minutes will likely learn the overall navigation to some extent.

However, in tree testing, we're asking them to find things over and over, without anything else to look at except the structure, so **they come to know it better than they would when they visit the real site**. (Most users don't arrive at a site with 10 successive things to look for.)

While some learning is inevitable, we can do two things to **minimize its effect on our results**:

- **Randomize the task order**  
By putting the tasks in a random order per participant, we make sure that earlier tasks don't "help" later tasks across all participants. See [Randomizing the order of tasks](#) later in this chapter.
- **Limit the number of tasks per participant**  
If we present each participant with 25 tasks, they're going to know the tree *very* well by the end, and this will unrealistically improve their performance on the later tasks. Limiting each participant to fewer tasks (say, 8-10) reduces the learning effect to something closer to what they would learn using the real website.

## Setting the number of tasks for each participant

To let us test a large number of tasks overall without overburdening participants, most tree-testing tools let us specify **how many of the tasks are shown to each person**.

For example, suppose we have a medium-sized tree with 20 tasks that cover it to our satisfaction. That's too many tasks to give each participant – they'll get bored or tired, and their tree "learning" will pollute our results. They will be happier (and we'll get sounder results) if we give them 8-10 tasks each.

This does mean that **we'll need to recruit more participants to get the same number of responses per task**. In the example above, if we had 20 tasks overall and showed 10 to each participant, we would need about 100 participants to get 50 responses per task. For more on this, see [How many participants?](#) in Chapter 9.

---

Next: [Mapping tasks to the tree](#)

## Mapping tasks to the tree

- Collecting task ideas
- Refining our task list
- Checking coverage

### Collecting task ideas

As we come up with ideas for tasks, it's important to track them against our tree. The simplest way is to jot them down in our spreadsheet, in the *Task* column:

	A	B	C	D	
	Level 0	Level 1	Level 2	Tasks	No
1	BikeSite				
2		Commuting by bike			
3			Bike routes	add a "cycle path" question	
4			Skills training		
5			Cycling and the law	Is it legal to bike in a bus lane?	
6			Tips for commuting		
7		Recreational/family			
8			Places to cycle		
9			Learning to ride	Add a task for this or "skills training"?	
10			Safety tips		
11		Road training/racing		No tasks for roadies in this round	
12			etc.		
13		Mountain biking			
14			etc.		
15		Touring & camping		Find bike-friendly campsites near Dunedin.	
16		News & events			

In this first pass, **the goal is to collect our ideas quickly.**

- **Don't sweat the wording yet.** At this stage, we're just compiling a list of task ideas, some of which we won't end up using. Trying to fine-tune wording now may be wasted effort, and it will slow us down early in the process.
- **Don't worry about coverage yet.** If we come up with 5 task ideas for a single small area of our site, that's OK. It means we'll have 5 candidates to choose from later when we're trimming the task list. And if we don't have any ideas for a given section, that's OK too – we'll get a chance to fill that in when we check task coverage later.
- **Get everyone involved.** If we're working with a team, we can distribute the work. That makes everything go faster, and gets everyone engaged with the testing. This is particularly easy if we use a shared online spreadsheet.

### Refining our task list

After we've taken a first pass through the tree, either by ourselves or with others contributing too, we'll likely come up with too many task ideas. This is a good thing, because we can then refine our task list down to a smaller number of focused items.

To pick the winners for our task list, we look for:

- **Tasks that test our stated goals.** When we first planned this test, we wrote down the top things we wanted to find out. How well does each task idea answer those questions?
- **Tasks that are the most common, critical, and suspect.** See [Which tasks to include?](#) earlier in this chapter.

- **Tasks that provide good coverage** for the parts of the tree that we want to test. See [Checking coverage](#) below.

It's important for us to mark our task ideas to show which we're keeping and which we're not. For example, we could highlight the chosen ones in green, or we could leave them unmarked and highlight the not-chosen ones in red. It doesn't matter which method we use, as long as we're consistent (and we include a legend so that others can understand our method later.)

Using the spreadsheet method, we find it's better to mark tasks as deleted (e.g. using strike-through or a red highlight) rather than delete them outright. That way, if we change our minds later, we can reinstate a task that we initially "crossed out". In the example below, we've highlighted the chosen tasks in green and struck out the unchosen tasks:

	A	B	C	D	N
	Level 0	Level 1	Level 2	Tasks	
	BikeSite				
		Commuting by bike			
			Bike routes	<del>Where are the cycle paths in Wellington?</del>	
			Skills training		
			Cycling and the law	Is it legal to bike in a bus lane?	
			Tips for commuting		
		Recreational/family			
			Places to cycle		
			Learning to ride	Add a task for this or "skills training"?	
			Safety tips		
		Road training/racing		<del>No tasks for roadies in this round</del>	
			etc.		
		Mountain biking			
			etc.		
		Touring & camping		Find bike-friendly campsites near Dunedin.	
		News & events			

## Checking coverage

When deciding which tasks to include, we also want to keep an eye on **coverage - which parts of the tree we're testing, and which we're ignoring**.

It's not usually feasible to test every subtree of every section of the site, and usually it's not necessary either. For example, if participants can find the specifications for product A in our site (by looking in the *Products* section, say), it's safe to assume that they can also find product B's specs in the same section.

Once we've created tasks that cover the most common, critical, and contentious areas of the site, it's a good idea to pull back and see **which parts of the tree are covered too much or too little**.

- **If we are using the spreadsheet method** to refine our tree and write the tasks, checking task coverage is simply a matter of **scrolling down the tree and looking for entries in the Task column**. We can see which sections of the tree are "covered" by tasks, and which aren't.
- **If our tree-testing software** checks coverage for us, we can activate that feature and see where the gaps are.

If we find gaps in coverage (and we likely will), we may want to change up our tasks a bit:

- **If a given section is particularly important to test**, we will want at least one task for it, and maybe even two. If we add too many tasks to a given section, however, participants may "learn" that part of the tree better than they would during a real visit to the site, and skew our results.
- **If a section is not important to test** (e.g. it worked well in the past, and we're not changing it now), it may be OK to have no tasks for it. Note, however, that if we change the tree around it, this might affect how users then navigate this area of the site. If we have any doubts about this, we should consider adding a task to make sure.

Keep in mind that **some of our tasks may have several correct answers** in different parts of the site (or at least several places where we suspect participants will look). If we're using a spreadsheet to track tasks, it's often a good idea to copy our task ideas into these alternate slots too. We typically use some kind of styling (e.g. italics) to indicate that these are references to a task mentioned elsewhere in the tree.

---

Next: [Different tasks for different user groups](#)

## Different tasks for different user groups

- Reasonable pretending
- Mixing and splitting user groups
- Tracking who can do each task
- Splitting tests based on different tasks

So far, we've assumed that we're creating a list of tasks for all of our participants to try – that is, a single pool of tasks that doesn't care about who each participant is or what they know.

The reality is that **our site is probably servicing several different user groups**. As a power company, for example, we may have residential customers, farmers, and businesses. As a bank, we may have new customers we're trying to convert, and existing customers we want to keep happy.

*On the tree side*, one of the biggest challenges for an information architect is to come up with a single site structure that works well for all major user groups. So **we're typically testing the same tree on all user groups**. (We may be testing several variations of the tree, but *each variation* is being tested to make sure it performs well for all user groups.)

*On the task side*, however, we can't always try the same set of tasks with every user group, because **sometimes user groups are too different**.

In *Which part of the tree?* in Chapter 6, we saw that the Shimano company has three major user groups – cyclists, anglers, and rowers - as reflected in their top-level navigation:



Suppose we create a list of tasks that includes cycling gear (e.g. bike chains), fishing information (e.g. tips on fly-fishing), and rowing specs (e.g. the weight of carbon-fibre oars). Then we launch the tree test and invite people from all of these user groups to participate.

Imagine a cyclist participant, who knows nothing about fishing or rowing. They can reasonably attempt the cycling-related tasks, but they have no clue about the fishing and rowing tasks. It may be that they don't understand the task and its terminology, or they understand it but don't know the subject matter well enough to know where to look, or it may be that they just don't care about it. In any case, their results will be suspect. If they skip the task or wander around a lot, we'll never know if it was because they understood the task (but the tree didn't work for them), because they didn't understand the task at all, or because they had no interest in that task. The result is the same - garbage in, garbage out.

**We should only ask participants to do tasks that their user group might conceivably do on the real site.** It doesn't have to be something they have actually done themselves, but it should be something that they might reasonably expect to do at some point.

## Reasonable pretending

There's also a middle ground that comes up frequently – **tasks that a given user group may not do, but that they could reasonably “pretend” to do, because the knowledge is not too specialized.**

Suppose we are redesigning a bank site, and the two user groups we've identified are prospective customers and existing customers. We've jotted down tasks for each group, and we're deciding if we can test them all together – a single tree test with a single, combined group of tasks.

Will either user group run into a task that they don't understand, or that they couldn't pretend to switch hats for and have a reasonable chance of success?

- The *existing customers* were once prospective customers, so they should be able to do a good job pretending. Yes, they know more about the bank than new customers, but we may decide that this is OK to ignore for the purposes of this study.
- The *prospective customers* may not know as much as existing customers about this particular bank site, but they are presumably existing customers at another bank, so they may know enough to be able to pretend to be existing customers.

So, in this example, it's probably OK for us to give the same set of tasks to both user groups. We'll still need to identify which participant belongs to which user group, but this is easily done using survey questions – see [Adding survey questions](#) in Chapter 8.

**Not all pretending is reasonable, though.** If we're designing a medical site that serves both patients and doctors, it's unlikely that either group could reasonably pretend to be the other:

- Medical knowledge is highly specialized – doctors have it (lots of it) and most patients don't.
- Even for the same medical topics, doctors and patients use different terminology.
- They are probably going to be looking for different things, in different ways, on the site.



In this case, we would almost certainly give each group their own set of tasks, which means running a separate test for each group.

### Mixing and splitting user groups

There are also cases where some user groups are similar enough that they can share tasks, and other user groups are different enough that they need their own set of tasks.

Recall the power company we mentioned earlier. They had the following groups of customers:

- Residential customers
- Farmers
- Small/medium business
- Large business

For this tree test, we came up with a list of tasks that covered each of these groups. We then reviewed each task to see if there were any user groups that wouldn't normally do that, or couldn't reasonably pretend to do it.

- Every customer has a home, so all the residential tasks were good to go.
- The farm tasks and small-business tasks were a bit more specialized, but nothing that a home customer would have trouble understanding. We did tweak the wording a bit to be less technical on certain tasks.
- The corporate tasks were mixed – some were general enough that anyone could understand and pretend to do them, but some were quite specialized (e.g. signing up as a customer using a tender process) and we didn't think the other user groups would reasonably be able to do those.

This suggested that we should:

- run one tree test combining residential, farm, and small-business participants
- run the corporate participants separately with their own set of tasks.

### Tracking who can do each task

The simplest way to track who can do each task is adding an "Audiences" column to our spreadsheet, noting which user groups could reasonably do that task:

Level 0	Level 1	Level 2	Tasks	Audiences	Notes
BikeSite	Commuting by bike				
		Bike routes	Where are the cycle paths in Wellington?		
		Skills training			
		Cycling and the law	Is it legal to bike in a bus lane?	everyone	
		Tips for commuting			
	Recreational/family				
		Places to cycle			
		Learning to ride	Add a task for this or "skills training"	everyone	
		Safety tips			
	Road training/racing		No tasks for roadies in this round		
		etc.			
	Mountain biking				
		etc.			
	Touring & camping		Find bike-friendly campsites near Dunedin.	trippers	
	News & events				

That usually suffices if there are only a few tasks that are specific to each group. But if it starts getting more complex, or we have several user groups, we may want to create a **separate spreadsheet tracking tasks against audiences**, like this:

task ideas for schools	school parent/student	school board	school principal	school teacher	school admin
study awards for autism teachers	no	OK	OK	OK	OK
support for blind children	OK	OK	OK	OK	OK
National Admin Guidelines templates	?	OK	OK	OK	OK
how schools can estimate funding from the Ministry	no	OK	OK	OK	OK
advice on confiscating property from students	OK	OK	OK	OK	OK
materials for teachers	OK	OK	OK	OK	OK
leadership newsletter	no	OK	OK	OK	OK
Education Review Office	no	OK	OK	OK	OK
The National Library	no	OK	OK	OK	OK

## Splitting tests based on different tasks

In practical terms, **different tasks for different users = different tests**.

The simple case is that we find that all of our tasks could be reasonably attempted by all of our user groups. In that case, we could run a single tree test and invite everyone to participate. (We would still want to identify which user group each participant belonged to, so we can analyze for differences, but that's a separate (and easy) nut to crack – see [Adding survey questions](#) in Chapter 8).

**However, if we have tasks that some of our users cannot reasonably do (because those tasks are aimed at a different user group), we'll need to set up separate tree tests.**

Don't panic – setting up a separate test for a specific user group is usually easy. We already have the tree (everyone is being tested on the same site structure), so we're just changing out some tasks. Online tools typically have a "Duplicate Project" feature that makes it easy to copy and tweak the initial test setup.

When we set multiple tests based on different user groups, each test will usually have both of the following:

- Tasks that are common to all groups (e.g. "Find the company's mailing address.")
- Tasks that are specific to the target group (e.g. "You buy electricity for a large corporation. Does Acme Power use a tender process?")

In the power-company example mentioned earlier, we determined that residential customers, farmers, and small business could all understand each other's tasks, while corporate customers had some tasks that only they understood. We would then set the tests like this:

- Test 1 (residential, farms, small business)
  - 14 non-corporate tasks
  - 8 tasks shown to each participant
- Test 2 (corporate customers)
  - 10 non-corporate tasks (a subset of the 14 above)
  - 4 corporate tasks
  - 8 tasks shown to each participant

This means that, once the tests are run, we can compare 10 of the 14 tasks across all user groups, while also getting data on 4 corporate-specific tasks. Most importantly, **we did this without giving participants a task they couldn't reasonably do**.

Setting different tasks for different tests means that we'll also need to check our coverage separately for each test. In the example above, we may decide that corporate customers are more likely to use the *About Us* section, so you may shift some tasks from other areas (e.g. power for farmers) to tasks targeted at *About Us*.

---

Next: [Collaborating on tasks](#)

## Collaborating on tasks

If we're working alone on a tree test, we'll probably use a local file (e.g. an Excel spreadsheet, an OpenOffice document, etc.) to work on our tree and tasks.

In many cases, though, we'll be collaborating with others who work on the website too. In this case, we again recommend using a multi-user online spreadsheet (such as Google Sheets or Office 365) to provide a single document that shows everyone's latest updates.

In particular, we've found that **creating tasks is a job that lends itself well to a "divide and conquer" method of online collaboration**. There are often different people who know different parts of the website best, so if we can assign those people to create realistic tasks for their parts of the site, everything gets done faster (and everyone is more engaged in the exercise).

To do this, we create an "Assigned to" column in the spreadsheet, agree within the team on who is assigned to what, and fill in their names accordingly:

Level 0	Level 1	Level 2	Assigned to	Tasks
BikeSite				
	Commuting by bike		Sapna	
		Bike routes	Sapna	Where are the c
		Skills training	Sapna	
		Cycling and the law	Sapna	Is it legal to biki
		Tips for commuting	Sapna	
	Recreational/family		Christine	
		Places to cycle	Christine	
		Learning to ride	Christine	Add a task for t
		Safety tips	Christine	
	Road training/racing		Tony	No tasks for roa
		etc.	Tony	
	Mountain biking		Tony	
		etc.	Tony	
	Touring & camping		Fernando	Find bike-friendl
	News & events		Tony	

Here's an example of how this can speed things up and get everyone engaged:

On a recent IA-redesign project for a film organization, the project sponsor wanted their whole staff to get involved. They weren't kidding - they brought 11 people to the first workshop. We decided to take advantage of the numbers by assigning a small part of the site structure to each person. For the last half-hour of the workshop, we asked them to go back to their desks, open the shared spreadsheet in their web browser, and create a handful of tasks for their respective section.

While we worked on other project items (incentives, recruiting, etc.) with the project sponsor, we kept the spreadsheet open on the data projector in the meeting room. At the end of the meeting, we scrolled through the spreadsheet and found that everyone had finished their work and we had more than 40 task ideas ready to review. Go team! 😊

Next: [Writing a good task](#)

## Writing a good task

- Avoiding matching words
- Avoiding matching sequences
- Avoiding ambiguity
- Specific task = specific answer
- Using real-life situations and language
- Keeping it short and direct
- Varying phrasing to maintain interest
- Test-driving tasks

So far, we've talked about "task ideas" – rough jot notes about things we think would make good "find it" questions for our tree test.

Eventually, though, these need to be refined into properly worded tasks for our participants. But what do we mean by "properly worded"?

And how hard is it, really, to write a task? We just need to tell our participants to go look for women's jewelry, right? What could be hard about that?

OK, *here's the truth* - after running scores of tree tests for a wide range of clients, and after reviewing dozens more run by organizations on their own, we can truthfully say:

Without guidance and experience, each of us will write some *horrible* tasks.

We're not saying that *all* the tasks we've seen are horrible, but a surprising number are. And it's not just first-timers – some people repeatedly put tasks out there that are just not going to give them the clear results they want. Their intentions are good, but they keep making the same mistakes over and over.

Luckily, these mistakes are usually easy to fix. We can eliminate most or all of them by following the guidelines below.

## Avoiding matching words

**The most common cause of bad tasks, by a long shot, is using "give-away" words that point the user to the right answer.** We ask them to find something, and *that exact word* jumps out at them from the topics they're looking at.

For example, suppose the task is:

*Find out who to contact about your warranty.*

...and these are the top-level topics:

- Products
- Support
- About Us
- News
- Contact Us

99% of our participants are going to click *Contact Us*, not just because it's correct, but because the word "contact" matched the task. They didn't even have to think about it.

**We want participants to choose topics because they seem the best choice among alternatives, *not* because of simple word-matching.**

While this seems easy enough to do, it turns out that *all* of us are guilty of using keywords in some of our tasks (or, at least, in our first drafts of the tasks). That's because we're immersed in the site's content, its structure, and its jargon. Once we're in deep, it's how we naturally think and talk about the site, so it inevitably spills over into our task phrasing.

The cure is obvious – we need to **avoid those keywords, paraphrasing and substituting synonyms to end up with the same meaning.** Usually this is straightforward, especially if we remember to "speak" conversationally, as our participants would to each other.

Here's an example taken from an intranet tree test:

| *“You need to get reimbursed for expenses. Find the expense form.”*

And here's the tree:

- Library
- Expenses/Reimbursements
  - Rules for claiming expenses
  - Expense forms
  - Who to contact
  - Travel

Only a card-carrying cretin would get this one wrong, but what about the 99% who got it right? Did they succeed because they thought and decided that *Expenses/Reimbursements* was the best choice (as they're supposed to do), or because they simply matched the words “reimburse” and “expense”? Even someone who didn't understand English could find the right answer here based on word matching. So while we get a high success rate for this task, we still don't know if this was a result of the clarity of our tree or the wording of the task itself. We haven't properly “tested” the tree.

It's easy to solve this one, because it follows a common pattern; the tree uses jargon, and the task does too. If we just change the task to sound more like what a real employee might say in conversation, it might come out something like this:

| *“You paid for some work-related supplies on your personal credit card. You need to fill out the paperwork to get paid back.”*

Notice that we've avoided using the term “reimburse” (easy) and the term “expense” (a bit harder) by replacing them with synonyms. The revised task is a bit longer, but it's still clear and doesn't give anything away. Now, participants will have to make their own connections by *thinking*, not by just playing “spot the matching word”.

When phrasing a task, avoid keyword matches in the tree.

## Avoiding matching sequences

We also want to **avoid matching sequences between our tasks and the tree**. This happens when we phrase a task so that it uses the same order of words that the tree's headings use.

For example, an IT department used this task in their testing:

| *“For a fixed asset such as a laptop, how would you request an upgrade?”*

Their tree:

- Fixed assets
  - Computer equipment
    - New purchases
    - Upgrades
    - Warranties and repairs
  - Furniture
  - Heating/cooling/ventilation equipment
- Non-fixed assets

This task achieved a high success rate, but like the word-matching discussed above, we can't tell if this success was because of the clarity of the tree or the wording of the task.

In this case, the word-matching was **made worse by the sequence-matching** of fixed assets, computer equipment (laptop), and upgrades. Because the task used the same sequence of terms that the tree used, we gave away the answer to our participants, so we can't trust the resulting high score.

Luckily, sequence matching is easy to fix; we changed the phrasing of the task to use a different sequence, and used a more everyday tone:

| *“You want to request more memory for your company laptop.”*

## Avoiding ambiguity

Another common task mistake is **ambiguity** – **wording a task so it could be construed more than one way**. We may think the task is clear (after all, we wrote it), but will all of our participants understand what we meant?

Consider this example from an online-banking study:

| *“How would you get notified when your account balance is low?”*

We thought this task wording was fine – until we piloted it with some members of the project team.

- We intended the task to mean “How would you set up a notification?”, and half the team thought the same thing. The answer was in the *Setup* section.
- However, the others thought it meant *receiving* the notification itself, so they went to the *Secure Mail* section.

Once we discovered the problem, of course, it was easy to fix. We revised the task to:

| *“How would you arrange to get notified when your account balance is low?”.*

This meant the same thing to everyone, and we managed to avoid using the give-away phrase “set up”.

Avoiding ambiguity is hard, because it doesn’t usually seem ambiguous to the writer. **The best way to avoid ambiguity is to do the simple paraphrasing exercise** we describe in [Test-driving tasks](#) below.

## Specific task = specific answer

We should beware of tasks that are too broad – that is, **tasks that have so many answers that we don’t learn much from the results**.

Note that it’s OK for a task to have several correct answers. In fact, a mark of an effective tree is that it supports more than one reasonable path to success, when we find that substantial numbers of users follow those different paths.

But that’s only OK when the task is specific, and there are **one or more specific answers that we’re looking for**.

If the task is too broad, however, we may find that most of the headings in a certain section could be considered correct. Consider this example from a power company’s website:

| *“You’re looking for a new power company. What advantages does Acme Power offer you?”*

The tree offers the following correct answers (marked here in bold green):

- Why join us?
  - **Renewables AND gas**
  - **Discounted pricing**
  - **Smart metering**
  - **Online tools**
- Your account
  - **Sign up for online billing**
  - How to read your bill
  - **Track your usage**
  - (etc.)
- About us
  - Staff
  - In the community
  - (etc.)

One problem here is that the entire *Why join us* section is correct – they are all reasons to join Acme Power. Yes, we learned that the section is labeled well, but we don't learn anything more specific.

Worse, there are several other topics (highlighted above) that could be considered correct. The ability to get online billing and usage tracking are selling points, as is the community work that Acme Power does. These weren't intended to be the targets, but they need to be counted as correct for this task. If the intent of this task was to find if people knew what the *Why join us* section is for, these extra answers just muddy the water.

If we made this task more specific, we could evaluate the tree's effectiveness better. For example, we could revise the task to:

| *“You're looking for a new power company. Does Acme offer a way to measure your consumption on the web?”*

Now it's easy to determine which answers are correct (marked in bold green), and we find out if those topics are clear and distinguishable from their siblings and parents:

- Why join us?
  - Renewables AND gas
  - Discounted pricing
  - Smart metering
  - **Online tools**
- Your account
  - Sign up for online billing
  - How to read your bill
  - **Track your usage**
  - (etc.)
- About us
  - Staff
  - In the community
  - (etc.)

## Using real-life situations and language

In the same vein as being specific, we also want to use real-life situations and real-life “things” in our tasks, even if the tree we're testing doesn't.

For example, we tested an intranet site for an IT department. Their tree included the following section:

- Requests
  - Fixed-asset requests
  - Fixed-asset upgrades
  - Software requests

When they wrote the first draft of tasks, one of them was:

| *“How would you request a fixed-asset upgrade?”*

Ouch. Some major word-matching problems here. But no worries, this was just a first draft; the idea is there, it's just the wording that needs work.

However, there's another problem here beyond the word matching. Even if the task didn't use give-away words from the tree, it's still badly worded. Why? Because **the user is being asked to do something abstract**, something that sounds institutional and artificial. When was the last time we decided to “request a fixed-asset upgrade”?

To turn this into a good task, we need to **move it into real life**. In our example, it turned out that “fixed-asset upgrades” were IT-speak for “computer upgrades”, so we revised the task to:

| *“Your laptop needs more memory. How would you ask IT for this?”*

We've done two things here:

- We've **replaced an abstraction** (upgrade a fixed asset) with a **specific real-life situation** (asking for more memory for a laptop).
- We've **revised stilted institutional jargon into everyday conversational language** (how real people talk to each other).

The result is a task that is both easier for participants to understand and a better test of the clarity of the tree.

## Keeping it short and direct

Where possible, we should **keep our tasks concise – no longer than necessary to get the idea across clearly**.

Yes, we want to give them a real-life situation (as we saw above), but we don't want to write a novel. Consider this example:

*Your brother has decided to try running a marathon, and that will take a lot of training. His old running shoes are in sad shape, so you volunteer to find him a new pair at this sporting-goods website. Where would you look?*

Long-winded tasks like this pose several problems for participants:

- They're intimidating to see at first glance.
- They're tiring to read. Imagine doing 10 tasks like the one above – phew!
- They often end up providing too much detail, confusing the participant and forcing them to read it again.

When we're writing a task (and particularly when we're revising a task that was unclear in its first version), we must be careful about how long our text becomes. We should put it away and come back later, or ask a colleague to edit it down for us. It's easy to get stuck when we're trying to do it by ourselves.

Here's the example above, after some judicious revising:

*Help your brother find a new pair of running shoes for marathon training.*

## Varying phrasing to maintain interest

Some people write their tasks using the same formula each time, like this:

*Where would you find men's running shoes?*

*Where would you find gold necklaces?*

*Where would you find electric lawnmowers?*

**Generally, it's OK to repeat the sentence format.** Tree tests are typically short, so it's not too likely our participants will abandon the study just because the task wording is boring.

In the longer term, though, if we'd like those participants to continue doing our studies, **making the tasks a little more varied is one way we can keep their interest.**

Here are some common phrasings that we've used in our studies:

*Your son needs some new running shoes.*

*Your son's running shoes are worn out. Look for a new pair.*

*Find out if this site sells running shoes for men.*

*Running shoes would make a great birthday gift for your brother. Find them.*

## Test-driving tasks

Following these guidelines will improve the wording of our tasks. But **wording is harder than most people think**, and it's likely that our first draft of tasks will still contain at least one clunker.

No worries, though, because we can quickly reveal any remaining problems by doing this **simple paraphrasing exercise**:

For each of our tasks:

1. We get a colleague to read the task.
2. We ask them to paraphrase it back to us. That is, we have them explain (specifically) what they think we want them to look for.



3. We make sure that what they understand is what we intended. If there is any doubt, we discuss it until we're sure one way or the other.

If we do go back and revise some wording, we shouldn't assume we've fixed it first try. Repeat this paraphrasing exercise, but with a different person who didn't see the first version.

---

Next: [Identifying correct answers](#)

## Identifying correct answers

- Multiple answers
- Intermediate vs. leaf nodes
- How correct is correct?
- Correct now vs. correct later
- Changing the tree or tasks after marking answers

Tree testing is not just about seeing where people go to find things. It's about seeing if they go to the **right place**, where the real site would give them the content they're looking for.

At some point, then, we need to decide where the right place is. And if we're using tree-testing software, we need to tell it too, so it can tally up the results for us.

## Multiple answers

When it comes to correct answers, the most important thing to remember is that **there are often several correct answers for a given task**.

When we initially set up the tasks and tick off their correct answers, we'll spot at least a few tasks that have more than one "hit" in our tree.

### Correct answers

1. Where are the cycle paths in Wellington?

#### BikeSite

##### Commuting by bike

- Bike routes
- Skills training
- Cycling and the law
- Tips for commuting

##### Recreational/family

- Places to cycle
- Learning to ride
- Safety tips

##### Road training/racing

- etc.

**We may also miss a few correct answers.** After all, some trees are large, a given task can sometimes be interpreted more than one way, and our prep time is often limited.

The good news is that it's OK to miss a few up front. That's because **our participants will probably find more correct answers during the test**. When we analyse the results, we may see that some of these "wrong" answers really would find the right content. Once we confirm these are right, we can then mark them correct and recalculate our results.

Suppose, for example, that we've marked two correct answers for the task below:

## 4. Your 6-year-old wants to graduate to a two-wheeler. What's the best way to teach them?

## BikeSite

## Commuting by bike

- Bike routes
- Skills training
- Cycling and the law
- Tips for commuting

## Recreational/family

- Places to cycle

## Learning to ride

- Community classes
- Learning on your own
- Beginner tips
- Safety tips

## Road training/racing

- etc

When the test is over and we're analyzing the results, we notice that many participants chose *Beginner tips* as their answer. When we think about it more, we realize that this is a reasonable answer as well, so we go back and mark it correct and recalculate our results.

### Intermediate vs. leaf nodes

Occasionally, we may create a task where the answer could be any of the subtopics under a topic.

In the example below, all topics under *Learning to ride* could be considered correct:

## 4. Your 6-year-old wants to graduate to a two-wheeler. What's the best way to teach them?

## BikeSite

## Commuting by bike

- Bike routes
- Skills training
- Cycling and the law
- Tips for commuting

## Recreational/family

- Places to cycle

## Learning to ride

- Community classes
- Learning on your own
- Beginner tips

- Safety tips

## Road training/racing

- etc.

Sometimes, this happens because our task is not specific enough - see [Writing a good task](#) earlier in this chapter.

If, however, we decide it's OK for the entire section to be correct, we'll need to do one of the following:

- Mark each subtopic correct, or
- Mark the parent topic correct

Depending on the tree-testing software we're using, we may not be allowed to mark a parent (intermediate) topic as correct – only leaf nodes (at the end of a branch).

Even if the software allows us, **we recommend NOT marking intermediate topics as correct answers.**

There are a few reasons for this:

- We want to see the **full path** that the participant takes through the tree, and we want to make sure that, even at the lowest level, they can choose the correct answer. If we let them choose at a higher level, we don't find this out.
- Often, **not all** answers in the subtree are correct, so marking the parent as correct is sloppy.
- Only allowing leaf nodes to be correct simplifies both the participant experience and the later analysis.

If we're using software that restricts us to choosing leaf nodes only, and we really want to make an entire section correct, there are two alternatives:

- Mark each subtopic as correct, or
- Delete the subtopics and mark the parent topic (now no longer a parent) as correct. This may work if we don't need those former subtopics for other tasks in our study.

## How correct is correct?

Some answers are clearly correct; many more are clearly wrong. But what about those answers that are kinda-sorta-OK-but-not-really-the-one-we-intended correct?

Using our bike example above, suppose that we have this task, with these answers initially marked as correct:

## 4. Your 6-year-old wants to graduate to a two-wheeler. What's the best way to teach them?

## BikeSite

## Commuting by bike

- Bike routes
- Skills training
- Cycling and the law
- Tips for commuting

## Recreational/family

- Places to cycle

## Learning to ride

- Community classes
- Learning on your own
- Beginner tips

- Safety tips

## Road training/racing

- etc

But when we get the results back, we find that many participants chose *Commuting by bike* > *Skills training* as their answer. It's not the answer we intended because we don't normally think of kids as commuters. On the one hand, kids can ride their bikes to school, and the skills training course does accept children, but it also assumes they already know how to ride a bike. Hmmm. How do we decide if this answer is correct? In general, how do we decide which borderline answers to accept?

This will be a personal call, of course, but we need to draw the line somewhere. Here's where we've drawn it in our projects:

- In the real site, if the borderline topic would give them **enough content to be helpful** (in our opinion), we mark it as correct.
- If the borderline topic will feature a **prominent cross-link** to the "right" topic, we mark it as correct. (By "prominent", we mean that nobody going to that page could miss it. We don't mean a "see-also" link parked at the right or bottom of the page, where it could be missed.)
- Otherwise, we mark it as wrong.

Whatever criteria we decide on, **we need to be consistent** across all our tasks and (more importantly) across all our tests. If we aren't consistent, we won't be able to compare scores reliably across different trees.

## Correct now vs. correct later

Normally, we mark our correct answers just before we launch our test. However, it's common for participants to reveal more potentially correct answers **while the study is running**.

For more on this, see [Cleaning the data](#) in Chapter 12.

## Changing the tree or tasks after marking answers

In a perfect world, we would get our tree right the first time, or at least get our tasks right the first time.

The whole point of tree-testing, however, is that we know our initial creations will need revising, so we test to find out what needs to change.

When we inevitably make changes to our tree or our tasks, **we need to check how this affects the correct answers we previously identified**. There are 3 common scenarios to watch for:

- **The correct answers may have moved in the tree.**  
This is easy to fix - just update the correct answers in the tree.

- **The correct answers may have been removed altogether.**

We know of at least one test where revisions to the tree left one task without any correct answers. While it was instructive to see where people (vainly) went, it's definitely not something we want to do on purpose. 😞

- **Revisions create new correct answers.**

Whether we're revising the tree, the tasks, or both, it's not unusual for new answers to "emerge" from our changes. It's best to identify these before we run the test, but if we don't, our participants will usually spot them for us.

---


Next: [Entering tasks and their answers](#)

## Entering tasks and their answers

We may have been writing tasks directly in our tree-testing software. This is more likely if it's a simple test – a small tree with a small number of tasks.

However, if we've been using a spreadsheet to plan our test (a good idea for larger and more complex tree tests), we'll eventually need to transfer the finished tasks to our tree-testing software. Typically this is a simple (but manual) copy/paste exercise:

Level 0	Level 1	Level 2	Assigned to	Tasks	Ass
BikeSite					
	Commuting by bike		Sapna		
		Bike routes	Sapna	Where are the cycle paths in Wellington?	
		Skills training	Sapna		
		Cycling and the law	Sapna	Is it legal to bike in a bus lane?	eve
		Tips for commuting	Sapna		
	Recreational/family		Christine		
		Places to cycle	Christine		
		Learning to ride	Christine	Add a task for this or "skills training"?	eve
		Safety tips	Christine		
	Road training/racing		Tony	No tasks for roadies in this round	
		etc.	Tony		
	Mountain biking		Tony		
		etc.	Tony		
	Touring & camping		Fernando	Find bike-friendly campsites near Dunedin.	trip
	News & events		Tony		



Where are the cycle paths in Wellington?

Commuting by bike > Bike routes

Recreational/family > Places to cycle

Correct answers Post-task questions

We also need to mark the correct answers in our tree-testing software. The mechanics of this will depend on which software we're using, but in all cases, be sure to **mark all the correct answers for a given task**. (See [Identifying correct answers](#) earlier in this chapter.)

Next: [Randomizing the order of tasks](#)

## Randomizing the order of tasks

- You should randomize tasks
- Reasons for randomizing tasks
- Not randomizing the first task

### You should randomize tasks

Most tree-testing tools give us the option of randomizing the order of tasks that are shown to the participants.

If we don't randomize tasks, each participant will see the same tasks in the same order:

Participant 1	Participant 2	Participant 3
Task 1	Task 1	Task 1
Task 2	Task 2	Task 2
Task 3	Task 3	Task 3

If we do randomize tasks, each participant will get the tasks in a different (random) order:

Participant 1	Participant 2	Participant 3
Task 1	Task 2	Task 3
Task 2	Task 3	Task 1
Task 3	Task 1	Task 2

For most studies, we should randomize the order of tasks.

### Reasons for randomizing tasks

Why randomize tasks? Because we want to **reduce the learning effect** on our results:

- If we ask all participants to do the tasks in the same order, then they are likely to do better on tasks presented late in the test because they have learned at least some of the tree by then (by browsing it repeatedly). This gives late tasks an unfair advantage.
- By putting the tasks in a random order per participant, we make sure that task 15 (for example) is sometimes shown late, sometimes shown early, and sometimes shown in the middle. Mixing up the task order negates ensures that no task gets an advantage by always appearing late in the test.

### Not randomizing the first task

Some online tools provide an option to **not** randomize the first task; that is, **the first task in our list is always shown first** to participants, even if the other tasks are randomized.

The reason for this option is that most participants who do a tree test have never done one before, and no matter how good our pre-test instructions are, they may not fully understand how the test works until they've done a task.

The idea here is to provide our participants with a "warm-up" or "training" task, which we can exclude from our later analysis. (Participants may use this first task to click around the tree and explore a bit, even if they know the correct answer, so the results may be dodgy and therefore omitted from our scoring.)

What makes a good first "training" task?



- **The task is not too hard.**  
We don't want participants to be intimidated or dismayed at how hard our warm-up task is.
- **The most likely paths go at least 2 clicks down the tree.**  
We want participants to see that the tree is multi-leveled, and is only shown one branch at a time.
- **The results for this task are not important to our study.**  
Because this is a "throw-away" task, we want to pick something that we don't really care about (because we'll be ignoring this task's results later).

---

Next: [Letting participants skip tasks](#)

## Letting participants skip tasks

Most tree-testing apps provide the option of letting participants “skip” a task – that is, they can move on to the next task without providing an answer for this one. The question is, should we let them?

We recommend **always letting participants skip tasks**, because:

- This more closely mimics users’ actual behavior on websites. If they can’t find what they’re looking for, they frequently give up (and go elsewhere).
- If we force them to pick an answer, we’re likely to get a “garbage” result for that task – perhaps a semi-random answer that they pick after wandering semi-randomly around the tree. They could do this for any task, of course, but it becomes a much bigger analysis problem if they are not allowed to clearly “give up” by skipping that task.
- Participants may not enjoy skipping a task (because it’s “giving up”), but they enjoy being forced to make a choice (what they will likely consider a weak choice) even less.

---

Next: [Asking questions after a task](#)

## Asking questions after a task

needs content

---

Next: [Chapter 7 - key points](#)

## Chapter 7 - key points

The tasks we create should cover the **most common and critical activities** that visitors will do at our website, and any “**suspect**” **areas** of our tree.

We can “borrow” tasks from any **previous card sorting or usability testing** that we’ve done for this site.

We should create as many tasks as we need to cover the key parts of our tree, but **we shouldn’t ask a given participant more than 8-10 tasks**.

**We should decide if we need different tasks** (and therefore separate tests) for different audiences, or if they can all “reasonably pretend” to do each other’s tasks.

Avoid the most common task pitfalls by using our guidelines to **write clear, effective tasks**.

We need to **be careful and consistent in marking our correct answers**, because there may be more than we expected.

In most cases, we should **randomize the order of tasks** to reduce the learning effect.

In most cases, we should **let participants skip (give up on) tasks** to avoid user frustration and pollution of the results.

---

Next: [Chapter 8 - Setting up a test](#)

## 8 - Setting up a test

*"When you come to a fork in the road, take it." - Yogi Berra*

Once we've constructed one or more trees to test, and come up with a list of the tasks we'd like our users to try, we should take time to pause and congratulate ourselves – most of the heavy lifting is done.

Now we need to turn this raw material into an actual test that we can run with users. Luckily, the tools now available make this a fairly straightforward process, as well as offering a raft of handy options to tailor the test to our needs.

In this chapter, we'll cover how to get a test set up, and which of those options we should choose when.

---

### Naming the test

Tips to make our tests easy to identify later

### Disguising the test address

Avoid giving away the grouping method and client

### Selecting languages

For the prompts and controls shown by the tool

### Password-protecting the test

If we need to keep the study private

### Setting closing rules

Finishing manually, by # of participants, or by date

### Redirecting after the test

Linking back to a panel, to another study, or to a recruiting survey

### Setting up the tree and tasks

Pointers to Chapters 6 and 7

### Writing supporting text

Welcome message, instructions, thanks message, and T&C

### Adding survey questions

For screening, splitting results, research, comments, etc.

### Choosing a visual look

Setting the logo and color scheme

### Providing a support contact

For participants' questions, concerns, or suggestions

### Alerting the organization about our study

In case customers inquire about it

### Key points

---

**Next:** Chapter 9 - Recruiting participants

## Naming the test

---

All of the online tools let us name our tree test, so we can find it later. There's no special magic here, but we do recommend identifying:

- The site we're testing
- Which variation of the tree we're testing (if we're testing more than one)
- Which revision of the test we're working on

For example, suppose we're helping the Acme Supply company reorganize their website. We're testing their current site structure (to get a baseline score to compare against) and two new trees – one grouped by topic, and the other by audience. So we create 3 tree tests with the following names:

- Acme current
- Acme topic
- Acme audience

In [Chapter 10 - Piloting the test](#), we'll see that it's a good idea to do a dry run of each test, revise it, and launch the second (or third) version. So the list of tests may eventually look like this:

- Acme current – draft 1
- Acme current – final
- Acme topic – draft 1
- Acme topic – draft 2
- Acme topic – final
- Acme audience – draft 1
- Acme audience – final

Any name will work, of course, but if we do more tests over time, it does pay to be systematic now so we can make sense of our tests later.

---

**Next:** [Disguising the test address](#)

## Disguising the test address

---

Some tools let us specify the web address (URL) that participants will see when they do the test. This address is often auto-generated from the name we give to the test.

For example, for the “Acme topic – final” test shown in [Naming the test](#) earlier in this chapter, the test address might be something like this:

*<https://yourtool.com/acme-topic-final>*

There are two reasons why we might want to change this default address:

- **Giving away the grouping method**

An observant user may notice the address and see that the tree is based on topics (rather than, say, audience):

*<https://yourtool.com/acme-by-topic>*

This is a minor giveaway, but we still recommend changing the address to something less descriptive, such as:

*<https://yourtool.com/acme-tree-1>*

- **Giving away the client’s identity**

If we’re doing a blind test (where the participants are not supposed to know who the site owner is, for whatever reason), a quick glance at the web address can blow our cover:

*<https://yourtool.com/walmart-study-1>*

We can keep the test blind by using a generic address like this:

*<https://yourtool.com/study-1>*

---

**Next:** [Selecting languages](#)

## Selecting languages

- [A single language](#)
  - [Several languages](#)
- 

Most online tools let us pick one or more languages to present to our participants. This usually covers the tool-supplied prompts and UI controls that the tool shows to participants during the tree test.

### A single language

If we're running our test in a **single language**, then our job is simple:

1. We pick the tool language that we need, to cover the tool prompts and controls.
2. We write our content (tree, tasks, and other text) in that same language.

### Several languages

If we're running our test in **several languages**, most tools will make us create a separate test for each language.

For each instance of the test:

1. We pick the tool language that we need, to cover the tool prompts and controls.
2. We write our content in that language.
3. We make sure that any revisions we make (e.g. during our pilot testing) are applied to each instance of the test as needed.
4. When it comes to analyzing the results later, we'll need to manually compare the results between tests.

For more on multi-language testing, see [Multi-language testing](#) in Chapter 15.

---

**Next:** [Password-protecting the test](#)



## Password-protecting the test

---

Most tools provide a way to secure the test so that users must enter a password to participate. This is a basic (though not foolproof) way to protect confidentiality if we need it.

Because this is closely related to how we recruit participants for the test, we'll discuss it in more detail in [Restricting access with a password](#) in Chapter 9.

---

**Next:** [Setting closing rules](#)

## Setting closing rules

---

Part of setting up a test is deciding when to shut it down – when do we consider our test done?

The 2 biggest drivers are usually **numbers** and **time**:

- We may decide that we need 100 participants, so we run the test until we get that number. Note that this might take 3 days or 3 weeks. Some tools let us automatically close the study when we hit a specified total.
- We may only have a limited time to get our results – say, a week. Some tools let us set a date when the study will automatically close, and we have to be satisfied by whatever numbers we get.

In our studies, **we prefer to close the test manually**. While we always have a good idea of the numbers we want, and the time we can spend on this round of testing, we also like to have some wiggle room. Perhaps we got more responses than we expected, so we can close the test early and get started on the analysis. Or (more often) we get fewer responses than we hoped, so we might want to run the study for an extra week. Manual control lets us decide based on how the test is going.

Once the test is closed, there are a few more things to do – see [Closing the test](#) in Chapter 11.

---

**Next:** [Redirecting after the test](#)

## Redirecting after the test

---

Some tree-testing tools let us send the participant somewhere after they've finished our tree test. That "somewhere" is a URL that we can supply when we set up the test.

There are a few common reasons for setting up a "redirection" URL:

- **Rewarding commercial-panel participants**

If we're using a commercial research panel, we need to send the participant back there when they've completed our test, so the panel knows they've done the study and can then reward them. The research panel should supply us a URL to use along with a participant identifier.

- **Chaining several studies together**

We may want our participants to do more than one study for us. For example, we may want them to do a tree test and a survey, or perhaps (especially in the first round of IA research) a tree test of the old site and an open card sort to generate ideas for the new site. By supplying a URL to the second study, we can take them directly to the next study.

Note that we don't recommend chaining two tree tests together (for example, to test alternate trees) unless we have a way to mix up the order of the tests (to avoid bias and the learning effect).

- **Recruiting participants for further studies**

We can also use an unmoderated online study (such as a tree test) to help us recruit for follow-on studies. In her article [Two great online recruiting techniques](#), **Lisa Fast** describes how her team ran an online tree test, then linked to a "tell us more about yourself" survey that helped them recruit participants for a subsequent in-depth moderated study.

---

**Next:** [Setting up the tree and tasks](#)

## Setting up the tree and tasks

---

If we haven't already entered the tree into our online tool, we should do that now - see [Transferring the tree to a testing app](#) in Chapter 6.

For help on entering tasks into the tool, see [Entering tasks and their answers](#) in Chapter 7.

---

**Next:** [Writing supporting text](#)

## Writing supporting text

- Welcome message
  - Instructions
  - Thank-you message
  - Terms & conditions
- 

Unlike online surveys, tree tests are new to most participants. To reduce the number of people who drop out because “this is not what I was expecting”, we need to do a good job of holding their hand throughout the test. This is especially important at the beginning, when they don't know what to do next.

Most tools provide boilerplate text for the test, but some let us alter it to our needs. The following are tips for tweaking this text to make the study smoother for participants.

### Welcome message

This is the first thing participants see when they start a study, so it should concisely answer their two most pressing questions:

- What is this study about?
- How long will it take?

Here's an example of a welcome message we use in our studies:

*Thanks for agreeing to participate! This study will help us find out which parts of the Acme website need re-organizing.*

*This should take you about 5 minutes to complete, and you may find it more fun than a traditional survey.*

### Instructions

Because a tree test is not a conventional survey, and because most participants haven't done one before, **we will need to show them how it works**.

Ideally, the tool will do this for us, either by providing its own instructions (using text and pictures) or by guiding the participant through an actual sample task (that doesn't count in the final results).

Either way, the participant needs to understand that:

- We're asking them to browse a “skeleton” list of headings, not the real site.
- They can back up if they take a wrong turn.
- They can skip a question if they get lost (provided the tool allows this).

### Thank-you message

At the end of the study, it's always polite to thank participants for their time, and remind them how they've just helped us improve the next version of the website.

This is also an opportunity to direct them to other content that we'd like them to see (perhaps another study that they may be interested in), or to return to where they came from.

*All done, great!*

*Thanks again for helping us out. Your input will give us get a better idea of how we should organise the Acme website.*

*If you have questions about this study, please contact us at [info@acme.com](mailto:info@acme.com). We're happy to help!*

*You may now close this window or navigate to another web page.*

If we're integrating with a commercial panel, we may need to return to their page to register that the participant has completed the study and earned their reward.

## Terms & conditions

It's common to offer an incentive for user research (see [Offering incentives](#) in Chapter 9), and because tree tests typically only take 5 minutes to do, it's not worth paying each participant. This is why the incentive is usually a prize draw (e.g. "Do our 5-minute survey – win an Apple Watch!").

And if we're running a prize draw, we'll need to state the terms and conditions of the draw – who's eligible, what the exact prize is, when the draw will be done, etc. For an online study, the terms & conditions are usually summarized on a separate page of the site, linked from the study's landing page or from the email invitation we send to participants.

Luckily, the T&C for most studies is straightforward and mostly legalese-free. We can tweak an existing one that the organization uses, or start with this prize-draw terms & conditions template and customize it as needed:



---

**Next:** [Adding survey questions](#)

## Adding survey questions

- For screening participants
- For splitting results later
- For customer research
- For identification
- For comments and additional feedback
- For further studies
- Ask before or after the test?

Most tools let us add survey questions to our tree test. This can help us in several ways:

- We can ask questions that **help us analyze the results** (e.g. by comparing results between groups of participants).
- We can ask questions that **fill gaps in our user research** (e.g. things that we didn't ask in our last user survey).
- We can ask if participants would like to be **involved in future studies** for this website or product.

### For screening participants

Our tree-testing tool may provide the ability to screen participants based on their answers to opening questions: if they answer “correctly”, they are allowed to proceed with the test; if they don't, they are politely dismissed.

If the tool doesn't provide this option, there are other ways of screening participants – see [Screening for specific participants](#) in Chapter 9.

### For splitting results later

Probably the most useful way to use survey questions is for **filtering results later so we can compare between groups of participants**.

For example, if we post a web ad on our site that asks visitors to do a tree test, we may want to find out if return visitors do better than first-time visitors (presumably because they're already familiar with our site's structure). If we add a survey question that asks them “Is this your first visit to our site?”, we can later split up the results into two groups (yes and no) and see if and where their results differ.



1

Is this your first visit to our site?

Multiple choice, single answer (radio select)

Yes

No

+ Add option

Required

Random order

Note that we do need to plan our questions carefully to make sure we're getting the splits we want to compare.

- For more on testing different groups of users, see [Different user groups](#) in Chapter 9.
- For more on filtering results to analyze subsets of data, see [Analyzing by user group or other criteria](#) in Chapter 12.

### For customer research

Good user experience depends on knowing who our users are and what makes them tick. That means that every chance we get to learn from users, we should be trying to fill gaps in what we know about them, and continually asking new questions to guide our designs.

The main purpose of a tree test, of course, is to test the tree we've come up with, before we build the site. However, because tree tests are very brief (typically about 5 minutes), we also have a chance to ask a few fill-in-the-gap questions.

Most tree-testing tools let us add survey questions to our tree test. But before we jump in and add a whole slew of research questions, however,

consider the following:

- **Questions that help us analyze the results later** are usually more important than general research questions, so add those first.
- Tree-test tools typically do not offer the advanced capabilities of dedicated survey tools (such as if-then question logic and fancy reporting), so we should **keep our “mini survey” simple**.
- **Don't irritate participants with a long list of follow-up questions**. They have done what we asked by completing our tree test, and while they are probably willing to answer a **few** more questions, their goodwill will erode rapidly if we demand another solid chunk of time and effort from them.

**We recommend keeping the total number of questions to 5 or less**. Because we usually ask some data-filtering questions to help our analysis, this leaves us room to ask 2-3 general research questions at most.

If there are 5 or 10 (or more) questions that we really want answers to, we should consider creating a proper online survey to run separately from our tree test. We will be happier with the features offered by a full-on survey tool, and our participants will not feel like they've been tricked into doing more work than they signed up for.

## For identification

In most online studies, we do not need to know the individual identities of our participants. As long as they are part of our target audience, we're simply grateful that they take the time to help us with our research.

There are cases, however, where we do need to know participants' identities. For example, we will need their contact info if:

- We're rewarding participants individually for participating.
- We're running a prize draw as an incentive (a very common way to get participants for tree tests).
- We want to follow up with certain participants (based on their choices in the tree test or how they answer survey questions).
- We want to build a pool of willing participants for future studies.

## For tracking individual participants/rewards

There are cases where we need to know exactly which participants have completed the tree test. For example:

- An organization may want all of its employees to do a study for a new intranet that it's planning.
- We may be using a [commercial research panel](#) where panelists are automatically rewarded once they complete the study.

In this case, **we'll need a mandatory question that asks for a unique identifier from each participant**:

- For commercial panels, this is often a numeric identifier that the panel assigns to each panelist.
  - The identifier may be automatically passed to the tree-testing tool by the panel software, in which case we just need to set up the integration of the two tools. (See the tool's docs for details.)
  - If there is no automatic integration, we'll need to ask each panelist to enter the identifier that their panel gave them.
- For other situations, we'll need to find out which identifier to use (email address is a common choice), and ask each participant for it.

## For prize draws, follow-ups, and further studies

In most of the studies we run, we don't really need the contact info of every participant.

- If they're willing to give it to us (e.g. for a prize draw, for follow-ups on this study, or for future studies), that's great.
- But if they want to stay anonymous, that's fine too. We really just want their tree-test responses, and we don't want them to abandon the test because they feel they're being strong-armed to give up personal information.

In most cases, we typically just ask for an email address because it's easy (one field that they type all the time) and pretty much everyone has one.

Whether they want to give us their email address is another matter, so we need to be polite about it:



- **We make the email address optional.**

If they don't want to give it, that's fine (and we inform them that they won't be included in the prize draw, if any), but we still want their test results.

- **We promise not to spam them.**

We assure them that we will not spam them (by using their email for other purposes like advertising) or share their address with other organizations.

Here's an example:

*Your email address (optional - only used for the prize draw, or if you asked to be notified about future research. No spam!)*

## For comments and additional feedback

Unmoderated online studies are great because they can collect data for us 24/7, but that also means that we lose the ability to converse with users directly.

One way to get part of that benefit is to give participants an optional "comments" field where they can give feedback, ask questions, offer suggestions, and so on. It's pretty much guaranteed that we'll get a few snarky comments, but we're also likely to get a few gems that give us additional insight into how our users think.

If we want to extend the conversation, we can also ask participants if it's OK to contact them to follow up on their test results and comments. We've got some of our most useful insights from talking to participants who were very critical in their feedback during the test; these are users who are typically very invested in the product/website we're testing, and they're pleasantly surprised to be contacted by a real person who wants to hear more about their needs and issues.

## For further studies

If we're running a tree test, it's unlikely to be the only research we're doing for our project. And participants can be hard to come by, so why not have one study help the next one?

We often finish our survey questions with an invitation to do future studies about this product/website. It usually looks like this:

*To improve the Acme website, we're doing more studies like this over the next few months. Would you like to be notified to help us out?*

**In our experience, a lot of people are willing; we typically get a "yes" response rate of 40-60%.**

Note that we'll need contact info for these people (usually an email address), so we can notify them when our next study is open for business. If we already asked them for an email address for a prize draw, we just use that instead of asking for it again.

## Ask before or after the test?

Some tree-testing tools let us decide when the survey questions will be presented – before the tree test, after, or some of each.

We need to be careful about asking questions *before* we present the tasks. Our questions (and any multiple-choice answers we supply) can give away information about the structure of the tree or the terminology we're using in it.

Even if our questions don't explicitly give anything away, they may still have a "priming" effect on the participant, by putting them into a particular mindset before they try the tasks.

**In most cases, it's best to ask survey questions after the tasks**, because they won't affect the tree test itself. While most questions won't skew the results much, it's best to be safe.

---

**Next:** [Choosing a visual look](#)

## Choosing a visual look

---

Some tools let us adjust the look and feel of the participant session. Beyond setting the language and supporting text (discussed earlier), we may also be able to specify:

- **The logo displayed before/after/during the test**

If we're running this study for a client, be sure to get an approved logo from them.

- **The color(s) used for backgrounds and such**

Again, it's good to pick actual brand colors if we can, while making sure the resulting UI has **enough contrast for visually challenged participants**.

If we're using brand elements like logos and colors, we should be sure to **include the organization's "brand" people (usually the Marketing department) in our pilot testing**. We've worked with some companies that are very protective of their brand, and will read us the riot act if we deviate even slightly from their guidelines. Don't say we didn't warn you. 😊

---

**Next:** [Providing a support contact](#)

## Providing a support contact

---

It's always a good idea to give participants a person to contact if they have questions, concerns, or suggestions about the study. We usually do this by adding a sentence with a "mailto:" email link at the end of the invitation (whether that invitation is an email message or a web page that we've set up). It typically goes like this:

| *If you have any questions or comments about this study, please email us at [research@acmesupply.com](mailto:research@acmesupply.com)*

While we don't get many emails from this, it is important to be able help participants if they need it. The most common issues we've seen are:

- old/unsupported web browsers
- restricted connections (where the tool is blocked by an organization's firewall)
- accessibility (some tools do a better job than others in this regard)

---

**Next:** [Alerting the organization about our study](#)

## Alerting the organization about our study

---

One last thing to do when we're getting ready to run a tree test (or any other unmoderated research): **we should tell our organization's support channels that we're running a study.**

When participants are invited to do the study, whether by email or by clicking a web ad, they may want to check that it's legitimate. This is especially true for organizations such as banks who are frequent targets of phishing attacks.

Because of this, we may get a few people contacting our customer-support staff to ask if the invitation they received is for a real study. If we have informed them beforehand of our research, that makes things easy for them and for the participant.

When we inform support channels about the study, they may inform us in turn that they have certain guidelines and procedures for contacting customers. (In fact, a few larger companies are quite strict about this.) While jumping through these additional hoops is never fun, it's usually better to find out about them before our study runs than having to ask forgiveness for our ignorance of them afterward.

Finally, we should remember that informing support channels is not only polite and efficient; it's also a bit of **free internal publicity** for the UX work we're doing.

---

**Next:** [Chapter 8 - key points](#)

## Chapter 8 - key points

We should **name our tests systematically** so we can make sense of them later, and **choose URLs that don't give away anything important**.

We need to **decide how we want to close the study**. We usually close the test manually depending on how well (or badly) the recruitment is going.

At the end of a test, **we can link back to a commercial panel, to another study, or to a survey to recruit** for our next piece of research.

Tell users up front **how long the study will take**, and make sure the instructions **make it clear how a tree test works**.

If we're running a prize draw, we'll need to state the **terms and conditions** of the draw. Tweak the organization's template or customize our sample.

Adding **a small number of survey questions** to the test can help us analyze the results later and provide willing participants for subsequent studies.

We should **provide a support contact**, and **let our organization's support channels know** that you're running a study.

---

**Next:** [Chapter 9 - Recruiting participants](#)

## 9 - Recruiting participants

*"O ye'll take the high road, and I'll take the low road,*

*And I'll be in Scotland afore ye." - The Bonnie Banks o' Loch Lomond*

User research requires – you guessed it – **users**.

Whether we're running an online study or an in-person one, we will need to get a certain number of representative users to volunteer their time to do our tree test.

In some cases, recruiting is easy. We put a study invitation on our high-traffic website, some fraction of visitors click it and do the test, and we're done.

In many cases, however, our job is more difficult. Perhaps we're looking for a certain subset of users. Perhaps we don't have an existing website to draw traffic. Perhaps our audience is small enough that it's hard to get enough users to volunteer.

In this chapter, we discuss how to decide which (and how many) participants we need, how to get their attention, and how to persuade them to do our study.

---

### How many participants?

~50 per user group, depending on the # of questions per participant

### Different user groups

Recruiting for separate tests vs. a single "everyone" test

### Using web ads

Choosing sites/pages/position, creating the ad, and explanation pages

### Using email lists

Inviting in batches, filtering lists, opting out, etc.

### Using social media

Who to target, what to say

### Using commercial panels

How panels work, quality of participants, and caveats

### Using integrated recruitment tools

Similar to commercial panels, but easier to use

### Other ways to recruit

Mechanical Turk, trade groups/forums, friends & family, etc.

### Dealing with selection bias

What it is, what causes it, and how to reduce it

### Coordinating audiences and channels

Recruiting multiple audiences using multiple channels

### Screening for specific participants

Filtering databases, using targeted email lists, and asking explicit questions

### Restricting access with a password

Sending the invitation separately from the password

### Writing a good invitation

4 tips and a template

### Offering incentives

Usually necessary, and usually a prize draw

### Recruiting for in-person sessions

Getting the right participant in the right room at the right time

### Key points

---

**Next:** [Chapter 10 - Piloting the test](#)

## How many participants?

- Counting by user group
- More participants for fewer questions

| *“How many people do we need to get?”*

This is the #1 question we hear when we help clients run online studies. And because we're consultants, our stock answer is “It depends.” 😊

The simple answer is:

**Aim for about 50 participants per user group.**

The more sophisticated answer is:

- **30 participants will start showing patterns in the results**, but it will be hard to know what to do with “small effects” because we don't have enough participants to know if these are real effects or just outliers.
- **50-100 participants will make the patterns much clearer**, and we'll be able to identify which results are significant and which can be discarded as noise.
- **Hundreds of participants give diminishing returns**, and we're potentially “using up” participants who might be better employed as fresh participants in a subsequent round of testing.

For a more rigorous look at how many participants we should aim for, see this [MeasuringUsability article on tree testing](#).

### Counting by user group

Most products/websites have more than one major type of user. In [Which part of the tree?](#) in Chapter 6, for example, we saw that the Shimano website has 3 user groups – cyclists, anglers, and rowers.

**If our study is covering several user groups, we'll ideally want about 50 participants for each group.** That way, we can filter the results by user group and still have enough data to see clear patterns in the results.

(We'll also need a way of identifying which participants belong to which user group. We often do this with survey questions – see [Adding survey questions](#) in Chapter 8.)

Some user groups are more important than others, and our pool of participants is often limited, so we try to get more participants from our major groups. If we end up with too few participants of a less important group, that's something the project team can probably live with.

### More participants for fewer questions

The other factor that affects how many participants we need is how many tasks (out of the total) that each participant does.

**If each participant is only asked a subset of tasks, we'll need proportionally more participants.**

To understand this, let's consider two cases:

- **Each participant does all tasks.**  
Suppose we have 10 tasks in our tree test – that is, 10 things that we want our participants to find, and this provides adequate coverage of the important parts of the site tree.  
Suppose we decide that each participant should do all 10 tasks. This is a reasonable number because, as we saw in [How many tasks?](#) in Chapter 7, 10 tasks makes for a quick test and minimizes the learning effect.  
Because each participant is doing all the tasks, we would simply aim for **50 participants of each user group**.
- **Each participant does half the tasks.**  
Suppose now that we actually wrote 20 tasks, perhaps because the tree is large and 10 tasks just wasn't enough to test everything we wanted to cover.



If we asked each participant to do all 20 tasks, the number-of-participants answer is the same – about 50 per user group.

However, we saw in the Tasks chapter that it's not a good idea to ask participants to do that many tasks: it takes too long, they get bored or tired, and they are more likely to "learn" the tree (which skews the results).

If we did the prudent thing and asked each participant to do 10 tasks, that's half the tasks in the test, so we would need twice the number of participants (about 100) to get each task "hit" by the 50 we're aiming for.

In the end, the formula for this is simple: if **we divide our total number of tasks by the number per participant**, this gives us a **multiplier** for how many participants we need. For example, if we have 20 tasks total and we ask 10 per participant, this would be  $(20 \div 10) = 2$ , so we'll need 2 times the normal number of participants. If we wanted 50 responses per task (the minimum recommended), that means we'll need to get  $(50 \times 2) = 100$  participants in total.

---

**Next:** [Different user groups](#)

## Different user groups

- [Identifying for separate tests](#)
  - [Identifying in the same test](#)
- 

As mentioned in [How many participants?](#), the tree we're testing may serve more than one user group. As discussed in [Different tasks for different user groups](#) in Chapter 7, there are two basic methods to testing multiple types of users:

- **Running a separate test for each user group.**  
Each test has its own set of tasks aimed at that specific user group. Our recruiting challenge is to point the right participants to the right test.
- **Running a single test with multiple user groups.**  
The test has a set of tasks that can be reasonably performed by several user groups. Our recruiting challenge is to identify (after the results are in) which user group each participant belongs to.

### Identifying for separate tests

If we're running a separate test for each user group, the test design is simple – we have our tree and a list of tasks that are aimed at that user group.

To recruit for this test, we would typically identify qualified participants first, then point them to the study. In other words, the qualifying step would be done **before they start the test**. For more on this, see [Screening for specific participants](#) later in this chapter.

### Identifying in the same test

If we're targeting several user groups with the same test, we first need to make sure that each user group can reasonably perform the tasks we've set. For more on this, see [Different tasks for different user groups](#) in Chapter 7.

Because they're all doing the same test, we need a way to identify each participant's user group. This is usually done by adding a survey question to the test. Later, when we're analyzing the results, we can filter them by this question to compare between the user groups.

Here's a typical survey question that identifies the participant's user group:

#### What is your main way of getting to work? \*

- Drive
- Public transit
- Walk or pedal
- I work at home
- Other

Note that, like any survey question, it should:

- Provide clear and mutually exclusive choices
- Include an "Other" option (unless we're sure we've covered every possible answer)
- Be written in the user's language (which may be different from the organization's terminology)

Sometimes, it may not be possible for a single question to determine the user group. If the testing tool lets us filter results based on combinations of survey questions, we can use 2 (or even 3) questions to determine the user group.

For more on how to filter results by user group, see [Analyzing by user group or other criteria](#) in Chapter 12.

---

**Next:** [Using web ads](#)

## Using web ads

- Choosing websites for the ad
- Where to place the ad
- Creating an ad
- Creating an explanation page

One of the most common ways of attracting participants is placing an “ad” on our website (and sometimes other websites as well).

The ad usually takes the form of an **at-a-glance proposition** (e.g. “Help us improve our site – win an Apple Watch!”).

Clicking the ad can either take the user directly to the tree test (where the study is briefly explained) or to a page on our website that explains the research and links to the tree test from there.

## Choosing websites for the ad

If we’re running a tree test to improve our own website, then that’s the obvious place to run the ad, because we’re automatically getting the right users.

If our website gets a lot of traffic, then the small percentage of visitors who click through to the study may provide the numbers we need to confidently analyze the results.

If our website doesn’t get enough traffic on its own, or if it’s a new website that we’re creating, then we’ll need to find other websites that:

- Attract the type of users who would conceivably use our website too, and
- Are willing to put our ad on their pages (for free, for money, or for some other arrangement we can make with them).

For example, we ran a tree test for a New Zealand government agency using web ads. Their own site didn’t get much traffic, so they contacted colleagues who ran other government sites and asked for help. Eventually their ad ran on half a dozen Ministry and department websites and they got a very healthy number of participants. (And they returned the favor later when those other sites ran their own online studies.)

If we do place web ads on sites other than our own, we should be sure that our ad and our explanation (discussed below) are clear about what they’re testing. For example, if we’re testing site A, and we place an ad for it on site B, make sure the ad is clearly for site A. Otherwise we run the risk of:

- Attracting users who may not be who we want
- Confusing or unintentionally misleading users who then abandon the study.

## Where to place the ad

The **global header** is usually the best (and often the easiest) place to put a web ad.

- It appears on every page of the site, so every website visitor has a chance of seeing it.
- Adding a “banner” ad across the top of a page is usually easier to do than trying to insert it into the various page layouts that the site uses.
- Many sites already have an “announcement” banner feature that we can use for the ad. We just provide the text and the destination link:

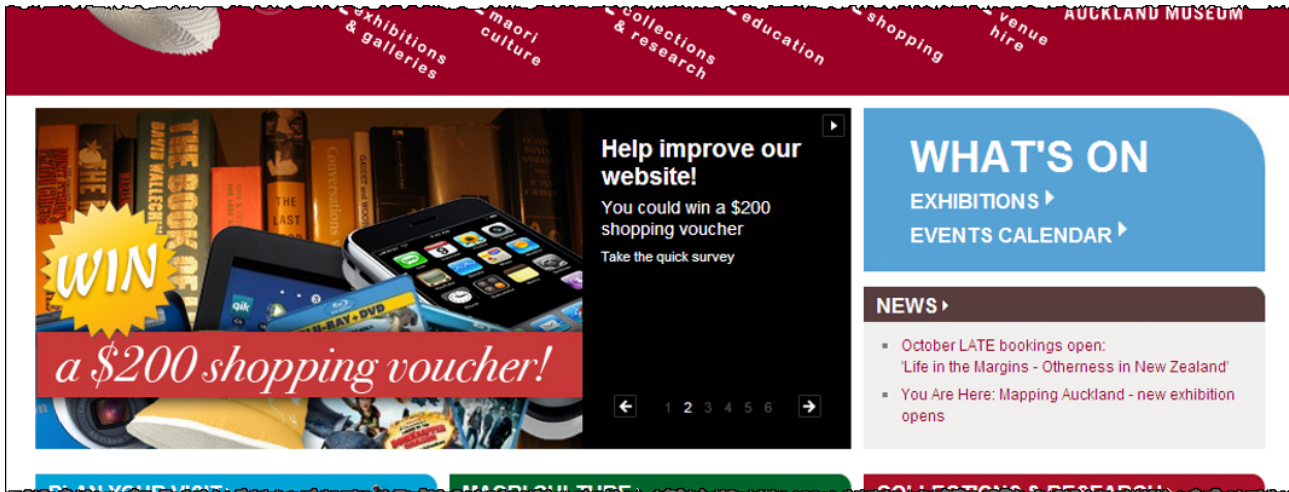
Help us make things easier to find! Do our [short survey](#) for a chance to win \$300.

Personal Business Agribusiness Investors About us Search Sign in

meridian Join now Pricing & rates Manage account Tips/guides Contact us

\$100 Free

The **home page** is another popular spot for a web ad, because it usually already has a "feature" area or carousel that we can use:



The problem with home-page ads, however, is that many visitors never see the home page because they jump directly into the depths of a site from a search engine.

If the site has a library of ads that it serves up on various pages, we can also get ours added to the mix. In this case, we just need to create an ad with the proper dimensions, graphic format, and so on.

If we're looking for specific types of users, we may want to run the ad on corresponding pages of the site. In the Shimano example we saw earlier (where they split their website into sections for cyclist, rowers, and anglers), if we were targeting cyclists, then we would obviously advertise our study on the cyclist pages of the site (and perhaps the general pages too), but not on the rowing or fishing pages.

We may also want to use **pop-up ads**, also known as "intercept" ads. These are often triggered when a user has been on a page for a certain period of time, when they scroll up, or when they try to leave the page. The more obnoxious variety appear in the middle of the screen, blocking the user's view of the content, and we don't recommend them. The more polite variety pops up in a corner of the screen (often the lower right), and tries not to get in the user's way.

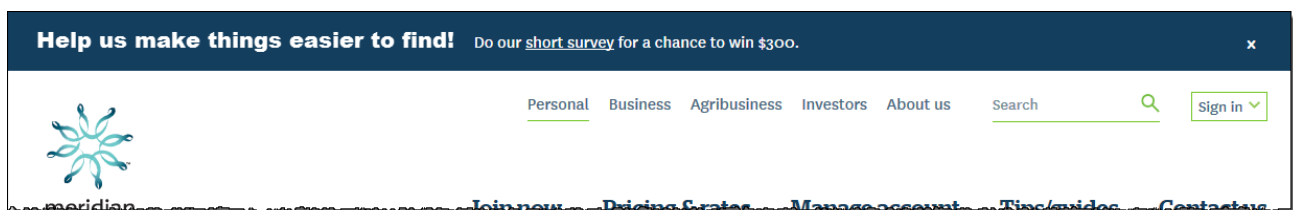
## Creating an ad

There are countless books and articles on how to create effective ads. For our online study, though, we really just need to focus on two factors:

- **Attracting attention to the ad**
- **Conveying the proposition at a glance**

To attract attention:

- **Make sure the ad appears in a spot that website visitors are likely to see.**  
Generally this means anything at the top of the page, especially the top center.
- **Use colors or graphics that draw the visitor's eye.**  
For example, when we do banner ads across the top of website pages, we usually pick a saturated color that stands out from the other content.



We can also use an image to create visual interest. If we're doing a prize draw (a common incentive), we'll sometimes use an image of the prize (e.g. an iPad).



Once we have a visitor's attention, we have a few seconds (at most) to convince them to participate. Like a roadside billboard, we are usually limited to the small amount of text that a "passer-by" is willing to skim.

We typically phrase the text as a simple value proposition: *Do this, and we'll give you that.* Here are some examples:

*Do our 5-minute survey, get in the draw for a tablet!*

*Help us make things easier to find - win an Apple Watch!*

We also need to make it clear what they should do next, if they decide to participate. This can be a clear text link (colored/underlined text) or a separate button.

**Note:** If we are advertising the study on a website different from our own, we'll need to add some context to the ad so that visitors are clear about what they're volunteering for.

## But it's not a survey, is it?

Wondering why we keep calling our study a "survey" in the examples above? To be sure, a tree test is not a survey; at least, it's not the traditional questionnaire that most people call a "survey".

The problem with using the term "tree test" is that **most people don't know what a tree test is**. When presented with an unknown like this, many people will turn it down just because it's something new that they aren't interested in learning about.

By calling our study a "survey", we are offering them something familiar, which is easier to say yes to. Once they click to the tree test, they'll find it's a bit different from a traditional survey, but most will proceed as long as it looks reasonably brief and easy to do.

## Creating an explanation page

Before participants actually do our test, we'll need to give them a bit of preamble, typically including items such as:

- What the study is for
- How long it will take
- What their reward is
- What we will do with the results
- Any other "fine print" that we need to include (such as terms and condition of the prize draw, if we're doing one)

We may want to cover this in the test itself (for example, on the welcome page), or we can put it all on an explanation page on our website. We typically do the latter, because many of our clients already do it this way for other online studies they conduct (such as user surveys).

While we do need to cover the items mentioned above (and perhaps other points specific to our research), we also need to do this briefly and concisely, and **give them an obvious way to start the study**. Otherwise, we'll see a fair number of people dropping out because this is all taking too long or looking a bit complicated.

Here's a typical explanation page that covers the basics and provides a clear way to start the study:



(Feel free to download and customize this example as needed.)

This content will probably be similar to what we put in our email invitations (if we're also recruiting by email) – see [Using email lists](#) later in this chapter.

---

**Next:** [Using email lists](#)

## Using email lists

- How many should we invite?
  - Inviting in batches
  - Filtering lists to get the right people
  - Letting people opt out
  - Hiding participants from each other
  - Who should send the email?
- 

Along with web ads, using email lists is a very common way to get participants for online studies.

- Many of our clients have maintain **customer databases** that they use for sales and marketing purposes. These often include useful data on demographics and product use.
- Even small organizations usually have **lists of customer contacts** (often stored in spreadsheets). These are typically modest in size and detail, but may still give us a good pool of people to invite.

One big advantage of using customer lists is that we're **contacting people who already have some kind of relationship with the organization**, usually as current users of their products or services. This relationship usually boosts the response rate, because these people are likely to have a vested interest in improving those products and services.

Another advantage of customer lists is that **we often get to pick who to invite**, usually based on information in the lists such as region, age, usage, and so on.

The downside is that, unlike passive web ads, **email invitations are an active (albeit minor) intrusion** into people's lives. Organizations should be very careful about how (and how often) they "bother" their customers with unsolicited messages, no matter how good the cause. For more on this, see [Letting people opt out](#) below.

### How many should we invite?

Earlier we recommended getting about 50 participants from each user group we want to test.

However, we all know that most people will ignore most email invitations to research studies like this. So, to get 50, we have to invite many more than that.

How many more?

- **Ask what the organization's traditional response rate has been.**  
Most organization have done email invitations at one time or another (usually for customer surveys), so they should have some idea of their response rate. Certain organizations with very loyal/vocal customers can get a 30-40% response rate, but most are much lower (often under 10%).
- **If unknown, assume a response rate of 5-10%.**  
Note that this number can vary greatly depending on factors like the desirability of the incentive (see below) and even the time of year (e.g. farmers are unlikely to participate during harvest).
- **Do the math to determine the number of emails to send out.**  
For example, if we expect a 10% response rate and we need 50 participants, we'll probably need to send about 500 invitations to hit our number.

For many organizations, this is more people than they have on their lists, so the question is not "how many should we invite?" but rather "how else can we get participants?". Luckily, we don't need to be tied to any one method of recruiting. Most of the studies we do include web ads AND email lists, and sometimes even then we have to start beating the bushes for more people – see the other methods described in this chapter.

### Inviting in batches

If we have access to a large list of customers (perhaps thousands), we may be tempted to email them all and get lots of results fast.

**Careful – emailing everyone in a large pool is a rookie mistake.**

- First of all, we should almost never email everyone in a big list. Lists of those size usually have more detail in them that we can use to filter the list down to the people we really want (not just bank customers, for example, but those with home loans who use Internet



banking). For more on this, see [Filtering lists](#) below.

- Second, even if we still have a big list after filtering (lucky us), remember that people have a limited appetite for invitations from a given organization. If we or anyone else in our organization wants to run another study in a month or two (remember that we recommend at least 2 rounds of tree testing to get it right), we should probably avoid emailing the same people we just pinged this week. Many large organizations have formal rules about this, typically along the lines of “Do not email a given customer more than once every 3 months”. Even if the organization has no such policy, it’s still a healthy rule of thumb.
- Third, we may not need that many responses to get the results we want. 50 responses shows us patterns, and 100 responses makes them clearer, but beyond that we’ll just get diminishing returns.

So, if we have a large number of potential invitees, we recommend **inviting them in smaller batches according to the expected response rate**.

For example, suppose we need 50 participants and we have 1000 people on our list. How many should we invite?

- If our response rate was known to be 10%, we would send 500 invitations.
- If we expect the rate to be higher, we might send out only 200-300 invitations in the first batch.
- After a few days, if we hadn’t reached our target of 50 participants, we could send out another few hundred. And so on until we reach our target.

The big win here is that we “save” a bunch of people to use on our next study; we’re rationing them so that we always have a pool of users to fuel our ongoing research.

The other factor at work here is urgency. Using batches slows down the study (because we wait a few days between batches).

- **If we need results fast, and we have users to burn (so to speak), we can invite larger batches of users.**
- If we don’t have many users on our list, and so need to conserve them, invite smaller batches so we can get just enough participants to show clear results.

## Filtering lists to get the right people

We don’t just want any 50 people to do our study; we want the *right* 50 people – people who match our idea of a representative user.

If our study is for all users, then a simple web ad or a blanket email blast to a customer list is probably OK. This should ensure that most of our participants are current (or past or future) users.

Often, however, we may want to get more specific about who does our study. If we are reorganizing the Large Business section of a bank’s website, for example, we want large-business users to test the new structure, but we don’t want personal-banking users because they do different tasks, use different terminology, and would generally be irrelevant to this study.

When we’re going after a specific user group, there are two common approaches:

- **Using customer lists that are specific to that user group.**  
For example, the bank may have a separate customer list for their business customers. If we invite people from that list, we’re automatically picking the right users.
- **Filtering a broad list down to the users we want.**  
The bank may have a customer database with fields that let us narrow down to just the business users.

Having a database that we can filter is very useful if we have specific recruiting criteria. While this is mostly used for targeted studies like in-person usability testing (we’re only testing 10 participants, so we want to be sure we get just the right users), it can also help improve our tree-test results. For example, we may want to recruit not just personal-banking users, but specifically those who use Internet banking frequently.

If we do want specific users, we can do your filtering early or late:

- **Early filtering** – We only send invitations to people who fit our criteria (by filtering a customer database first).
- **Late filtering** – We invite anyone to do the study (via a blanket email blast or a web ad), then screen out the people we don’t want (by using screening questions just before the tree test starts). For more on this, see [Screening for specific participants](#) later in this chapter.

## Letting people opt out

We mentioned earlier that inviting people by email is intrusive – most people have a limited appetite for unsolicited invitations, and some people may not want to be contacted at all. We need to respect their wishes and keep their goodwill.

There are two common ways to handle this:

- **Don't contact people who have already opted out.**  
Many customer databases have a field indicating whether the person has opted out of non-essential communications (often termed "marketing and promotional" messages). Obviously, we don't invite people who have opted out. Related to this is an **embargo period**, where we don't contact people too soon after we last contacted them. The database shows when the last contact was, so we only invite those who have not been contacted recently. (3 months is a typical waiting period.)
- **Make sure the invitation includes a way to opt out.**  
Most people who don't want to participate in our study will just skim the email and delete it. But there will be some who don't want to receive more of these invitations, so it's a simple courtesy to give them a way to easily opt out of future invitations. A clear link at the bottom of the message handles this. How we implement it (as a web link to an "unsubscribe" page, an email to an automated system, or an email to a staffer who remove them from the list) is up to the organization.

## Hiding participants from each other

When we send a batch of email invitations, it's important that the recipients don't see each other in the received message. Beyond the clutter of several hundred names in the "To" field, it's also a privacy violation – people shouldn't be able to see who else is on a email list.

To prevent this, we can either:

- **Use the Blind Carbon Copy (BCC) field** – If we're sending from a normal email account, we set the "To" field to ourselves and add the recipients to the BCC field. The BCC recipients are "CC'd" on the email, but the "blind" part means that they don't see anyone else on the BCC list.
- **Use a bulk-email service** – If we use a third-party email service (such as MailChimp or Mailerlite), it will give us the option of hiding recipients from each other.

## Who should send the email?

Because spam and phishing emails are a fact of Internet life, we need to **make sure that our email looks legitimate** to both the email system and the recipient themselves.

- The easiest way to do this is to make sure that the email is sent from an account officially belonging to the organization. If we're an employee of the organization, we can use our own email address, or we may prefer to set up a dedicated address for research purposes (e.g. *research@company.com*).
- If we're a consultant running the study on behalf of an organization, we should still send the invitation from an organization address rather than our own. People who use Acme Supply's products and services are more likely to believe (and respond to) an email from Acme than they are from Bob's Research Inc.
- Some recipients may contact our organization to see if the invitation is legitimate, so we should alert our support channels that we're doing a customer study - see [Alerting the organization about our study](#) in Chapter 8.

We can increase the response rate by having the invitation **sent by someone the user knows** (or knows of). When we had trouble recruiting enough people for a study with businesses, we asked the company's account managers to forward our email to their respective customers. Because the invitation was sent by someone they knew (and had a business relationship with), we got a much higher response rate.

---

**Next:** Using social media

## Using social media

- Who to target
  - What to say
- 

**One of the easiest ways to recruit participants is social media** – Facebook, Twitter, LinkedIn, and so on. Not only do we get to “ask” our followers if they can help us by taking part in our study, but they may also share the study with their own social networks, giving us much wider exposure than we can get directly.

### Who to target

If our organization already has social-media accounts, these are the natural candidates to start with.

First, **we must decide if a particular social network is appropriate for our study invitation**. Mostly this is a matter of deciding if our intended audience is likely to follow us on that network. For example, if we’re trying to recruit customers for the study, we may want to post on the organization’s Facebook and LinkedIn accounts, but not on the Yammer account that we use for internal company announcements.

If our organization has more than one account on a given social network, the same logic applies; we pick the accounts that target our intended audiences.

To reach more people, we may also want to use the personal social-media accounts of the project team (if they are willing, and if we think they can reach the intended audience). A request to help out with a study often gets a good response when it comes from someone the recipient knows personally.

### What to say

Posting on social media is like writing for a roadside billboard – it’s best to **keep it short and punchy**. This also helps our pitch fit the constraints of a medium like Twitter.

If we have already created a web ad and/or an email invitation, we can usually edit these down for our social-media pitch.

For example, if we already created this email invitation:

**Subject:**

*5-minute survey for Ministry of Silly Walks - win a \$200 gift card*

**Body:**

*Got 5 minutes? Want to win a \$200 gift card?*

*Help us design the **new Silly Walks website** by doing this quick online “scavenger hunt”:*

*<https://www.mosw.edu/study1234>*

**What’s this about?**

*We’re redesigning the Ministry of Silly Walks’ website to make things easier to find. We’re testing our site headings with real users, so make sure they work well.*

**etc.**

...then we could write our Facebook post like this:

*Got 5 minutes? Want to win a \$200 gift card?*

*Help us design the new Silly Walks website by doing this quick online “scavenger hunt”:*

*<https://www.mosw.edu/study1234>*

...and our Twitter post like this:

| *5-minute survey for the new Ministry of Silly Walks site - win a \$200 gift card!* [bit.ly/1gk4321](http://bit.ly/1gk4321)

Note that, in the Twitter post, we may need to replace our normal URL with a shortened version (using a service such as Bitly or TinyURL) to keep within the Twitter character limit.

---

**Next:** [Using commercial panels](#)

## Using commercial panels

- [How panels work](#)
  - [Quality of participants](#)
  - [Caveats for panels](#)
- 

Another source of participants are the many companies that run large **online research panels**.

We can use a commercial panel by itself, or as an addition to other recruitment channels (to help fill out our desired numbers).

### How panels work

Users sign up to these panels to earn rewards for doing online studies. When they sign up, the company collects all kinds of information about them – demographics, buying habits, hobbies, and so on. This creates a large database that can be queried for specific types of participants (e.g. women aged 40-60 who shop online at least once a month).

When we (as researchers) ask the company for participants, we can then specify which kind of people we're looking for.

While each research panel is different, the process typically goes like this:

- **We find an online panel we want to use**, based on what kind of users they offer in our region, how much they charge, and whether their participant system can work with our online testing tool. This usually involves studying their website and emailing them specific questions.
- **We set up an agreement with the panel company**, specifying how many users we want and how much we will pay for that number of responses.
- **We configure our study to receive participants from the panel**. Usually this involves receiving a "participant identifier" from each incoming participant, then notifying the panel when that participant has completed the study (so the participant can earn their reward from the panel company).

Some tree-testing tools let us configure this as part of our study. Others may require us to do a bit of extra work or ask the vendor to help us set it up, or may require us to ask each participant for their identifier so we can send it back to the panel company later.

- **We configure our study to return participants to the panel after the study**. This informs the research company that their participant has completed our study and can be rewarded. The company should supply us a URL to use.

### Quality of participants

In terms of quality of the results we get, research panels follow the same caveat as most other recruiting methods – we will always get a small percentage of participants who race through the test and give "dummy" answers, just so they can get the reward.

We haven't found these paid panels to be any worse than other methods in this regard (and, in fact, some of them actively cull members who don't give a decent effort, if notified about specific cases), but remember that regardless of the methods we use to recruit, we will still need to watch for garbage responses when we analyze the data later. (See [Cleaning the data](#) in Chapter 12.)

### Caveats for panels

While recruitment panels can work well and can save us a lot of time and effort, there are a few questions we should ask before using them:

- **Does the panel cover our region?**  
If we want participants from a particular province, state, or country, we need to check with the vendor to make sure they have adequate numbers in that region. International panels usually do a good job of covering North America and western Europe, but may be patchy elsewhere.
- **Is the panel likely to include the types of people we want?**  
Most commercial recruitment panels consist of consumers. This works well if we're targeting a portion of the general public, but it's harder to use consumer panels for other audiences like business people, farmers, government employees, and so on. We may need to find a more suitable source of participants - see [Other ways to recruit](#) later in this chapter.
- **Will we need to do additional screening of your own?**  
While the criteria we supply to the panel will net us a subset of their members, this may not be specific enough for our study. For example, we may have asked the panel for women aged 40-60 who shop online at least once a month, but we're really only interested in those who have returned at least one item that they bought online.  
Once we go beyond the criteria that the panel offers, we need to do additional screening – see [Screening for specific participants](#) later in this chapter.

---

**Next:** [Using integrated recruitment tools](#)

## Using integrated recruitment tools

---

If we're using an online tree-testing tool, it may provide its own recruitment tools that give us access to a participant pool that the vendor maintains itself, or which it has "hired" from a commercial research panel. (See [Using commercial panels](#) earlier in this chapter.)

Here's how integrated recruitment typically works:

1. We create our tree-test study using the online tool.
2. In the same tool, we choose their online-recruitment feature.
3. We select criteria that will get us the participants we want.

Here is an example from Treejack:

The screenshot shows a series of four dropdown menus for selecting recruitment criteria. The first menu is labeled 'United States of America' with a downward arrow. The second menu is labeled 'Education level' and has 'College/University' selected. The third menu is labeled 'Occupation status' and has 'Full-time Employee' selected. The fourth menu is labeled 'Gender' and has 'Any' selected.

4. The tool then gives us an estimated cost and time required:

The screenshot shows a box titled 'Estimated cost' with a green vertical bar on the left. The text inside the box reads: 'We could do this in 2-3 days for \$378 (USD)' and '\$7.56 (USD) per participant'.

5. If we decide to proceed, we launch our study normally, and the tool sends invitations to participants in its database who match the criteria we supplied.
6. When the desired number of participants have completed the study, the automated recruitment ends and we are notified that the test results are ready to view.

Integrated recruitment tools have similar pros and cons to [commercial research panels](#), but assuming they offer the kinds of participants we want, they are definitely easier to deal with because they are already integrated into the testing tool.

---

**Next:** [Other ways to recruit](#)

## Other ways to recruit

---

Sometimes we need to dig a bit deeper to get enough participants, or to get a more representative sample of our users. (See [Dealing with selection bias](#) later in this chapter.)

In either case, we can ask ourselves in any of these other methods would work for our particular study:

- **Amazon's Mechanical Turk**  
This service lets us offer micro-payments to people to do online tasks (in this case, our tree test). But because we're tapping into a global audience that is keen on earning a series of tiny fees quickly, we need to be extra careful about getting garbage data.
- **Trade associations and customer groups**  
Our customers may already have their own groups (online or not), or they may belong to a trade association (e.g. a farming collective or a plumber's union). We should consider asking permission to contact the group's members for our study, or ask for our invitation to be added to their next newsletter.
- **Universities and other post-secondary institutions**  
Students are often willing to participate in a study, both for the experience and for the chance of reward. Ask permission to post a online ad or paper bulletin.
- **Targeted publications and forums**  
There are message boards and online forums for every conceivable profession and interest. We can post on those that are frequented by the type of user we're looking for, but we should first find out if we need permission first to post our request.
- **Colleagues, friends, and family**  
Depending on the type of user we're looking for, we may be able to "shoulder-tap" people we know to participate. If we're looking for graphic designers, for example, we can start with a designer we know and ask if they can pass the invitation along to other designers.
- **SMS (TXT)**  
If our organization has a list of mobile phone numbers for customers, we can text our study request to these people, using the same short format we may have used for Twitter invitations (see [Using social media](#) earlier in this chapter). If they have a smartphone, they can activate the link directly from the TXT message.  
We rarely use this channel, however, because (a) it usually costs money to send these messages, (b) the limited length of a plain-text message is harder to work with, and (c) recipients are more likely to consider this to be spam than if we sent the same request as an email.

---

**Next:** [Dealing with selection bias](#)



## Dealing with selection bias

- [What is selection bias?](#)
  - [Which recruitment methods cause it?](#)
  - [How can we reduce bias?](#)
- 

Whenever we recruit participants, we are looking to get a representative sample of our actual (or desired) users.

We may even go to a fair bit of effort to get (or exclude) specific types of users, by querying customer databases, posting invitations to specific user forums, and so on.

In the end, though, **all recruiting is imperfect**; we'll miss some users we were hoping to get, and we'll get some that we were hoping to miss.

It is important, nonetheless, to **try to identify any selection bias in our recruiting**, so we can take that into account when we analyze our results, or when we do our next study.

### What is selection bias?

From Wikipedia's article:

***Selection bias** is the selection of individuals, groups or data for analysis in such a way that proper randomization is not achieved, thereby ensuring that the sample obtained is not representative of the population intended to be analyzed.*

In other words, certain recruitment methods may yield a **skewed selection** of participants, rather than the representative sample that we normally want.

For example, suppose that we only use a web ad on our site to recruit for our study. *Only people who visit our site in the next few days* (the duration of our recruitment) will see the ad. This means that:

- We are ignoring customers who don't use our website. (For many businesses, such as banks, this may be a big chunk of customers.)
- We are more likely to get people who visit our site frequently (say, several times a week), but less likely to get people who use our site once a month (e.g. to check their bill).

### Which recruitment methods cause it?

Here are some common causes of selection bias:

- **Web ads only get web users**  
If we only use web ads to recruit users, then (by definition) we're only getting those users who visit our website. While this is OK for many studies, it does ignore those customers who use other channels instead of the website. If we need offline users too, we'll need to find another way to recruit them.
- **Customer email lists only get existing customers**  
If we only use a customer list to email invitations, we are missing prospective customers (a potentially valuable audience) and ex-customers (who are often good sources of honest feedback).
- **~more sources of bias?**

### How can we reduce bias?

We may decide that a given selection bias is acceptable; in our example above, we may only want customers who visit our website, so this implicit selection actually serves as a useful screening mechanism.

However, it's important that **we consider what kind of selection bias each recruiting method adds to our study**. Then, we can either:

- **Try to reduce that bias**, and/or
- **Acknowledge the bias** and take it into account when analyzing our results and presenting the findings.

The most common way to reduce selection bias is to **use several different types of recruitment**. For example, instead of just running a web ad

(which only yields site visitors), we could use customer lists to reach those customers who don't use the website.

Another (generally less effective) way to reduce bias is to modify the single recruitment method we are using. If we only use a web ad, for example, we could post that ad on several different websites. We will still only get website visitors, but some of them will be people who have not visited *our* website.

While we will never eliminate all bias from our studies, these steps should help minimize it so we can be reasonably confident in our results.

---

**Next:** [Coordinating audiences and channels](#)

## Coordinating audiences and channels

Sometimes recruiting can get complicated. We may be targeting several different audiences, and we may be using several different media channels to reach each of them. Moreover, this work may be spread across several people on our team.

In cases like these, we've found it helpful to maintain an **audience/channel spreadsheet**:

A	B	C	D	E
method	primary school	secondary school	post-secondary	content to post
email, web news, newsletter	<del>National office sector distrib. list</del>	ABC news item	TEC	<a href="http://link-to-invitation-for-this-method">http://link-to-invitation-for-this-method</a>
	staff networks		regional offices (via staff networks)	
	ECAC	<del>Working Group</del>	Careers site	
		<del>previous volunteers</del>		
			NZQA	
			staff networks	
web ad	Ministry home			<a href="http://link-to-invitation-for-this-method">http://link-to-invitation-for-this-method</a>
	School holidays page			
	Starting Out	Learning Leaders	Teach.gov	
	ece.gov	ABC	Uni sites' news	
	Teacher's council	EDUCANZ		
	ERO	ERO		
	NZQA	NZQA		
	Little Learners	NZSTA		
	PTA	PTA		
Twitter	Ministry			<a href="http://link-to-invitation-for-this-method">http://link-to-invitation-for-this-method</a>
		HighSchoolLink	Uni Spotlight	
Facebook	Govt web community	Govt web community	Govt web community	same as Twitter
	Ministry	Ministry	Ministry	
<b>Legend</b>				
Dave				
Susan				
Chantelle				
done = strikethru				

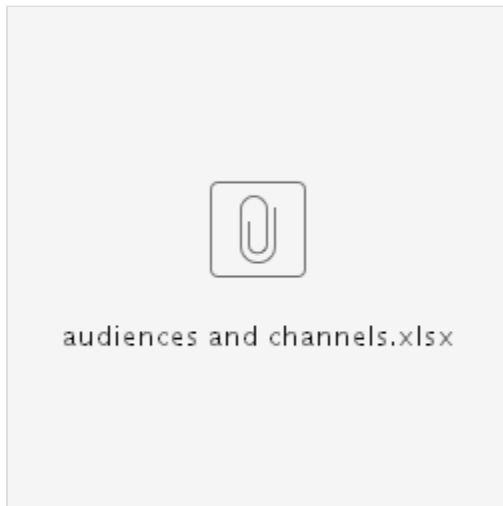
- The **audiences** are listed as columns across the top.
- The **channels** are listed as rows down the left side.
- For each cell (a specific channel for a specific audience), we record the **solution** we'll use.
  - For a web ad, this would be the specific website (or web page) to place the ad on.
  - For email, this would be the email list to send the invitation to.

- ...and so on for each medium.
- If we have several targets for a given medium (e.g. several websites for our web ad), we just insert extra rows as needed.
- We then color-code each cell to the person who will do it, with a legend at the bottom.
- As each person sends their recruiting invitation to that medium, they ~~strike off~~ the corresponding cell.

This makes it much easier to track the progress of our recruiting effort. We can see at a glance:

- Which audiences are using which channels
- Where the gaps are, in case we need to fill them
- Which invitations have not yet gone out.

Here's the Excel version - feel free to customize it as needed:



---

**Next:** [Screening for specific participants](#)

## Screening for specific participants

- Screening by filtering a database
  - Screening using separate email lists
  - Screening with explicit questions
- 

As mentioned earlier, we may want to control who gets to do our study (and who doesn't).

The easiest case is that we're happy to get results from anyone who visits the site (in the case of web ads) or anyone we've emailed (presuming that being on the email list is enough qualification in itself).

However, there will be times when we want to be picky – when we want a particular subset of users (or *don't* want them).

For example, if we're testing a site structure for a bank's online banking service, we probably want online-banking users. More specifically, we may want to *exclude* bank customers who have never used online banking and never intend to, since their answers would be irrelevant and could actually skew the results.

### Screening by filtering a database

If we're recruiting using a customer database, we may be able to filter the users down to those who have logged into online banking, say, once a month or more. That may be enough "screening" for us, so we can go ahead and run our study without more qualifying questions.

### Screening using separate email lists

If we have email lists that are already separated into groups that meet our specifications (e.g. business customers vs. consumers), then the job is easy – we invite people from the appropriate list to the corresponding study.

However, because email invitations can get forwarded (beyond our control), we may still want to include a screening survey question in the tree test, just to make sure we're getting the participants we intended.

### Screening with explicit questions

But what if we're using a web ad (which every site visitor sees) or we have a customer email list that doesn't include enough data to filter the users the way we want? In these cases, we'll need to explicitly ask each interested person if they meet our specifications.

There are several ways to do this screening:

- **Using descriptive links**

The easiest way to point specific types of participants to specific tests is to use a list of links that describe who they're for. We can use this list in both an email invitation and in a web ad's explanation page. This works well as long as the descriptions are clear and distinguishable from each other:

Help us design the new Acme website by doing one of these quick online "scavenger hunts":

**Personal banking - do this exercise:**

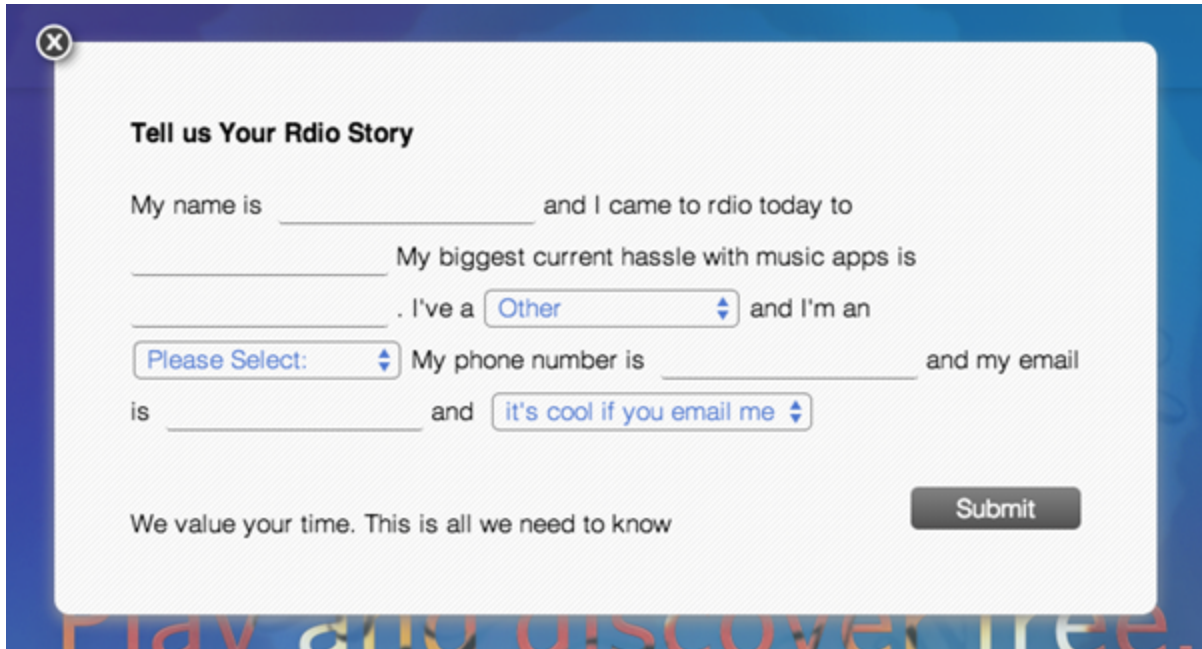
<https://your-testing-tool/study1>

**Business banking - do this exercise:**

<https://your-testing-tool/study2>

- **Using a screening tool**

We can also use a dedicated screening tool (such as [Ethnio](#)) to make sure we're getting the right participants. When users click our web ad, Facebook post, or Twitter tweet to participate, these tools pop up a window that asks the qualifying questions that we've provided. People who "pass" are directed through to our tree test, while the others are politely thanked and dismissed.



The image shows a survey form titled "Tell us Your Rdio Story" with a close button (X) in the top left corner. The form contains several input fields and dropdown menus. The text reads: "My name is \_\_\_\_\_ and I came to rdio today to \_\_\_\_\_ My biggest current hassle with music apps is \_\_\_\_\_ . I've a [Other] and I'm an [Please Select:] My phone number is \_\_\_\_\_ and my email is \_\_\_\_\_ and [it's cool if you email me]". At the bottom right, there is a "Submit" button. Below the form, the text "We value your time. This is all we need to know" is visible. The background of the form is white with a blue border, and the overall page has a blue background with the text "Play and Discover Tree." at the bottom.

- **Using the tree-testing tool**

If our tree-testing tool offers a screening feature, we could of course use that instead.

Even if it doesn't, as a last resort, we could ask a screening question during the tree test itself (as a survey question either before or after the tasks), and then only include the qualified participants in our results. However, that would be a waste of time, effort, and data for those who didn't meet our screening criteria, and we'll need to include them in any reward we're offering, because they did the study as we requested (even though we didn't use their results).

---

**Next:** Restricting access with a password

## Restricting access with a password

---

Normally we're happy to get participants from all the ads we post and invitations we send.

In some cases, however, we may want to keep a tree test confidential – for example, if we're testing the structure of a new site that we want to keep under wraps until it launches.

Some tools offer a way to password-protect a study. That is, the participant not only needs the link, but also a password that we supply. If someone stumbles over our study online, or is forwarded the email inadvertently, they still can't do the study unless they have the password.

If we're going to use a password, then, it's good practice to **separate the delivery of the link and the password**. If we are emailing invitations, for example, we send the password in a separate email blast, or ask recipients to reply so we can send them the password.

---

**Next:** [Writing a good invitation](#)

## Writing a good invitation

---

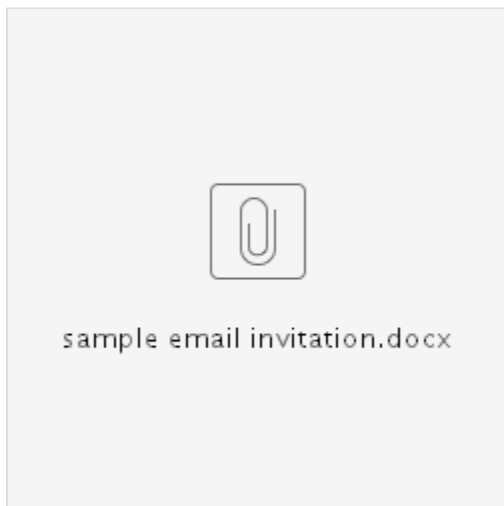
Earlier in this chapter, when we discussed web ads and creating an explanation page for them, we covered the basic information that participants are looking for when they're deciding if they will do our study – items including:

- What the study is for
- How long it will take
- What's in it for them

Because people are continually asked for their input, and also have to sift through spam emails, most have learned to sift these invitations ruthlessly. If we want their participation in your study, we must make our case clearly and concisely. This means that:

- **The invitation should be from a person or organization they know** (and preferably have a relationship with), as discussed in [Using email lists](#) earlier in this chapter.
- **The basic pitch should be clear at a glance.**  
If this is a web ad, the ad text needs to convey the basic proposition. If this is an email invitation, the subject line must perform this duty.
- **The person's most immediate questions must be answered by skimming the invitation.**  
Not only do we need to answer basic questions (How long will this take? What do I get in return?), but we need to do it in a way that a busy, only slightly interested person will absorb.
- **It should sound like something they're familiar with, so call it a "survey".**  
As we mentioned in [Using web ads](#) earlier in this chapter, the term "tree test" is unfamiliar, so we use the much more familiar "survey".
- **There must be a clear way to start the study.**  
This seems obvious, but we've seen lots of invitations that included so many other links (to more information, terms and conditions, privacy statements, etc.) that it took some effort to figure out which link was for the study itself.

Here's an example of an effective email invitation. Over time we've refined it to maximize the response rate:



Note that the subject line offers a clear and concrete proposition, the "start study" link is early and prominent, the most important questions are listed first, and the whole message is formatted to be easy to skim.

Also note that this content is very similar to what we put in our web-ad explanation page – see [Using web ads](#) earlier in this chapter.

---

**Next:** [Offering incentives](#)





## Offering incentives

- Prize draws
- Rewarding each participant

For most studies, we should offer an incentive.

One of the strengths of tree testing is that we can evaluate our proposed site structure *quickly*, fix it, and test it again until we get it right.

To get quick results, an incentive is almost always the way to go. We offer incentives in 90% of the studies we run because we're not willing to wait for results to dribble in.

We've encountered some client organizations (usually government agencies) that balked at offering a reward for a 5-minute online study, but after running a no-reward study and seeing the glacial pace of returns for themselves, most were very willing to pay for an incentive for the next study.

**We have only found a few situations where an incentive is *not* necessary:**

- **The participants are the organization's staff.**  
This is common for intranet studies, where management considers participation to be part of the employee's job. This can work if it's made clear to employees that they're expected to do the study. It helps if each manager makes sure their respective staff do it by a certain date.  
If this is unlikely to happen, then offering a modest incentive is still a good way to get results faster.
- **The organization's audience is very large.**  
If an organization has a huge user base, and we only need a few hundred responses, we can probably get that many in a reasonably short time, even without an incentive. But we may still need to offer incentives if we want to target certain specific segments of the organization's users.
- **The organization's audience is very engaged.**  
Many organizations *think* they have very loyal and dedicated users, but in our experience, only a few actually do. We can easily judge this by the response rates to previous studies they've done. If that rate is unusually high, we may try the study without an incentive and monitor how it goes. Otherwise, we encourage the organization to add an incentive; compared to the overall cost of the research, it's a small expense.

## Prize draws

For incentives, we usually set up a prize draw, because:

- **Draws are easy to set up.**  
We just need to decide on how much we want to spend and what our participants would desire at that price point.
- **They let us offer participants an enticing prize.**  
For a 5-minute tree test, offering each participant a small reward (say, \$5) is not usually feasible. And most people would rather have a chance at winning a big prize than the certainty of getting a very small reward.

For a standard study, we may offer a **prize worth anywhere from \$200 to \$500**, where the amount depends on:

- How much the organization offers for other online studies (such as customer surveys). Sometimes this is the organization's own products or services (which presumably costs them less than the value to the prize winner).
- How much is likely to motivate the desired audience to participate. For example, we would need to offer more to orthopedic surgeons than to university students.

To save money and reduce administrative effort, we may want to **combine prize draws across tests**. For example, if we're testing 3 alternative site structures (which means 3 tree tests), we can probably put all the entrants into the same prize pool. Offering a single \$300 prize is a more enticing incentive than offering \$100 to the winner of each test. We just need to make sure that the single prize is something that all of those participants would want.

Given a certain price point, we also need to **pick a prize that your audience desires**. A power company might offer \$300 off the winner's next bill, while a software company might offer a full version of their flagship program. If we're offering a gadget such as an iPad, pitching the latest version will get more interest from our audience.

Also, we should **choose a prize that appeals to all** (or at least most) of our participants. Being offered a free version of a program that you

already own is not much of an incentive. 😞

Having trouble thinking up a “custom” prize for the draw? Consider one of these generic prizes:

- **Gift cards**

These could be vouchers from a specific retailer (such as Amazon or Best Buy), but that may not suit some participants. We prefer to offer generic gift cards that can be used at a variety of retailers; they essentially act as prepaid debit cards that the participant can use at any store or online.

- **Tablets**

These are popular prizes because they're fun and useful without being as need-specific as mobile phones. We usually offer the winner a choice of an Apple iPad (or iPad Mini, if the budget is smaller) or equivalent-value Android tablet.

If we opt for a prize draw, we escape the need to reward each participant, but we do need to do a few extra administrative things:

- **State the terms and conditions of the draw**

For more on T&Cs, see [Writing supporting text](#) in Chapter 8.

- **Collect contact information for draw entrants**

We need to be able to contact the prize winner, so we'll need some kind of contact information from each participant who wants to enter the draw.

If our study requires an email address or participant identifier for other reasons, then we may not need to do anything extra.

In most of our studies, however, we make anonymity the default. For participants who want to enter the draw, we ask for their email address (and promise that it won't be used for anything else):

**Your email address (optional - only used for the prize draw or if you asked to be contacted about further research. No spam!)**

- **Pick the prize winner and notify them**

Once the tree test is closed, we download a list of the participants and pick a random winner. (We use an online random-number generator to do this.) We then notify the lucky winner and arrange to get the prize to them.

- **Publicize the prize winner** (for example, by including a picture and short blurb about them in the organization's public blog), assuming this was stated as a condition of the draw. This highlights how we're working to make things better for our customers, and provides a way to generate interest for our *next* study.

## Rewarding each participant

Because a tree test only takes 5-10 minutes to complete, paying each person is not usually feasible unless they are participating through an existing reward system, such as a [commercial research panel](#) or services like [Amazon's Mechanical Turk](#).

---

**Next:** [Recruiting for in-person sessions](#)

## Recruiting for in-person sessions

---

In [Asking “why?” with in-person sessions](#) in Chapter 11, we'll discuss the option of testing some participants in person.

For these sessions, the main recruiting challenge is getting the right participant in the right room at the right time. Mostly that's a matter of good recruiting, though life does happen – traffic is heavy, kids get sick, etc.

Here are a few tips that should help:

- When we book someone for a session, we **make it clear to them that it's a one-on-one study**. People are less likely to no-show for these than they are for group sessions.
- We tell them **how much time we will need** (20-30 minutes for the tree test and subsequent discussion, more if we're doing additional activities with them).
- We tell them that they'll be reading text (from cards or a computer screen), so they should **bring reading glasses if they need them**.
- We make sure we're clear about the time and place, of course, and we **send a follow-up email** with this info, directions and where to park, and our contact number.
- We **get their mobile number** so we can contact them if something changes on short notice.
- We **contact them the day before** the study to remind them of their upcoming participation. If they can't make it, we may be able to reschedule.
- When they arrive, we **make sure they know where to go**. Instructions in the email are good, but signage (or meeting them in the lobby) is better.
- If they haven't arrived by 5 minutes after the session's start time, we **call them on their mobile phone** to see if they're still coming and if they need help finding the venue.
- If we are visiting them, we find out where to park, and we **don't go alone** (for both our security and theirs). Oh, and we **take slippers** (in case we need to remove our shoes).

For more on planning and running in-person user research, we recommend the excellent and comprehensive [Handbook of Usability Testing](#) by Jeffrey Rubin and Dana Chisnell.

---

**Next:** [Chapter 9 - key points](#)

## Chapter 9 - key points

We aim for at least **50 participants per user group**. 100 is ideal. We'll need more if each participant is only shown a subset of the tasks.

If we use web ads to get participants, we choose web pages (and even other websites) that **target the kinds of users we're looking for**.

Web ads should be designed to **attract attention** and present an **at-a-glance proposition**, and should link to a page that **briefly explains the study** and provides an **obvious way to start**.

If we email invitations, we should **qualify the recipients** as well as we can, and **send our invitations in batches** to conserve the participant pool.

If we use social media, we should consider **which channels are appropriate**, and keep our invitation **short and punchy**.

If we use commercial research panels, we choose those that **adequately cover our region and the types of users we're looking for**.

To **boost numbers** and **reduce selection bias**, we use a variety of methods to recruit our participants.

We make sure we're getting the participants we want by **filtering our customer lists** and/or **screening** those who see the invitation.

Recipients are more likely to participate if they receive a **clear invitation from a sender they know**.

For most studies, **offer an incentive**. **Prize draws** are usually easiest.

---

**Next:** [Chapter 10 - Piloting the test](#)

## 10 - Piloting the test

*"All the right junk in all the right places" - Meghan Trainor*

OK, so we've prepared our tree, created a good list of tasks, set it all up in a testing tool, and figured out which users we're going to target.

We're ready to go, right?

**Ah, not quite. There's something wrong with our study.**

It's likely there's at least one glitch in the study that we haven't discovered yet. The problem is that we don't know *what's* wrong. At this point, we can either:

- Launch the study anyway and let our participants find the problem for us (and maybe muck up the data as a result), or
- Take an afternoon to **pilot the study, find the glitches, then launch** a slightly revised study that gives us higher-quality results.

Luckily, doing a test run of a study is simple, and the problems we'll find are usually easy to fix.

---

### Trying out the task wording

A pointer to Chapter 7

### Previewing a test

Trying it out in almost-real conditions

### Running a pilot test

Who should participate, online vs. in person, getting feedback

### Checking for technical problems

Dealing with spam blockers, mobile devices, old browsers, and firewalls

### Revising the test

Editing vs. duplicating, deleting pilot results

### Key points

---

**Next:** [Chapter 11 - Running the test](#)

## Trying out the task wording

---

When we created our tasks earlier, we recommended that “test-driving” the wording with a colleague.

Haven't done this yet? Do it now – see [Writing a good task](#) in Chapter 7.

---

**Next:** [Previewing a test](#)

## Previewing a test

---

Once the tree test is set up in an online tool, the first way to “test the test” is to **try it using ourselves as participants**.

How we do this depends on the tool we’re using:

- Some tools provide a **preview mode** where the person setting up the test can see try it out as if they were a participant. The difference is that no results are saved in this preview mode.
- If the tool doesn’t offer a way to preview the test, we may need to launch it as a real study, then try it ourselves before sending it to our real audience.  
If we find things that need revising, the tool should give us a way to make our fixes, either by editing the current test or by duplicating it and editing the new copy.

Despite how carefully we enter the tree and tasks and the other setup options, it’s not until we try the test as a participant that we see it all in the right order and in the right context, and **almost always we’ll instantly see things that aren’t quite right**.

- Maybe the tree has a few glitches because of missing indents.
- Maybe one of the tasks now strikes us as ambiguous.
- Maybe there’s a typo in the thank-you message.

In any case, it’s time well spent to **find it now rather than later**.

---

**Next:** [Running a pilot test](#)



## Running a pilot test

- [Who should participate](#)
  - [Running an in-person session](#)
  - [Getting feedback from participants](#)
- 

Previewing the test ourselves is a good first step, but we all know from trying to proofread our own content that **we really need someone else to spot our mistakes**.

That's why, in our studies, **we always run a pilot test with a small group of people before launching the real test**. Because they bring fresh eyes to the study, this initial group will find things that we missed – confusing task wording, typos, and so on.

Note that we want to **make this pilot as realistic as possible**, so we use the same invitation we normally would (but tweaked to say that this is a dry run and the incentive (if any) doesn't apply to it).

### Who should participate

There are 2 types of people who should pilot a tree test:

- **Project stakeholders**  
It's good for team members and sponsors to see what made it (and didn't make it) into the study, and this gives them a final chance to approve (or raise issues with) the tree and tasks.
- **Representative users**  
If possible, we try to include a few users in our pilot as well. These could be actual users, or surrogates such as customer-service staff or friends/family that resemble the target audience.  
Getting some real users means that we can double-check that our tree and tasks are written in language that they understand (not just the jargon of the organisation and industry).

### Running an in-person session

Even though most tree tests are run as online (remote) studies, **the absolute best way of piloting a test is in person**.

This is because the feedback is quicker and easier:

- The pilot participant can just talk aloud as they do the tree test – they don't have to take the time to write down their feedback.
- We (as the testers) can jot down the feedback that we think is useful, and can ask the participant to clarify their remarks when necessary. We can also spot problems that the participant doesn't even spot themselves (such as misunderstanding the meaning of a task).

However, in-person sessions do take more time, so it's usually not practical to do all pilot sessions this way. Typically we run 1 or 2 in-person pilot sessions, then get the rest of the feedback by emailing the pilot invitation to a wider pilot audience.

### Getting feedback from participants

When we "launch" the pilot test and invite these people to try it out, we make 2 things clear to them:

- **They should report anything wrong, missing, or confusing.**  
It's better to get too much feedback than too little, so we encourage pilot users to tell us about *anything* that could be improved. We may not make all the changes they suggest, but we don't want to miss something just because they think it's too minor to report. In most cases, it's easiest to get feedback by asking them to email us. If we have included a comment field as a post-test survey question, we also check that in case they decided to enter their feedback there.
- **They're not eligible for the study's reward.**  
The invitation and test may mention a payment or prize draw, but we make sure that they know this reward is only for the real participants in the real test.

---

**Next:** [Checking for technical problems](#)

## Checking for technical problems

---

Running a pilot also helps us spot technical problems with the study. By this, we mean issues that are caused by the supporting hardware (computers, networks, etc.) and software (operating systems, browsers, etc.), not by the test content.

Here are a few technical gotchas to be wary of:

- **Spam blockers (for email invitations)**

If we're emailing invitations, be careful that they are not triggering the spam blockers built into the recipients' email systems. While we can never be sure what will trigger a spam block, the fact that we're asking our invitees to click a link in the email (and are offering some kind of reward to participate) makes this something to check with popular email systems such as Gmail, Yahoo Mail, Outlook.com, and so on. We may want to set up accounts with the most popular email providers just so we can send to our own addresses as a spam test.

- **Computers vs. phones/tablets**

Most people may be doing the study on a conventional computer, but some will try doing it on a tablet or even on a smartphone. Make sure that the testing tool works on these devices. If it doesn't support a particular platform, we should mention that in the invitation (or web-ad explanation page).

- **Old (or odd) web browsers**

Most testing tools will work in any up-to-date browser that supports web standards (Chrome, Firefox, Internet Explorer, Edge, etc.). Old versions of browsers (Internet Explorer in particular) are still commonly found in large organizations, so if we're targeting these users, we need to check with the tool vendor to see if they support them (or, failing that, if the tool warns the user to try a newer browser). Mobile browsers seem to vary more than their desktop counterparts when it comes to handling online tests, so we try the test on a handful of the most popular browsers on iOS and Android.

- **Firewalls**

Most home firewalls (usually built into routers or anti-virus software) are unlikely to interfere with a tree test, but some corporate firewalls might. Certain large organizations have very strict firewalls that have (in our experience) played merry hell with our studies. If we're running a study that targets users in specific large organizations, and we get reports from those users that they can't do the study, we check with the tool vendor to see if they can help solve the problem. If they can't (and these problems can be hard to pinpoint), we may need to ask users in that organization to try the study from a different location (e.g. from home).

---

**Next:** [Revising the test](#)

## Revising the test

- [Editing the test and deleting pilot results](#)
  - [Duplicating and editing the new copy](#)
- 

Once we have piloted the test, we will likely have found some minor things to change, and perhaps even a few major ones. The great part of this is that we get to fix the test before our real audience sees those mistakes.

Depending on the testing tool, we may be able to edit the existing test, or we may need to duplicate it and edit the new copy.

### Editing the test and deleting pilot results

If the tool lets us edit the existing test, we can make our changes right there.

Most changes can be done without consequences, but we should be careful about two things:

- **Editing the tree can affect the task answers.**  
If we make structural changes to the tree, we should make sure that the task answers are still correct. For example, if we move a topic in the tree, and it was an answer to one of the tasks, we should update the task answer accordingly.

For more on revising the tree and your tasks, see [Chapter 14 - Revising and retesting](#).

- **Delete any pilot data you've collected.**  
If we plan to use the same test instance for the real study, we should delete the data collected during the pilot testing. However, if the data came from representative users and we didn't make any substantial revisions to the test after piloting, we can keep this data to join the "real" data we'll be collecting later.

### Duplicating and editing the new copy

Some tools don't let us edit a test that has been launched. So if we launched a test for purposing of piloting, we'll need to duplicate it first, then edit the newly created duplicate.

This has the advantage of keeping a revision history (of sorts) in case we want to go back to a previous version to check something or to grab an older version of a task's wording (for example).

Because we're revising (and eventually launching) the newly created duplicate, we don't have to worry about deleting pilot results; they'll be attached to the previous copy, not the new one.

---

**Next:** [Chapter 10 - key points](#)

## Chapter 10 - key points

**Our study will have glitches.** Piloting it before launch gives us a chance to fix them before they mess up the results.

**Test-driving tasks with a colleague** and “**previewing**” a **test ourselves** are quick and effective ways to find glitches.

**Running a realistic pilot test** with stakeholders and representative users is an even better way to spot mistakes in a study.

**Observing a few in-person sessions** is the best method for finding problems.

In addition to content problems, **watch for technical issues** caused by spam blockers, firewalls, unsupported web browsers, or mobile devices.

If we make changes after the pilot, **we revise a duplicated copy of the pilot study**, and recheck the correct answers as necessary.

---

**Next:** [Chapter 11 - Running the test](#)

## 11 - Running the test

*It's time to play the music*

*It's time to light the lights... – The Muppet Show*

It's time to get things started on the Muppet Show tonight with our tree test. We're ready to launch it and see what happens.

---

### Splitting users randomly among tests

Using separate links or code

### Launching the test(s)

A launch checklist, and timing the launch

### Monitoring the test's progress

Dealing with low turnout, weird scores, and high drop-out rates

### Keeping stakeholders informed

Sending periodic updates to the team

---

### Asking “why?” with in-person sessions

When we get results that need more probing

### Closing the test

A few housekeeping items to remember

### Key points

**Next:** Chapter 12 - Analyzing results

## Splitting users randomly among tests

- Splitting using mutually exclusive links
- Splitting automatically using code

---

e

As we saw in [Different tasks for different user groups](#) in Chapter 7, we may have separate tests set up for separate user groups. We can then use several methods from [Chapter 9 - Recruiting participants](#) to direct each participant to the right test.

But what if we have **several tests that all target the same users**? This is actually quite common, because we often want to test 2 or 3 alternative trees against each other to see which ideas work best.

There are two basic ways of splitting our participants randomly among tests – using **mutually exclusive links** and using **code**.

### Splitting using mutually exclusive links

The **simplest way to split participants randomly** is to set up the tests as a list of links, then construct some arbitrary way of getting participant to pick different links.

We usually do this by asking participants to choose the link that corresponds with the first letter of their first name. If we are running 3 tests, for example, the first link is A – H, the second is I – P, and the third is Q – Z:

Help us design the **new Acme website** by doing one (or more) of these quick online “scavenger hunts”:

*If your first name begins with A - H, do this exercise:*

<https://first-tree-test-url-here>

*If your first name begins with I - P, do this exercise:*

<https://second-tree-test-url-here>

*If your first name begins with Q - Z, do this exercise:*

<https://third-tree-test-url-here>

If we want a weighted split (e.g. 70/30) for some reason, we can adjust the letter ranges accordingly. Midway through the test run, if we notice that we're not getting enough people on test 3 (for example), we can expand its letter range to help it catch up.

Beyond being easy to set up, this method is also guaranteed to work in all browsers (for web-ad explanation pages) and all email clients (for email invitations), because it does not rely on JavaScript (see below).

### Splitting automatically using code

A **more elegant way** to split participants randomly is to use a bit of JavaScript code. We present a single link to all participants, but when the link is clicked, it triggers some JavaScript that picks a random number (e.g. 1, 2, or 3 if we're running 3 tests), then directs the user to the corresponding test address automatically.

Here's how it looks in a web ad's explanation page:

Help us design the **new Acme website** by doing this quick online “scavenger hunt”:

[Do the study now](#)

Here is an vanilla HTML page with the JavaScript code included. Feel free to customise this as needed:

```
<!DOCTYPE html>
<head>
  <script>
    <!--
      var links = new Array()

      links[0] = "http://google.com"
      links[1] = "http://bing.com"
      links[2] = "http://duckduckgo.com"

      function getRandomLink() {
        window.location = links[Math.floor(Math.random() * links.length)]
      }
    //-->
  </script>
</head>

<body>
  <a href="javascript:getRandomLink()">Do the study now</a>
</body>
</html>
```

This works well in an explanation page on the website (that the web ad would direct to), but it may not work reliably in email invitations because some email clients may block JavaScript in incoming messages.

As a workaround, we could make the email invitation go to a web page on our site that shows a "One moment please..." message, does the random-test-by-code selection, then automatically redirects to the corresponding test address.

---

**Next:** [Launching the test\(s\)](#)

## Launching the test(s)

- [A launch checklist](#)
  - [Timing the launch](#)
- 

Once all the pieces are in place, we just need to kick it all into gear, so to speak.

### A launch checklist

We typically use a procedure like this to launch a tree test:

1. Launch the test(s) in the online testing tool.
2. Put the real test addresses (URLs) into the invitations (web, email, etc.).
3. Post any web pages needed (explanation pages, terms and conditions, etc.).
4. Send invitations (email, social media, etc.).
5. Notify support channels that the study is now running, with an estimated duration.

### Timing the launch

Does it matter which day of the week (or month) we launch? Are Mondays better than Fridays, for example?

For studies with the general public, timing is not usually an issue. People visit websites and read their email and social-media feeds throughout the week, so any day is OK to launch a study.

If we're targeting businesspeople, then early in the week is usually better than later, because fewer people will visit business-user sites or read business email over a weekend.

The main thing to watch for is **periods where the target audience is especially likely (or unlikely) to respond to an invitation**. For example, don't count on getting many farmers to participate during harvest, or university faculty during winter break.

It's also a good idea to launch when we have time to deal with any problems that may crop up. Tree tests aren't usually prone to operational problems, but it's best to be available anyway.

---

**Next:** [Monitoring the test's progress](#)



## Monitoring the test's progress

- Not enough responses
  - Missing user groups
  - Unbalanced numbers between tests
  - Low success rates at first
  - Very high task scores
  - High drop-out rates
- 

While a tree test is running, it's good practice to log in every day or two and see how things are coming along.

We all hope that our studies attract lots of participants, of each type we ask for, and that the results show how much better our new tree is than the old one.

While that does sometimes happen, more often we find that things are not *quite* running to plan. Here are the problems we encounter most when we check the progress of a study.

### Not enough responses

**Low participant numbers are the most common worry for any online study.**

If we're halfway through the test period and we only have a quarter of the participants we hoped for, we may need to work harder to find people.

- **Email:** If we only sent an initial batch of email invitations, we send another batch.
- **Web ads:** We re-check that our ad is very visible and presents a concise, attractive proposition. We also consider putting it on more web pages and (if possible) on more websites to get more views.
- **Social media:** We often repeat our invitation on our social networks halfway through the testing period.
- **Incentive:** If we suspect that our incentive is not big enough (and we've done everything else we can to boost our responses), consider increasing it. If management reduced our planned incentive because they thought it was excessive, we may want to revisit that decision with them (with the data in hand).

### Missing user groups

When we target several user groups in a single tree test, sometimes we get lots of people from group A and B, but hardly any from group C. If we included a survey question that identified the participant's user group, we can check that now to see if any groups are lagging and need more recruiting effort.

Obviously, the best way to boost a certain group's numbers is to invite more of that group.

- If we have **group-specific email batches** that we haven't sent yet, that's the easiest thing to do.
- If we can **place an ad on websites that this group frequents**, that should also help boost our numbers.
- If this group is likely to have **some kind of organization that they belong to** (a trade association, meet-up group, special-interest forum, etc.), we may want to approach the organization's administrator and ask for help.

### Unbalanced numbers between tests

Earlier we talked about splitting users randomly among tests. Usually this is an even split (e.g. two tests would each have a 50% chance of being selected), but sometimes we find that, halfway through the test period, test A has two thirds of the responses for some reason.

Whether we used code or a set of arbitrarily split links, we can change this partway through the test to even up the numbers.

- If we used a set of links, we can change the split from something like "[first name A-M](#), [first name N-Z](#)" to "[first name A-E](#), [first name F-Z](#)" so that the first test now gets 20% of the clicks, while the second test (the one that's lagging) gets 80%.
- If we used code, we can change it from a 50/50 split to 80/20 in favor of the under-supplied group.

### Low success rates at first

Besides the number of participants, the other big thing we're sure to check is the scoring – how well our tree is performing overall, and how

individual tasks are doing.

**Very often, we'll be surprised (and appalled) by how low the interim scores are.** Some part of the low scores will be justified – especially in a first-round test of a new tree, parts of that tree will simply not work well for participants. Testing simply lets us identify the parts that need rethinking.

However, we also find that interim scores are often lower than expected because:

- **Some tasks may be confusing or misleading.**

This is especially likely if we didn't properly pilot our test. Some tasks are hard to phrase clearly without giving away the answer, but remember that a confusing task is a problem in the study, not necessarily a problem in the tree itself. We shouldn't change the wording during the test, but we should revise in our next round of testing.

- **Some correct answers aren't marked as "correct".**

After doing hundreds of tree tests, we still run into this wrinkle all the time. When we set up each task, we try to mark all the correct answers for it. However, in a large tree, each task may have several correct answers, and it's likely we'll miss a few. Because of this, a good testing tool should let us (as test administrators) change which answers are correct for each task, either while the test is running or afterward when we're doing the analysis. We often find that test scores go up substantially when we do this post-test correction. For more, see [Cleaning the data](#) in Chapter 12.

## Very high task scores

Ideally, a high task score means that we did our job well when we created the tree.

Unfortunately, it can also mean that we included a "giveaway" word (and didn't spot it during piloting). If we did, then this isn't a fair measure of the real-word effectiveness of the tree.

Again, we shouldn't edit the task's wording while the test is running (unless we spot it *very* early); fix it in the next round.

## High drop-out rates

We may find that lots of participants start our study, but many drop out before they finish it.

We'll always have some drop-off (it's the nature of online studies), but if it exceeds about 25%, we should investigate.

- **At the explanation page**

If our web ads or email invitations link to an explanation page, we can use web analytics to compare how many people visit that page to how many actually start the tree test itself. A large drop-off here indicates that the explanation page is either confusing, hard to scan, or the "start" link is not obvious. (Is it prominent and above the fold?)

- **During the test itself**

If a person makes it to the tree test itself, try to find out where they drop out. (Unfortunately, most testing tools do a poor job of helping us in this regard.)

If it's during the welcome/instructions stage, they may be finding these pages confusing, too long, or simply not what they expected. We can check this by trying the test with a few people in person to see where the problem lies.

If they drop out during the tasks, it could be caused by:

- Having too many tasks (seeing "1 of 26" is daunting)
- Presenting tasks that are confusing ("Forget this, it's just too hard")
- Or simply because this is the first time they've done a tree test and they're not sure what to do. (Better instructions may help, but some people will leave no matter how well we explain it.)

---

**Next:** [Keeping stakeholders informed](#)

## Keeping stakeholders informed

---

If our test takes more than a few days, it's a good idea to provide periodic updates to our project team and other interested parties.

Typically we do this by **emailing a brief summary** to them that shows how our responses are tracking, early success rates (if we have enough participants to judge by), and any actions we need to take (e.g. trying harder to recruit a particular user group with low numbers so far).

Here's an example of an email interim update:

*Hi all, here's a quick update on the tree tests that we're currently running:*

**Tree 1** (organized by audience)

Existing customers	42	
Non-customers	7	<b>Low response rate!</b>
Success rate	55%	<i>A few low-scoring tasks bringing down the rest</i>

**Tree 2** (organized by activity)

Existing customers	36	
Non-customers	32	
Success rate	72%	<b>Good overall performance so far</b>

Actions:

- **We urgently need more non-customers for Tree 1.**  
*Susan and I will work on this today, but more ideas are always welcome.*

*Thanks!*

---

**Next:** Asking "why?" with in-person sessions

## Asking “why?” with in-person sessions

- [Running an in-person session](#)
  - [Recording results](#)
- 

The big advantage of using online testing tools is that once we launch a study, it runs by itself; people can participate any time (and any place) they want, and we don't have to be there to moderate the session.

The flip side of this, of course, is that **we can't ask remote participants why they made certain choices** or what specifically confused them in a certain task.

In many (if not most) tree tests that we have run, it was not hard to figure out why certain tasks scored poorly and certain parts of the tree did not work well. There are cases, though, where we've looked at some very low scores, inspected the tasks in question, gone back and studied the tree, and we *still* weren't sure why participants were getting those tasks wrong.

### Running an in-person session

In these cases, we've found it very helpful to follow up the remote study with an identical study using moderated sessions. These can be done in person, using a screen-sharing/audio app (e.g. Skype or Google Hangouts), or even a simple phone call while the participant is in front of their computer.

If the tree test normally takes 5 minutes, we schedule a 15-minute call to allow for testing and discussion.

**Each in-person session is run much like a standard usability test:**

- We ask the participant to do the tree test (using the testing tool) and encourage them to “think aloud” as they go.
- If they struggle to find a correct answer, we ask neutral questions to find out why they're having trouble (“Tell me what you're thinking now...tell me more about that...” and so forth).
- While not every participant will have trouble with the tasks we're concentrating on, we hope to get a few who do, so we can find the cause of the problem.

For more handy tips on testing in-person, see [Nick Bowmast's short article](#).

### Recording results

Another way to see why participants are making puzzling choices is to employ a **remote usability-testing tool** that records a screencast of the user's session. ([UserTesting.com](#) and [TryMyUI.com](#) are popular examples.)

Instead of using this type of tool to test a website, we give them the web address of the tree test, and we get back screen recordings of users doing the study. Because we can instruct them to “think aloud”, we'll be able to hear their reasoning (and possible confusion or misunderstanding) as they click through the tasks.

See [this UserTesting.com blog post](#) for an example of how recording remote sessions helped the author understand what needed fixing in their tree test.

---

**Next:** [Closing the test](#)

## Closing the test

---

Eventually, we will finish accepting participants and collecting data. This may be when we get enough participants, run out of time, or (as often happens) a combination of both.

We may close the test manually, or we may have configured the testing tool to close it automatically according to a rule. See [Setting closing rules](#) in Chapter 8 for more on on these options.

Once the test is closed, there are a few housekeeping items to attend to:

- We take down our web ads on all the sites we posted them to.
- If we posted an explanation page and linked to it from other channels such as email or social media, we replace its text with a simple “Thank you but this study is closed” message.
- If anyone who received an email invitation belatedly clicks the test link, the testing tool should display its own “study closed” message (but best to try it yourself to make sure).
- If we arranged a prize draw, we pick the lucky winner(s) and notify them. For more on this, see [Offering incentives](#) in Chapter 9.

Finally, after all our work setting up the study, and the participants each doing their best to find the right answers in our tree, we can now sit down, look at the data, and see how well the tree performed. The next chapter explains how to make sense of those results.

---

**Next:** [Chapter 11 - key points](#)

## Chapter 11 - key points

If we're running more than one test, and need to **split participants** between them, we can provide mutually exclusive links or use code.

When **timing our launch**, we should consider our audiences' availability.

**Monitor the study every few days** to see if there are any obvious problems, such as low or unbalanced response rates, lower- or higher-than-expected success rates, or high drop-out rates.

**Keep stakeholders updated** on the progress of the study, particularly if there are problems that they can help fix.

If we encounter low scores that we can't explain, we should consider **running a few in-person tree tests** so we can probe the participants' behaviour and thinking.

Once we close the test, **don't forget the housekeeping** (taking down web ads, doing the prize draw, and so on).

---

**Next:** [Chapter 12 - Analyzing results](#)

## 12 - Analyzing results

*"How many roads must a man walk down, before you can call him a man?" - Bob Dylan*

This is what we've been waiting for - **actual results from actual users**. Now, we just need to figure out what the data means.

Luckily, the basic results from tree tests are usually easy to understand - easier, say, than a typical open card sort. And if we're using [tree-testing software](#), the results have already been summarized and visualized, ready for us to discover how well our tree(s) performed.

---

### Cleaning the data

Backing up raw data, removing garbage, updating correct answers

### Sharing the data

Publishing via the tool, sharing logins, uploading results to the cloud

### Reviewing overall results

Overall success rate, directness, speed, and overall score

### Analyzing by task

Where they went, clicked first, backtracked, slowed down, gave up, etc.

### Analyzing by branches

Looking for clusters of right and wrong answers

### Analyzing by user group or other criteria

Filtering results using survey questions, and comparing them

### Discovering evil attractors

What they are, how to spot, why they happen, how to get rid of them

### Key points

---

**Next:** [Chapter 13 - Communicating results](#)

## Cleaning the data

- Safeguarding the original data
- Removing garbage sessions
- Updating correct answers
- Recalculating the results

The great thing about online testing tools is that, not only are the results compiled and partially analyzed for us, they are usually available instantly, even while the test is still “live”.

Very few people will wait until the test is done before peeking at the results. We generally take a look once we've hit 20 or 30 respondents, just to see how things are going.

**Often, they're going a lot worse than we expected.** For example, we may have been hoping for a success rate in the 60s or 70s, but the early results often come in at 40 or even less.

For example, we ran three tree tests for an electricity company. We expected a low score for the existing site tree, but we also got low scores for the new trees we had designed:

	Existing tree	New tree 1	New tree 2
Uncorrected score	36%	40%	35%

While this was lower than we expected across the board, we told ourselves not to panic. From experience, we knew that **the original data needs cleaning up before it accurately shows what happened.**

There are a few common reasons for this:

- Participants may have **misinterpreted** some of our tasks, leading to a lot of wrong answers. There's not much we can do about this once the results start coming in, other than to take those tasks' results with a grain of salt, and to fix those tasks in later tests.
- Some participants may have given **garbage answers**, for a variety of reasons. We can remove some or all of these after the fact.
- It's likely that participants discovered **some correct answers that we missed**. We can fix these after the fact too.

## Safeguarding the original data

**The first thing to do, before we touch the data at all, is to back it up.** We always want to have a copy of the original data in case anything goes wrong with the tool or our data clean-up.

For online tools, the most common way to preserve our data is to download it as a spreadsheet file (XLS, CSV, etc.).

Download the entire data set and save it in a secure place. Explicitly name it as the original data, then never touch that file again. It's not for later analysis – it's simply a backup of the raw data in case Murphy's Law strikes and we need something to go back to.

This is also useful if the tool we're using goes offline, either temporarily or (eventually) permanently. In the unlikely case that we need to go back and look at the raw numbers a year or two from now, we'll want to have our own offline copy of the data.

## Removing garbage sessions

**Most online studies collect some garbage data.**

This is pretty much a given in the land of online research. Most participants make an honest effort, but a few don't.

This is especially true when the study is open to the public and there's an incentive involved. We get more respondents, but some of them are just there for the prize, and will zoom through the study as quickly as possible to get to the pay-off.

- **Sometimes this means lower-quality data.** They did the task in a rush, so their decisions were not as considered as they might be in real life. On the other hand, many people “rush” through their normal web browsing, so it's hard to quantify this effect.
- **Sometimes this means garbage data.**



They clicked randomly, or chose the same option each time, just to get through the test quickly.

We can normally weed out the latter, and reduce some of the former.

## Going too fast

The first thing we look at is sessions that were done too quickly. Most tools track the total time taken for each participant, and this becomes a good way to weed out garbage data.

There's no hard-and-fast rule about what "too quickly" means, but if most respondents did our study in 8 minutes, and we find someone who did it in less than 2, that's what we're getting at.

After sifting through a lot of data, our general rule of thumb is: **be skeptical of sessions that were done in less than half the average time.**

So, in our example above, if the average session was 8 minutes, we would look at sessions that were done in less than 4 minutes.

So, what do we do with a suspect session?

If the session was done *really fast* (say, a quarter of the average time or less), we delete that session out of hand. It's extremely unlikely that the participant could have performed the tasks with any thought at that speed.

For the remaining sessions (those approaching "half time", or whatever threshold we're comfortable with), we review the data. This means **looking at the click paths of each task** for that participant, to see if there are any clear indications they were intentionally speeding through our tests. The most common indications are

- **Choosing the same item at each level** (often the first or last item)
- **Going down the same path for every task**
- **Choosing nonsense paths for every task**  
Careful with this one, because what we consider a "nonsense path" might have made sense to them. Only suspect those who do this for a large number of tasks.

If we find a session with a lot of this kind of garbage, we delete it, and if we are doing a prize draw for this study, we remove that participant from the draw. This is not a behavior to encourage.

Some tools let us "exclude" sessions from the analysis. This is what we think of as a "soft" deletion; the session is removed from the analysis, but it's not actually deleted, so we can get it back later if we change our minds. If our tool offers exclusions, we recommend using them to clean up the data instead of actually deleting the data outright.

## Skipping too many tasks

Another way that some participants speed through a test is by skipping tasks.

Again, we rely on a rule of thumb to expedite our clean-up: **be skeptical of sessions where the participant skipped more than half the tasks.**

If we find participants who skipped much more (say 70% or more), we delete those sessions without further review.

For the remaining few, we review their click paths (like we did above for those who went too fast), look for the same indicators, and delete/exclude those that look guilty.

## Being wrong way too often

The final check we do is for participants who got every task wrong (or something close to that). This is a clue that they may not have made an honest effort.

The low-achiever rule of thumb is: **be skeptical of participants who got 75% (or more) of the tasks wrong.**

Again, we review their click paths as described above, look for the same indicators, and mete out our justice accordingly.

## Updating correct answers

One more way that we “clean up” the data is **changing the correct answers** for our tasks.

This happens later in our analysis, when we’re examining where participants went for each task. Sometimes we’ll find that they have “discovered” answers that we should have marked correct at the outset. For more on this, see [Where they went](#) later in this chapter.

When we find additional correct answers (and this is surprisingly common), we need to go back and mark those new answers as correct, then make sure that the tool recalculates the results accordingly.

When we find pages that seem like very reasonable places to go for a given task, we need to make sure those pages actually help users who are performing that task in real life. We can either:

- **Include content** on that page that satisfies the given task, or
- **Include a prominent cross-link** to a page that does satisfy the task.

## Recalculating the results

If we do alter the results (either by excluding/deleting sessions, or by adjusting correct answers), **we need to make sure the scores are recalculated accordingly**.

Depending on the tool we use, this may be done automatically, or we may need to trigger it manually.

And we should remember to download another local copy of the revised results, for safekeeping.

For the power-company study we described above, when we recalculated the scores after adding some missed correct answers, our results changed substantially (although they were still lower than we would have liked):

	Existing tree	New tree 1	New tree 2
Uncorrected score	36%	40%	35%
Corrected score	46%	43%	47%

---

**Next:** [Sharing the data](#)

## Sharing the data

---

If we're tree testing for a project team, or for a client, we may want to share the raw results with them. By "raw results", we mean the task scores and click paths that tree-testing tools track for us.

This is a great way of getting others interested in the study, while we take some time to dig deeper into the results and pull out some patterns and findings.

Depending on the tool, there may be several ways to share the data:

- We can **use the tool's "publish results" feature**, if it has one. This typically gives us a web address that we can distribute to others; clicking the link shows them a read-only view of the results, without requiring a login.
- We can **share our username/password** for the tool, so others can log in and view the results for themselves. While this is easy, sharing a login is not ideal, especially if we use that account for other projects.
- We can **download the results** (typically to a spreadsheet file) and distribute that to others. However, the spreadsheet may not include the graphic visualizations that the tool offers in its web form.
- For collaborative analysis, we may want to download the spreadsheet and then **upload it to a collaboration system** (such as Google Drive or Office 365) that allows several users to view and edit the same file at once. This is helpful for splitting up work and viewing others' feedback.

---

**Next:** [Reviewing overall results](#)

## Reviewing overall results

- Overall success rate
- Overall directness (backtracking)
- Overall speed (time taken)
- A “total” score

The first thing we want to know is **how the tree did in general**:

- Did it perform well, and all we have to do is tweak a few things?
- Did it perform just OK, and some parts need reorganizing?
- Did it perform poorly, and we need to discard it and look at other options?

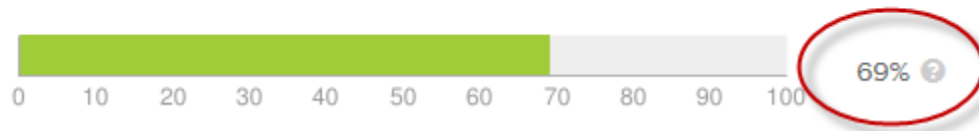
### Overall success rate

Not surprisingly, the most important thing to look at is success rate – **how many participants chose the correct answer, across all tasks?**

Most tools will give us this as a rating out of 10 or 100. For example, a score of 69 means that 69% of the time, participants chose a correct answer:

Overall

Success



Once we see a tree’s overall success rate, the natural question is “Well, is that good, bad, or just average?”

As any consultant will tell you, *it depends*. Mainly, it depends on two things:

- **Size of the tree** – All other things being equal, it’s harder to find things in a larger tree (or haystack, as the saying goes).
- **Complexity for the intended user** – If the topics and subtopics in the tree are challenging for participants to understand, they’re going to have a tougher time finding the right answers.

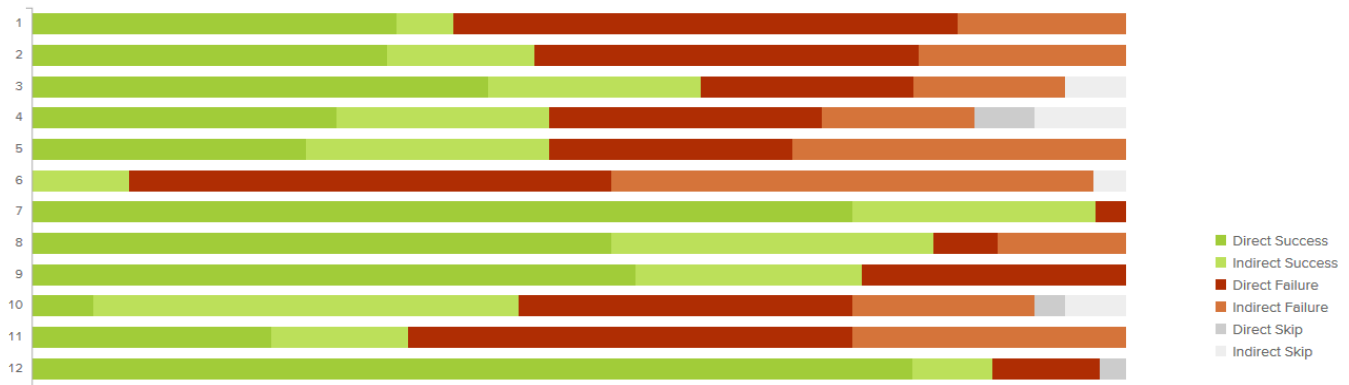
But we do need to start from somewhere. In our experience, over hundreds of tree tests, the following rough markers have emerged for trees of average size and complexity:

- **0-50 – The tree needs to be completely rethought or discarded.**  
Trying to tweak it will only bring it up to “mediocre”.
- **50-65 – The tree needs substantial revisions.**  
If our analysis reveals specific problems (and it should), and we think we can fix them, we should be able to revise this tree to perform well.
- **65+ - The tree is effective, but may need minor revisions.**  
Our participants are finding the correct answer at least two-thirds of the time, so the tree is doing its job well, and only needs tweaking.

A high score doesn’t mean “no revisions needed”. We’ve never run a tree test where everything worked so well that we couldn’t improve it a bit more. There are always a few lower-scoring tasks that suggest further improvements.

What the overall success rate doesn’t tell us is how much the success rate of the individual tasks varied. For example, a 60% overall score may

mean that all tasks hovered around 60%, or that half our tasks were 90% and half were 30%. To find out, we need a breakdown by task, which some tools summarize in a graph like this:



In this example, we can see that a few tasks had very low success rates, and two were very high. To find out more, we need to drill down to the task level - see [Task success rate](#) later in this chapter.

## Comparing tree-test scores to usability-test scores

People are often surprised that we consider 65+ to be a "good" score. Shouldn't we set the bar at 80 or 90?

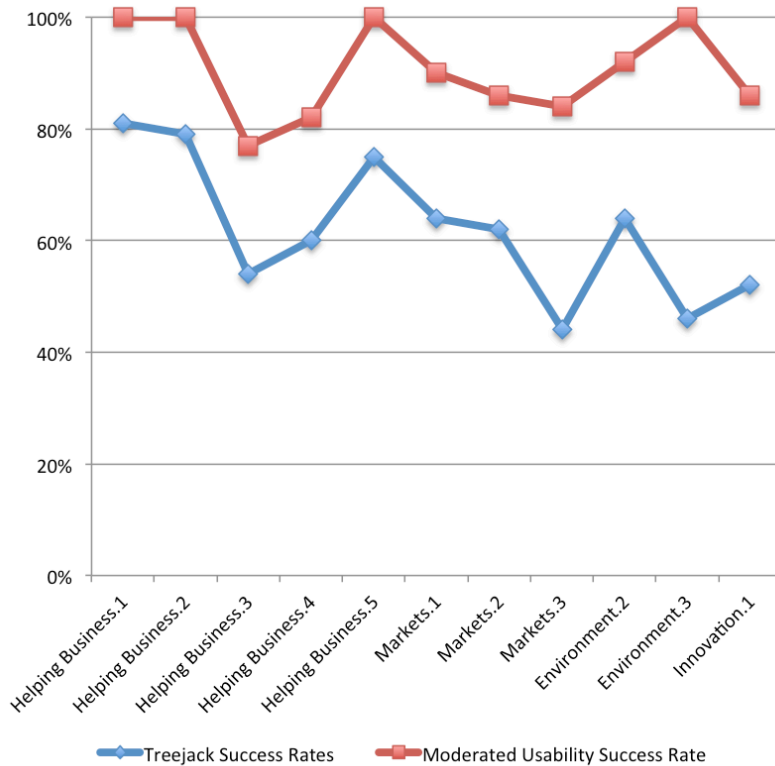
Effective trees don't usually score higher than 80 because **we're testing a top-down text-only tree with no other aids**. Our participants are making choices without the benefit of:

- **other navigation aids** such as see-also links, featured links, and multi-level browsing (e.g. mega menus)
- **visual design** - chunking of links/content, and emphasis on more important links/content
- **content** that explains headings using decoration text, hover text, etc.

Once we refine our text tree to be effective (i.e. perform well in tree testing), we should then be able to **add these other design elements to further improve the findability of items in our website**.

In our experience, success rates from the final website tend to be 20-30% higher than the scores we see in tree testing.

Lisa Fast at [Neo Insight](#) has written an interesting article [comparing tree-test scores to usability-test scores](#). Here's the graph of how they related in her study:

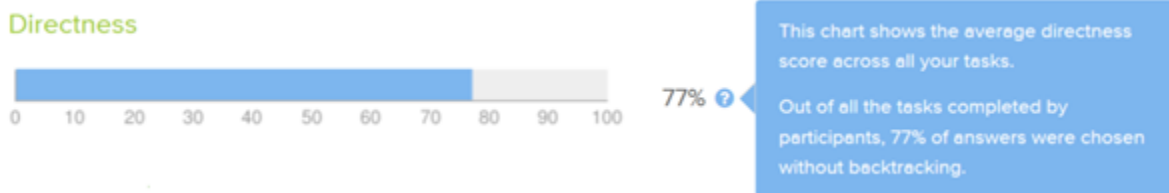


Lisa found that not only were the two scores correlated, but **the usability-test scores were 29% higher** (on average) than the tree-test scores.

Finally, we should warn that adding other aids is **no guarantee** of improvement. A poor visual design, clumsy navigation, and confused content can actually make a website perform worse in usability testing than it did in tree testing. A single method can only go so far. 😊

### Overall directness (backtracking)

To get a general idea of the effectiveness of our tree, it also helps to look at **how directly our participants found the right answer**. Did they go straight there, or did they have to try a few different paths first?



How this is scored depends on the tool we're using:

- **Some tools treat directness as a simple yes/no measure** – did the participant backtrack at all during a task? This method doesn't care if they backtracked 1 time or 5 times during the task – it's either yes or no.

**In our experience, 70% is an average score for this method.** Less than that indicates that users are having trouble finding the right path.

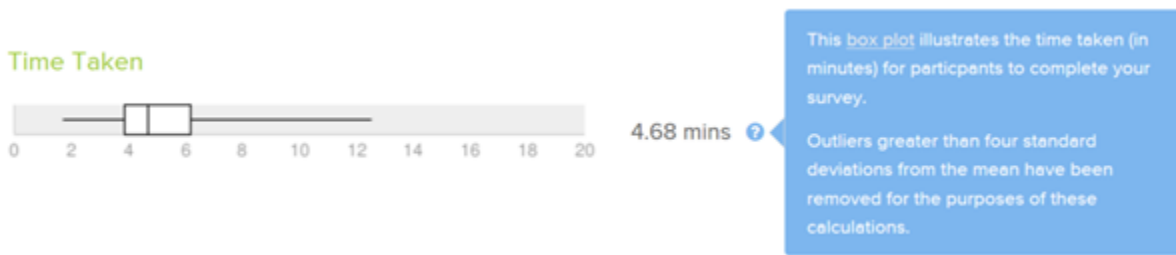
- Some tools try to quantify how much wandering a participant did. A single back step lowers their score a bit, but repeated meanderings through the tree lowers it much more.

~guidelines for scores using this method?

While the overall directness score gives us a rough idea of how clear and distinguishable our headings are, we'll need to drill down to specific tasks to determine where the most backtracking happens. For more on this, see [Directness – where they backtracked](#) later in this chapter.

## Overall speed (time taken)

Most tree-testing tools show us the average (or median) time taken by our participants to complete the tree test.



## Comparing times between trees

If we're testing several trees against each other, and the trees are approximately the same size (in breadth and depth), **we can compare these overall times to see if some trees are "slower" than others.** This suggests that participants either had to:

- think a bit longer between clicks, and/or
- click more times to get to their answer

This is a very rough measure, however, and to make sense of it, we'll need to drill down to see which tasks (or specific areas of the tree) are slowing down our participants. For more on this, see [Task speed - where they slowed down](#) later in this chapter.

## Keeping the study brief

A more practical use for the average time taken is making sure that our tree test is not taking too much of the participants' time.

In general, **we recommend an overall duration of 5 minutes for a tree test.** This is typically how long it takes the average participant to do 8-10 tasks (our recommended amount) for a medium-size tree (200-500 items).

If we have a larger tree, our test time may exceed this, but we still recommend keeping it under 10 minutes to avoid participant fatigue and boredom.

If the average duration is longer than this because we are asking each participant to do a lot of tasks (say, 12 or more), we are likewise inviting participant fatigue and boredom. More importantly, our results may be skewed by the "learning effect" – see [How many tasks?](#) in Chapter 7.

## A "total" score

Some tools present a single overall score, combining several measures: success rate; directness; speed; and so on. This overall score typically uses some kind of weighting, with success rate usually being the biggest factor.

This is useful when testing trees, because it makes us consider more than just the success rate itself. **If people can find items in our tree, but they have to do a lot of backtracking, or they have to ponder each click, there's something wrong and the score should reflect that.**

Note that the various online tools differ in how they calculate their overall score, making it harder to compare scores between tools:

- Treejack calculates its overall score as a weighted average of success rate and directness (at a 3:1 ratio), but does not include speed in its calculations. *~no longer provides a total score?*
- UserZoom?

---

**Next:** [Analyzing by task](#)

## Analyzing by task

The whole point of running a tree test is to find out which parts of the tree work well and (more urgently) which parts don't, for the most common and critical activities that our users do on the site.

We largely determine this by:

- examining the results of **individual tasks**
- looking for **patterns among tasks**

---

### Task success rate

The most important metric for a task

### Where they went

Inspecting high- and low-success tasks

### What they clicked first

Did participants agree on where to start, or did they scatter?

### Directness – where they backtracked

Directness scoring, backing up from incorrect vs. correct paths

### Task speed - where they slowed down

Task times vs. click times, and why they took longer

### Where they gave up

Looking for patterns in skipped tasks

### Confidence

Were they sure of their answer, or doubtful?

### Dealing with outliers

When to ignore strange results from a few participants

### Finding patterns among tasks

The most common patterns to look for

---

**Next:** [Analyzing by branches](#)



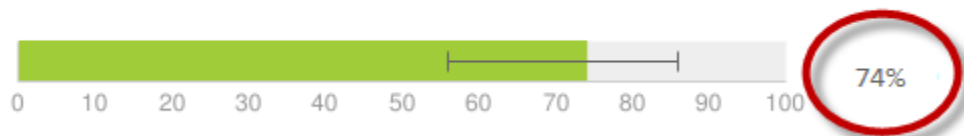
## Task success rate

---

Just as we saw for the study's overall score, the most important thing to look at for a task is success rate – **how many participants found the correct answer?**

This is typically rated out of 10 or 100. For example, a score of 71 means that 71% of the time, participants arrived at a correct answer:

### Success



The same rough markers we used for overall scores hold true for individual tasks:

Success score	Meaning
0-50	<b>Most participants had real trouble with the task.</b> If they understood the task properly, then the tree failed to do its job. If they didn't understand the task (see <a href="#">Where they went</a> later in this chapter), then we won't be able to judge the tree based on this task alone.
50-65	<b>A mixed bag: most found it, but quite a few didn't.</b> This usually indicates that the tree is roughly right for this task, but there are specific topics that are luring some users off the correct paths.
65+	<b>Most participants succeeded.</b> The tree is performing well for this task, and probably only needs minor tweaks to corral the wayward few.

**Success rate is a good general indicator of the effectiveness of our tree**, but immediately we want to dig deeper. For example, if we have a success rate of 27% on a task, we naturally want to ask "Where are all these people going wrong?". For that, we need to look at the paths they took through our tree.

---

**Next:** [Where they went](#)

## Where they went

- A typical success
- A typical failure
- Discovering more correct answers

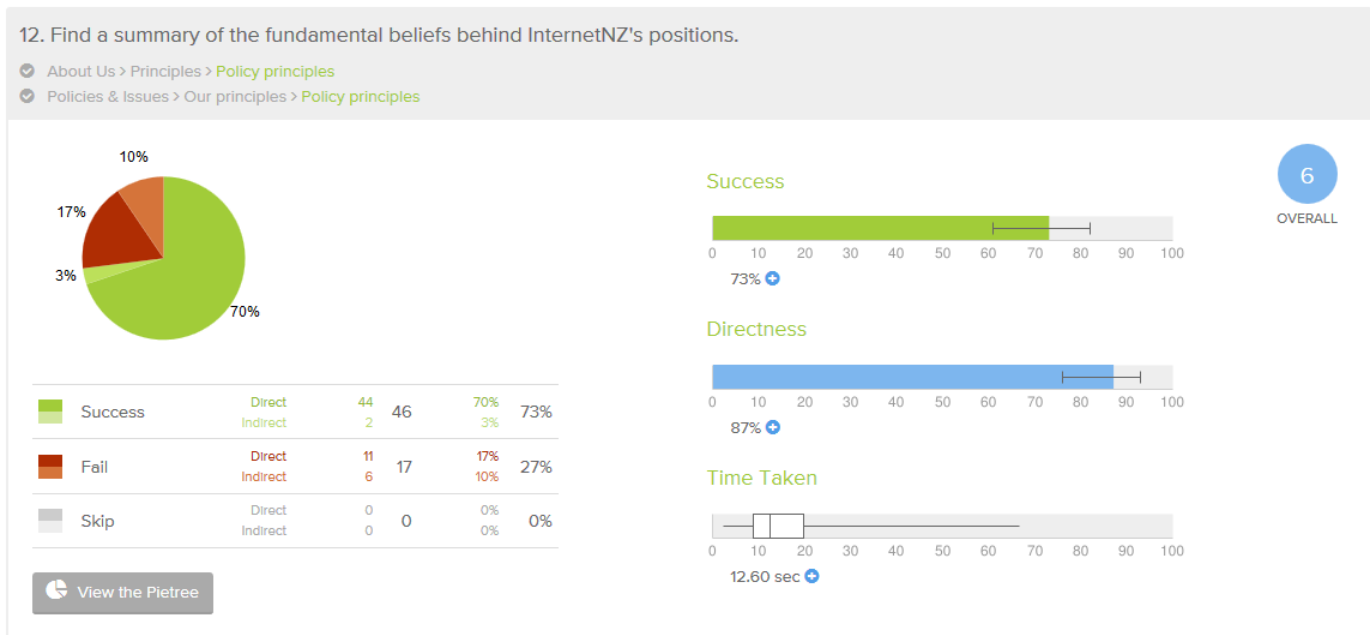
Once we get a score for a task, we naturally want to know why it scored like that:

- For high-scoring tasks, we want to know **which parts of the tree were particularly effective** in steering participants to the correct answers.
- For low-scoring tasks, we want to find out where participants went off the rails – **which parts of the tree confused them or lured them down the wrong path**.

## A typical success

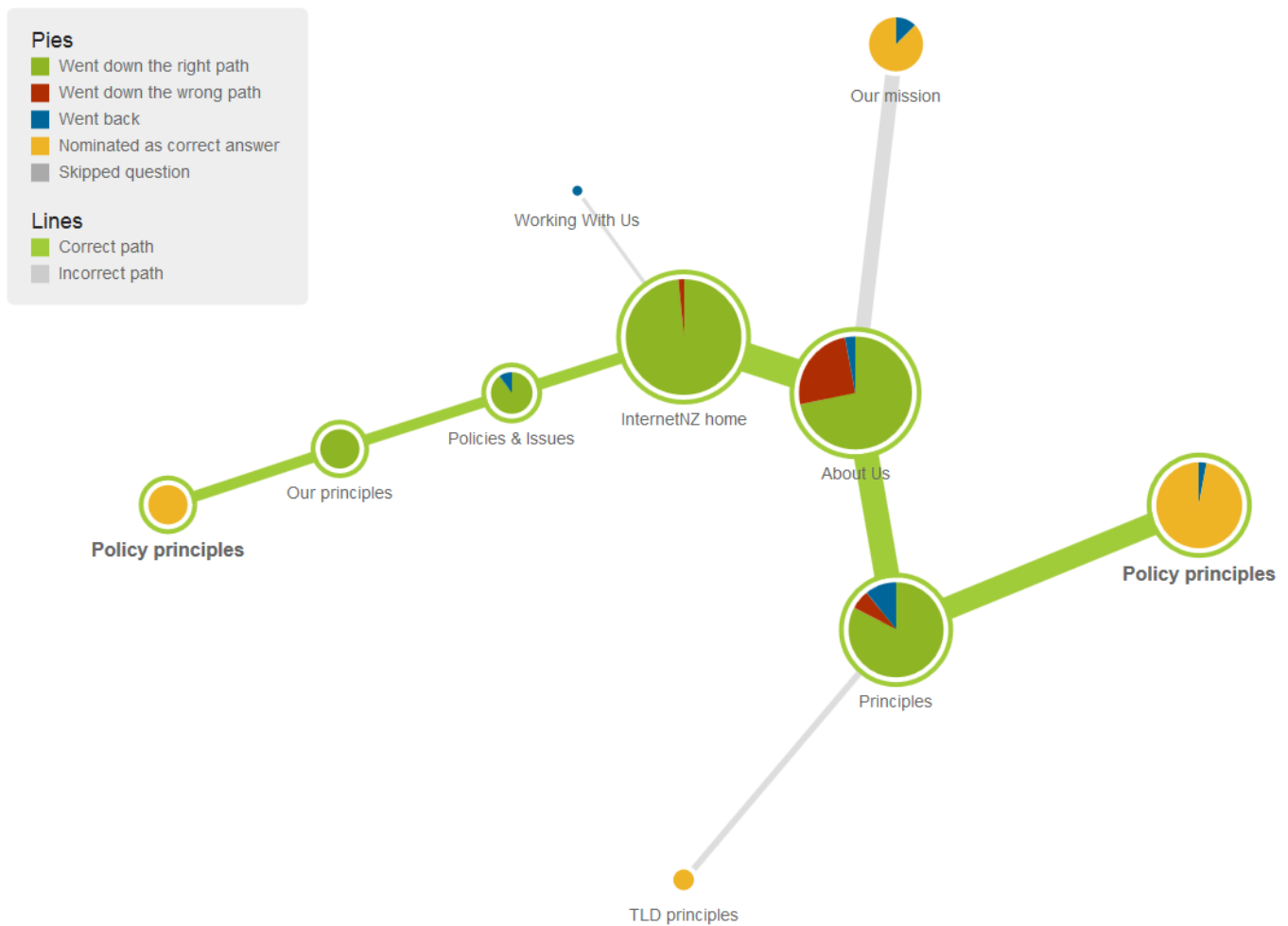
Let's start with an example of a high-scoring task, because they are usually easier to analyze.

This one comes from InternetNZ, a non-profit Internet advocacy group in New Zealand. When they redesigned their website, they ran several tree tests to check the effectiveness of the new structure. Here's a task that performed fairly well:



As we can see from the task summary, almost three quarters of our users found a correct answer, with very little backtracking. That's good to know, but it's not enough – **we need to see where they went, both for the correct and the incorrect cases**.

Most tools give us a way to examine the click paths for a given task. We used Treejack for this study, which offers a “pie tree” diagram showing where participants went:



For this example, let's concentrate on the paths, not so much the pies themselves:

- The **green** paths are the right ones.
- The **gray** paths are the wrong ones.
- The **thickness** of the line shows how many people went that way.

The first thing we notice is **how sparse the graph is** – it only shows paths that participants actually took, and they didn't take many different ones for this task.

When we run a tree test, this is what we want to see – **a small number of paths taken, lots of traffic on the correct paths, and everyone clear on what everything means**. The participants knew where they were going, and **agreed on the same answers**.

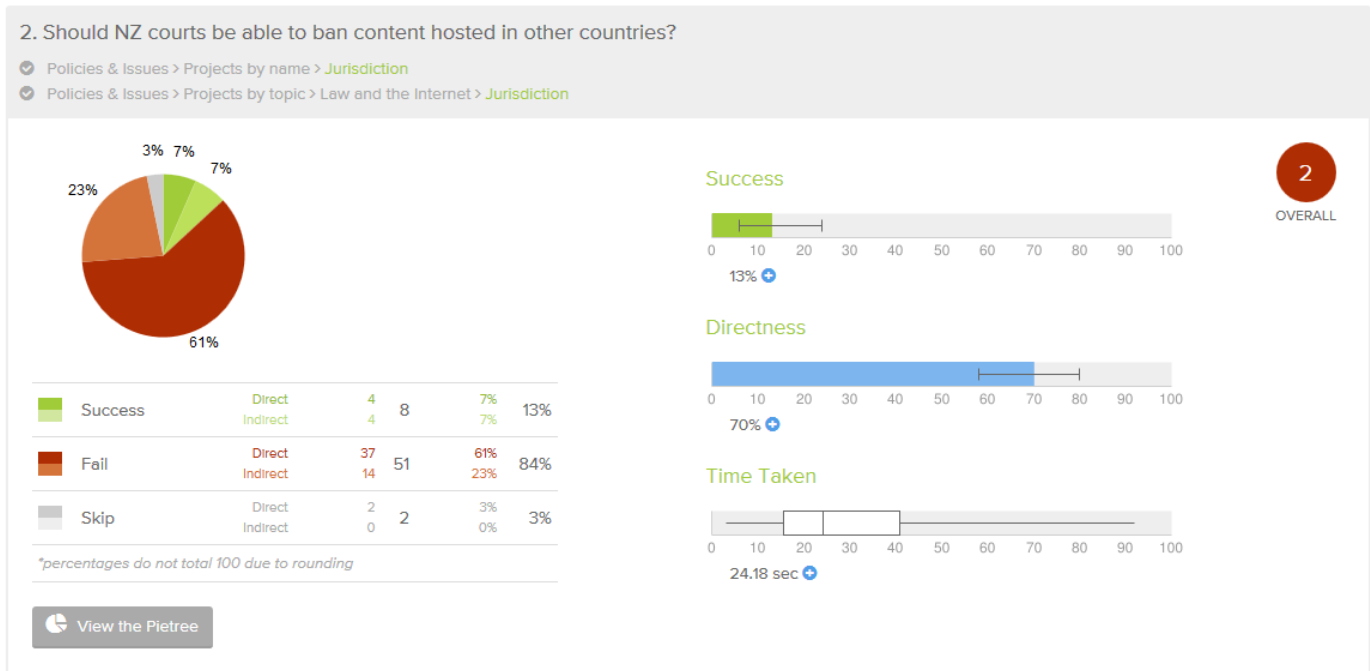
Note that there are **two correct paths** here. In the original tree, principles were only in the *Policies* section, but we saw most people go to *About Us*. So we put the *Principles* topic in both places. It would live in one place, and be explicitly cross-linked from the other. The same thing happened this time – most participants went to the *About Us* section, but this time there was a correct answer waiting for them.

Even the wrong answer is not so wrong here. We could feel pretty confident that users who went to an *Our Mission* page would get a partial answer for (and probably a link to) the organization's principles.

Notice also that there was very little "leakage" at the top level (the first click from the home page). Only 1 participant out of 63 made a wrong turn at the start. We'll discuss the importance of first clicks in *What they clicked first* later in this chapter.

## A typical failure

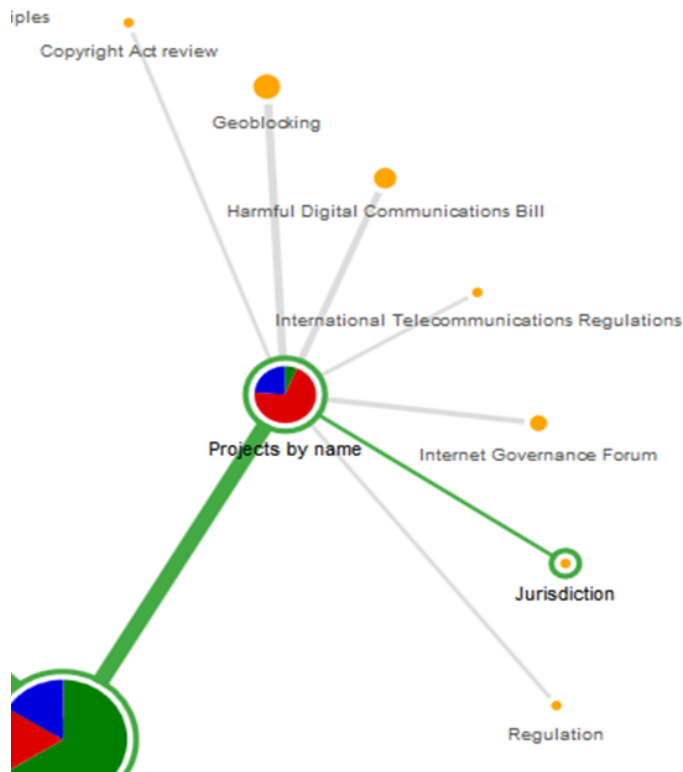
But tree tests are not all sunshine and lollipops. Some of our tasks will probably look like this one, again from InternetNZ:



The correct answer is under *Policies*, in a section called *Jurisdiction*, but **87% of our participants failed to find it**. Only 3% gave up, so where did all the others go?

The answer is that **they went everywhere**. Graphically, this is what “everywhere” looks like:

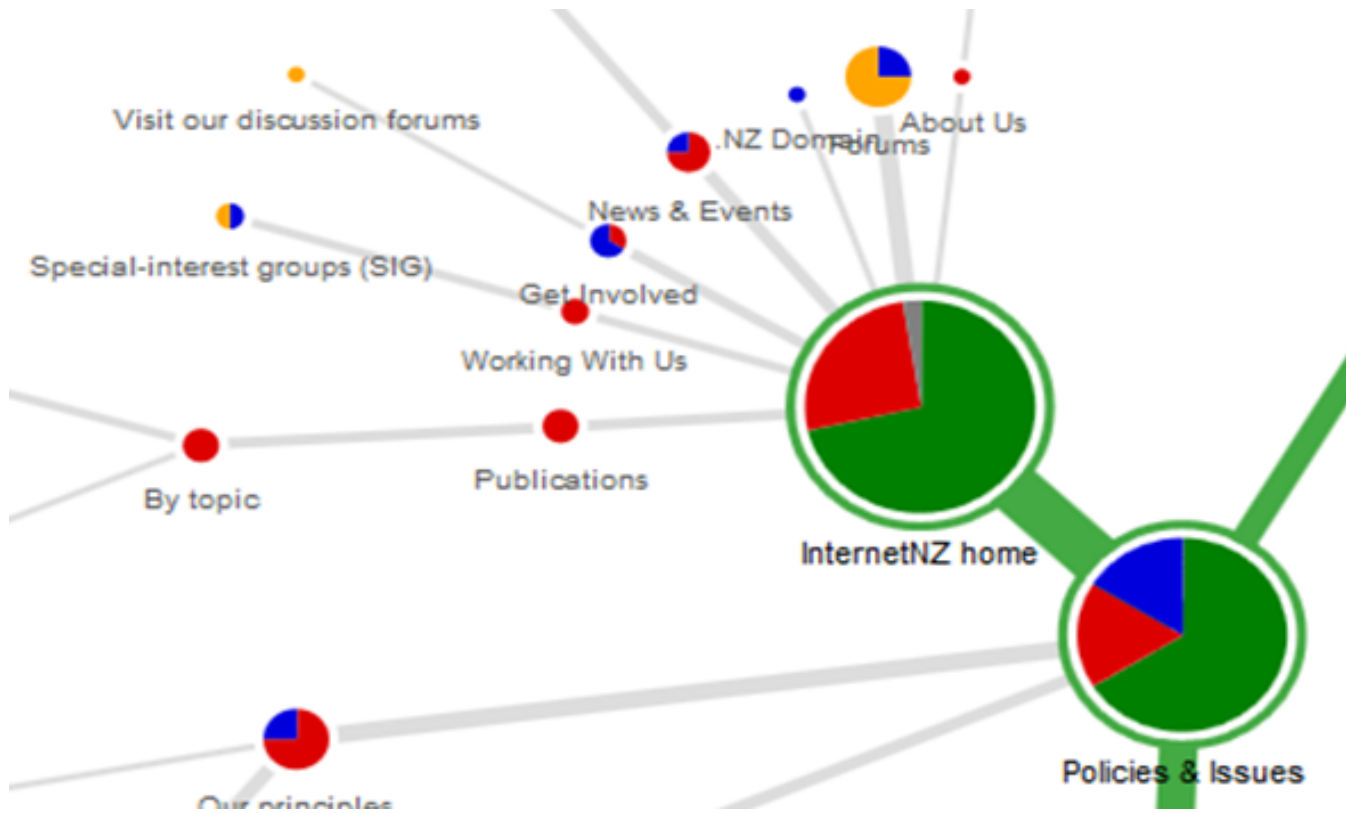




A large chunk of participants came here, and the correct answer (*Jurisdiction*) was waiting for them, but most didn't choose it (partly because it's more abstract than the other topics).

Not only did they not find the right answer, they couldn't agree on where to go instead. The **headings were not clear** themselves, and they were **hard to distinguish** from each other.

The graph makes another problem obvious too – the “leakage” at the top level of the tree:



Right at the start, as their first click, a large fraction of the participants **ran off in all directions**. This is not what we intend when we design a site structure. It suggests that either:

- **Our top-level headings are not clear and distinguishable** (which we should be able to confirm by looking for similar results in other tasks), or
- **The task itself was not clear** (probable if the top-level headings performed well in the other tasks)

We can also see scattering when we view the results as a spreadsheet. The **vertical cluster of cells** shows that, for a single task, participants chose a wide variety of subtopics under the correct topic.



Studying the click paths of a single task gives us insights into how the tree performed for that particular task. But **it's dangerous to draw conclusions based on only one scenario**. What we need to do is look for supporting evidence from the other tasks in our study – see [Finding patterns among tasks](#) later in this chapter.

## Discovering more correct answers

Participants select wrong answers all the time. And most of the time, they really *are* wrong. Occasionally, though, they're right, because we missed a correct answer in our tree.

It seems that no matter how thoughtfully we construct our trees and tasks up front, and mark which answers are correct, once we launch the test and start seeing where they go, **we always seem to find more correct answers that participants have “discovered” on their own**.

Often, it's clear that we missed an answer, so we should just fix it and recalculate our results – see [Cleaning the data](#) earlier in this chapter.

However, **we do need to be careful about changing our correct answers** based on incoming results, because adding correct answers will raise our scores. If we are analyzing a tree that we hope does well, it's often a bit too easy to convince ourselves that our participants' borderline answers should be marked as correct. For more on what we call "pandering to the task", see [Revising trees](#) in Chapter 14.

The best way to govern this is to agree on some consistent criteria for correctness ahead of time - see [Identifying correct answers](#) in Chapter 7.

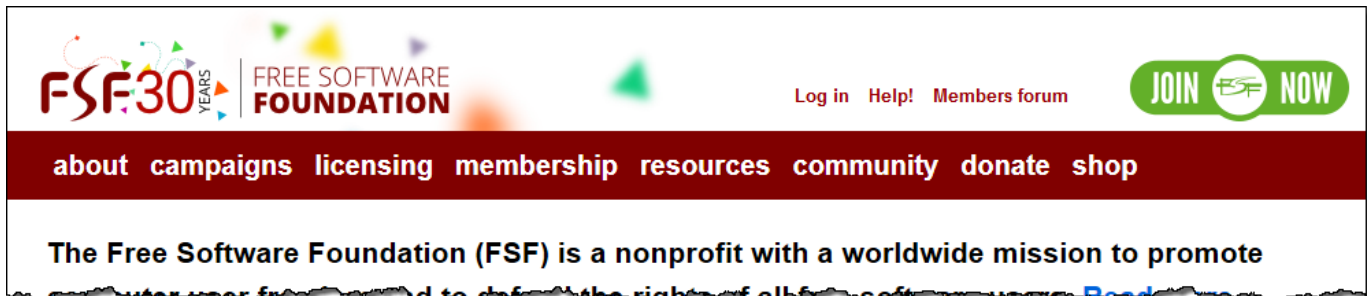


---

**Next:** [What they clicked first](#)

## What they clicked first

When we're evaluating a site structure, **one of the key things to get right is our top-level navigation**, also known as "level 1". These are typically the headings that appear in the global navigation, often as tabs across the top of the site:



Note that top-level navigation also includes the **utility links that appear in the header and footer** of most sites (items such as *Login*, *My Account*, *FAQ*, and so on). They're less prominent than the main headings, but that's a visual-design choice, and in tree testing we purposely don't factor in things like visual design.

**It's crucial to get our top-level headings right:**

- **They are the front line of our navigation.**  
They appear on every page of the site, so users see them a lot. This is one of the main ways users build a mental model of our content and how it's organized.
- **A correct first click can double the success rate.**  
[Bob Bailey's research on first clicks](#) shows that if we can get users started in the right direction, they're much more likely to find what they're looking for.

An obvious way to test the effectiveness of our top-level headings is to look at the clicks they made from the "home" node of our tree. This includes two cases:

- **The "first click" that participants make when they start a task.** If that first click is correct, their job becomes a lot easier, because they're now looking for their answer in a much smaller tree (the subtree for the section they chose).
- **Any subsequent clicks from the "home" node.** These only happen if the participant chooses one top-level heading, later changes their mind, backtracks to the home node, and chooses a different top-level heading.

While we can dive in and examine click paths to see where the top-level clicks are going, most tools provide an easier way to see this.

In Treejack, for example, the "First Click" tab that shows, for each task, which top-level headings were clicked. Here's an example from the InternetNZ tree test that we saw earlier in this chapter:

Task 10: Find out if InternetNZ manages the New Zealand country-code domain.		Visited first	Visited during
0	Our Work	50%	59%
1	Membership	2%	6%
2	Meetings	2%	4%
3	News		2%
4	About us	33%	72%
5	Login		
6	Privacy Statement	2%	2%
7	Disclaimer	2%	2%
8	Copyright		4%
9	Subsidiaries	9%	19%
Task 11: When was InternetNZ founded?		Visited first	Visited during
2	Our Work	4%	6%
3	Membership		
4	Meetings		
5	News		
6	About us	96%	100%
7	Login		
8	Privacy Statement		
9	Disclaimer		
0	Copyright		
1	Subsidiaries		

Task 11 (above) shows **a task where the top-level headings were very effective**. 96% of participants went to the right section (*About us*, highlighted in green) on their first click (the “Visited first” column). Besides that high score, notice also that **they agreed on where to go** – only 1 other top-level heading got any traffic at all.

Task 10, on the other hand, shows how **the same top-level headings were not effective for a different task**. Only 42% of participants got their first-click right (split between the two correct paths), while 50% mistakenly went to *Our work*. Notice also that several other top-level headings got traffic; this kind of “noise” shows that there was much less agreement on where the correct answer should live.

The “Visited during” column shows us **how often the top-level headings were clicked as a second (or third) try** after the participants backtracked from their first choice. For Task 10 above, the *About us* section only got 33% of first clicks, but that number increased to 72% when we added all subsequent clicks. This tells us that many participants made it their second choice after *Our work*. This in turn suggests that if we could keep people from going to *Our work* for this task, *About us* would likely become their first choice, raising our test score.

When we look at top-level clicks for a task, then, we’re mainly looking for two things:

- **Are the “correct” top-level headings getting the most traffic?** If not, which other top-level headings are?
- **Do participants agree on where to go, or do they scatter?** Are there 1 or 2 headings that get almost all the traffic, or is it “mushed” across several headings?

Top-level clicks can help us fix specific problems with individual tasks. But as we saw above, **top-level headings may work well for one task and not for another**. We need to look at all the tasks in our study to come up with a total assessment of our top-level headings.

---

**Next:** Directness – where they backtracked

## Directness – where they backtracked

- Directness scores
- Backing up from evil attractors
- Backing up from correct paths
- Examining click paths in more detail

We also need to know **how directly our participants found the right answer**. Did they have to wander around first, or did they try a first path then change their mind, or did they know where they were going from the start?

**Backtracking usually happens when a participant clicks a topic, expecting to see a certain set of subtopics, and gets something different.** They then decide they're in the wrong section, back up a level (or two), and try a different path.

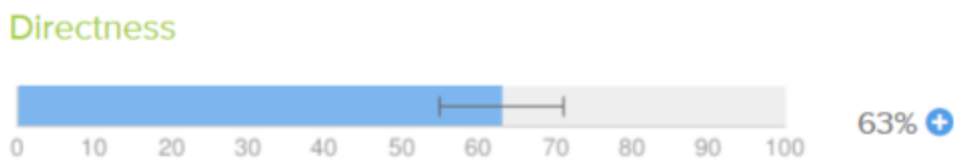
If we see a lot of backtracking at a certain topic, it suggests that either:

- The topic itself is attracting clicks when it shouldn't, or
- The subtopics under our topic seem like a dead end.

## Directness scores

As we saw for overall scores, backtracking can be measured in several ways:

- If the tool uses a simple yes/no measure for directness, then **70% is an average score for a task**, just as it is for the overall score. Less than that indicates that users are having trouble finding the right path.



- **~variable directness score and average**

**The main thing we're looking for is a score that's much lower than the average for the test.** That means that, regardless of their success rate, **participants are having a much harder time getting to where they're going:**

- Perhaps they're being lured down the wrong path by a **misleading top-level heading**, then have to double back when they discover they're in the wrong section.
- Perhaps they're finding **several headings that look promising**, and exploring each one until they find a good answer.
- Perhaps they're finding **no headings that look promising**, and they're reduced to guessing and wandering.

In any case, this is a trigger for us to dig deeper into the click paths to see what's really going on.

## Backing up from evil attractors

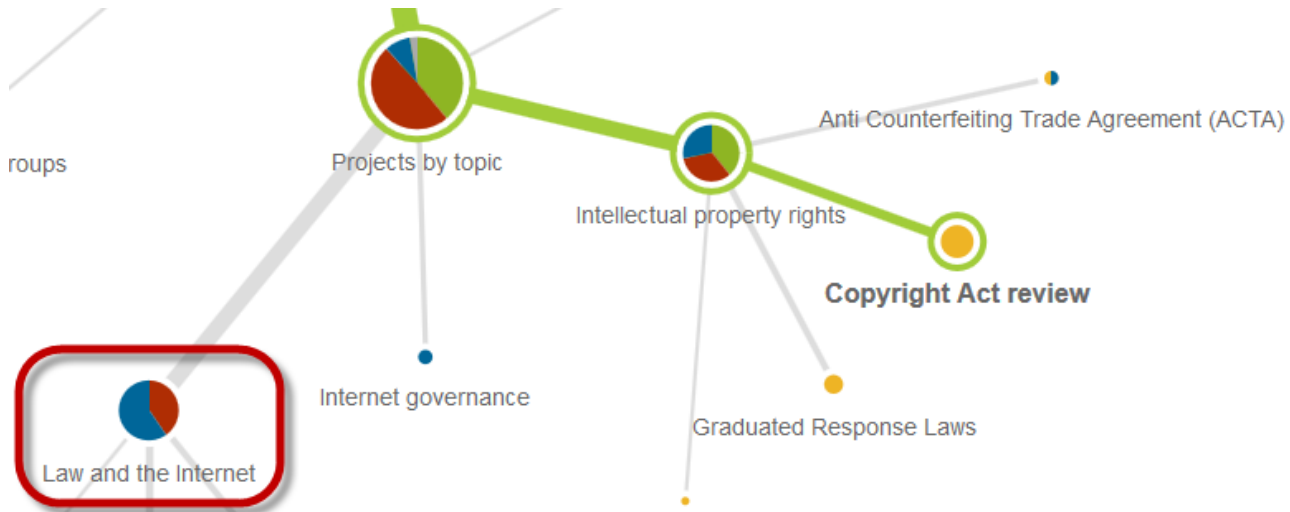
When we analyze a given task, we may find topics that attract a lot of clicks. We hope that the correct paths draw the most traffic, but often we find that **many participants are choosing an incorrect topic**.

Here's an example from InternetNZ, an Internet-advocacy organization. The task is:

| [Learn more about the penalties for sharing files illegally.](#)

In the "pie tree" diagram below, note how several participants chose "Law and the Internet". Most of them then backed up (the blue segment of the pie) when they didn't like the subheadings under it. "Law and the Internet" attracted their clicks (understandably), but the subheadings showed

them they were in the wrong place:



Sometimes this is just a “one off”, where the topic seems reasonable for the single task we’re analyzing. While we could “fix” this topic (by rewording it or moving it elsewhere), we need to be sure that our fix helps the tree in general (beyond this specific task), and that it won’t mess up any other scenarios that also go through this part of the tree.

Sometimes, though, we’ll see this topic **attracting incorrect clicks across several tasks**.

- **If the tasks are related** (for example, they’re all about getting support for products), then it means that participants think this topic is the place to go for those types of inquiries. We have a choice here: we can either reword the topic to avoid that incorrect interpretation, or we can add the desired content under that topic.
- **If the tasks are unrelated** (that is, they ask the participant to find very different things), then it’s likely that the topic is too general or vague. We call this an **evil attractor**, because it acts like a magnet to lure participants off the right path. For more, see [Discovering evil attractors](#) later in this chapter.

Occasionally, we may find a topic that attracts clicks when it shouldn’t, but it doesn’t seem like it has anything to do with the task. In this case, it may not be because of the topic itself, but **because of its sibling topics** (on the same level). If they are not clear and distinguishable from each other, the attractor topic may simply be the “best of a bad lot” – the participants’ best guess at this point in the tree. In this case, we’ll need to revise the sibling topics to make everything clearer at that level.

## Backing up from correct paths

Sometimes a topic attracts clicks when it should – that is, it’s on a correct path – but a lot of participants then backtrack.

This usually happens because those participants then see the subtopics, scratch their respective heads, and decide they’re in the wrong place. There are two common situations here:

- The **subtopics are not what they expected to find under this topic** (suggesting that they thought the topic itself meant something different than we intended), or
- The subtopics are what they expected, but they **didn’t find the specific subtopic they wanted** for this specific task – that is, our subtopic list was incomplete.

## Examining click paths in more detail

If we use a visualization like the “pie tree” graphs shown above, that may supply enough information for us to judge where participants got into trouble.

Sometimes, however, we need to see the **actual sequence of clicking** to determine how participants behaved during a task. For this, we can look at the **detailed click paths** for a task, usually presented in a spreadsheet like this:

	A	B	C	D	
	<b>Task: 1</b>		2	3	4
		You're not sure why power costs so much. How would you find out what makes up a typical invoice?	You'd like to let Acme know exactly how much electricity you used this month, so they don't send you an estimated bill. Find out how.	Does Acme supply natural gas?	Y h
2	1	> My account > My bill > How to read my bi	> Save energy & money > Agribusiness <	> About us > Generating energy < About us >	
3	2		> My account > My bill > Send my meter re		
4	3		> My account > My bill > How to read my bi		
5	4	> Reports & presentations	> Log in > MyAcme		[s
6	5			> About us > Generating energy > Our gen	>
7	6	> Save energy & money > Business > Equ		> FAQs > General enquiries	
8	7		> My account > MyAcme	> FAQs > General enquiries	>
9	8	> Why join us? > Join > Residential join for			
10	9				
11	10		> My account > My bill > Send my meter re	> About us > Generating energy < Acme hc	
12	11		> My account > My bill > Send my meter re		
13	12	[skipped]	[skipped]	[skipped]	
14	13	> My account > My bill < My account > MyA			
15	14			[skipped]	>
16	15	> Reports & presentations	> My account > MyAcme		>
17	16	> Reports & presentations	> My account > My bill > Send my meter re		
18	17	> My account > My bill > How to read my bi	> My account > MyAcme < My account > M	> FAQs < Acme home > Why join us? < Ac	
19	18				

This is too much to see at once, so we cut it down to the parts we're most interested in:

- We focus on **one task at a time**, widening that column to see the paths without wrapping.
- We **sort the column alphabetically** so all the similar paths are together.
- We **remove the paths where they went directly to a correct answer** (because these don't show us problems).

What we're left with is a detailed map of problems - an exact description of **where participants went partly or wholly wrong**:

	A	B
	<b>Task: 1</b>	
		You're not sure why power costs so much. How would you find out what makes up a typical invoice?
51		[skipped]
52		[skipped]
53		> FAQs > Billing & metering
54		> FAQs > Billing & metering
55		> FAQs > Billing & metering
56		> FAQs > Billing & metering < FAQs > Fees & payments
57		> FAQs > Fees & payments
58		> FAQs > Fees & payments < FAQs > Billing & metering < FAQs > Billing & metering
59		> FAQs > Pricing & plans
60		> FAQs > Pricing & plans
61		> FAQs > Pricing & plans
62		> FAQs > Pricing & plans
63		> Log in > MyAcme
64		> My account < Acme home > Reports & presentations
65		> My account < Acme home > Why join us? > Residential pricing > Rates by region
66		> My account > My bill < My account > MyAcme
67		> My account > My bill > How to read my bill < My bill > Got a high bill? < My bill < My account < Acme home > My account > MyAcme < My account > My bill > How to read my bill > How to read my bill
81		> My account > My bill > Usage check
82		> My account > MyAcme
83		> My account > MyAcme
84		> My account > MyAcme
85		> My account > MyAcme
86		> My account > MyAcme
87		> My account > MyAcme < My account > My bill > How to read my bill > How to read my bill
88		> Reports & presentations
89		> Reports & presentations
90		> Reports & presentations
91		> Save energy & money > Business > Equipment & industrial
92		> Save energy & money > Residential > Water heating & heating your home < Acme home > FAQs > Billing & metering < Acme home > FAQs > Pricing & plans < Acme home > FAQs > MyAcme
93		> Save energy & money > Residential > Water heating & heating your home < Acme home > My account < Acme home > [skipped]
94		> Why join us? > Join > Residential join form

We can now see patterns in where our tree failed, whether it's evil attractors luring clicks, or a certain heading betraying frequent backtracking.

---

**Next:** [Task speed - where they slowed down](#)

## Task speed - where they slowed down

- Task times
- Click times
- Why they slowed down

Another result we can measure is **speed (time taken)**. There are two metrics in play here:

- How long it takes participants to **complete a task**
- How long a participant takes **for each click**.

In both cases, high speed suggests *confidence*, or at least *clarity* - the participant found it easy to choose between competing headings. Conversely, if they took a long time to decide, this suggests that it was harder to understand the headings, and/or harder to choose between them.

## Task times

For a given task, some tools show us the average (or median) time it took participants to complete it:

### Time Taken



By itself, this is just a number. But if the tool also provides an average time across all tasks (or if we calculate it ourselves), we can then spot **which tasks took substantially longer to complete**. We can then drill into these tasks to look for possible causes.

**While task time is the most obvious measure for speed, it's problematic** because some answers are often deeper in the tree than others. Some tasks only take a few clicks to find (if they're on level 3 of the tree, for example), while others make take more clicks (if they're on level 5), so the total time is affected. For example:

- If a participant takes 15 seconds for the level-3 task, that's 3 clicks, meaning 5 seconds each, which is a bit slow.
- If the same participant takes 15 seconds for the level-5 task, that's 5 clicks, meaning 3 seconds each, which is considerably faster.
- This gets even more problematic if there are two correct answers for a task – for example, one at level 3 and one at level 5.

For this reason, we recommend **treating task times with a grain of salt**, and trying to factor in how many clicks were involved.

## Click times

**Ideally, we want to flag moments when clicks slowed down** – where a participant took longer than they usually do between clicks. If a participant falls below their usual "click pace" during a task, that's an indication that the participant took longer to understand their choices and make a decision.

The task's speed score can then be calculated as the percentage of participants who didn't slow down significantly during that task. For example, for each participant, we could decide that "slowing down" means at least one click time greater than a standard deviation from that person's average click time across the entire test.

This is a better measure of speed than the "task time" described above, because:

- It's not affected by how many clicks a task requires to find the right answer.
- It's not affected by the fact that some participants naturally click through the tasks faster than others.



This works because the **speed is measured relative to a single participant**. To spot a slowdown, we look for tasks where that person took a long time between clicks – and by “long”, we mean longer than that same person took for other tasks. For example:

- Suppose a “slow” participant takes 5 seconds per click as they move down the tree. But then they encounter a tough choice, and take 12 seconds to click. We would flag that task (even better, that exact spot in the tree) as a slowdown for that participant, because 12 seconds is substantially longer than their average of 6 seconds.
- Suppose a “fast” participant takes 2 seconds per click. When they encounter the same hard choice as above, they take 6 seconds to click. Again, we would flag that as a slowdown, because even though they did it in 6 seconds (normal speed for the first participant), it’s substantially slower than their own usual rate (2 seconds).

For a given participant, that slow spot might have been caused by any number of things – a tough choice in the tree, the doorbell ringing, anything really. But **if we look at all the participants who did that task**, we might see that most participants were slow at that spot. That suggests that those topics were hard to decide on for that task – a valuable thing to know.

## Why they slowed down

When we spot a task with a poor speed score, we need to **find out if there are specific locations in the tree that are to blame**. If the tool provides a way to inspect click times for tree headings (either with a graph or raw data that we can process), we can determine which parts of the tree are bottlenecks.

Once we locate the heading where participants are slowing down, there may be several reasons why it’s a bottleneck:

- **One or more of its subheadings are not clear** – that is, participants were unsure what those subheadings meant.
- **Its subheadings are not distinguishable** – that is, participants could not quickly see the differences between them.
- **None of the subheadings “answered” the task.**  
If we find a lot of backtracking at this spot, this is the likely cause.

---

**Next:** [Where they gave up](#)

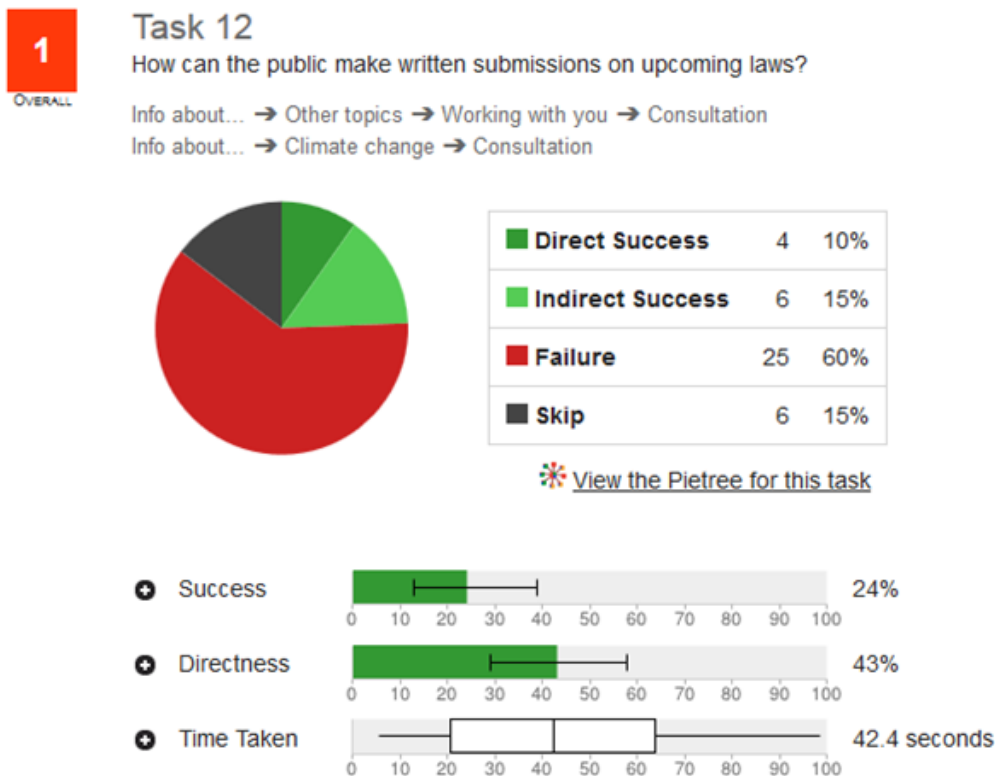
## Where they gave up

Sometimes, when participants don't find an answer they like, they back up and try a different path. Other times, they just give up.

Whether this happens after a few tentative clicks, or after an exhaustive search of every likely path, the fact that participants give up tells us that **our tree failed those people for that particular task**.

It's useful, therefore, to look at tasks where an **abnormally high percentage of the participants give up** (that is, instead of choosing an answer, they click a "Skip" button or something similar). For a tree of average size and complexity, we typically look for **skip rates of 10% or more**.

Here's an example from New Zealand's Ministry for the Environment. They suspected that it was hard for people to find out about consultations (where the government asks the public for input on proposed laws), so they created a task to find out. Here's the result:



The first thing we see is that *they were right* – only 25% of participants found the correct answer, because it was buried in an obscure part of the site. Note also the low Directness score (showing that more than half the participants had to back up at least once) and the long time they spent looking (more than 40 seconds on average).

The final evidence that this was really hard for participants is the skip rate – *15% gave up during the task*, compared to an average skip rate of 3% for the other tasks.

When we see a high skip rate for a task, we do two things:

- We look at the **click paths for this task** and see if there are **any patterns in where the participants gave up**. If they all lost hope in the same section, that helps us narrow down the culprit.
- We **compare this task to other tasks with high skip rates**:
  - If we see that a **certain section** has a lot of drop-outs between the tasks, this suggests where we should fix the tree.
  - If we see that **certain types of tasks** have high skip rates (for example, tasks where we ask them to do research on a product), this suggests that the tree is not supporting this type of task as well as it should.

---

**Next:** [Confidence](#)

## Confidence

---

- do any tools measure this?
  - <http://www.measuringu.com/blog/tree-testing-ia.php>
  - <http://www.measuringu.com/blog/first-click.php>
  - <http://www.measuringu.com/blog/cardsort-tree-test.php>
- 

**Next:** Dealing with outliers

## Dealing with outliers

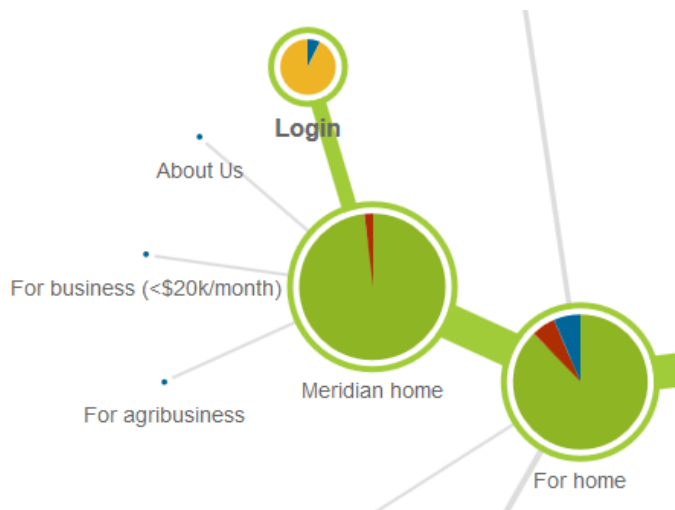
When we see lots of participants choosing a certain answer, or following a certain path through the tree, there's not much to argue with – clearly there's some reason why so many of them made the same choice.

But often we see just a few participants (or even a single person) choose a certain answer or path. What should we make of that? **Is it meaningful, or just an outlier that we can safely ignore?**

For example, here's a task from a tree test run by Meridian Energy (a power company):

| *How much electricity did your household use last week?*

And here's where they went from the *Meridian Home* page:



The few participants who chose *About Us* or *For Business* or *For agribusiness* may have done so honestly (because they thought it was the best answer) or because they were tired/bored/hurried and just clicked randomly. In an unmoderated online study, it's hard to know which.

Earlier in this chapter, in [Cleaning the data](#), we made sure we deleted the results of participants who “guessed” too often. But we may still have some cases where participants chose randomly on a task here or there.

As in any quantitative research, the **key to spotting and ignoring true outliers is a large number of participants**. Suppose, for example, that 3 participants choose a certain incorrect answer in a study:

- If it's 3 participants out of 100, that's 3% of responses, and we can safely ignore it.
- If it's 3 participants out of 20, that's 15%, which is much harder to ignore.

This, of course, is why we recommend getting at least 50 participants per user group:

- When we have small numbers, any answer that gets a few clicks must be considered.
- **When we have large numbers, it's much easier to spot outliers and ignore them.** 50 participants is a good minimum for this, and 100 or more make it very easy to identify answers we can ignore.

**As a rough rule of thumb, we ignore results of about 5% or less.** For 50 participants, that means ignoring answers that get 1 or 2 (or even 3) clicks, which is very common in tree-test results.

**Next:** [Finding patterns among tasks](#)

## Finding patterns among tasks

As Aristotle might have said (if he had been an information architect), “one task doth not a tree make”. It’s hard to judge the effectiveness of a tree (or even part of a tree) from a single task. **What we’re really looking for are patterns of behavior across several related tasks.**

For example, if we have **one task** where participants scatter at a certain mid-level topic, the topic may need revising. However, it may also be a problem with the wording of the task. We need to be careful because it’s just a single task.

On the other hand, if we have **several tasks** that send participants through this part of the tree, and they keep scattering at the same topic, it’s safe to say that the topic is the problem.

In our experience, these are the most common patterns we see across tasks:

Pattern	Observation	Probable cause
<b>Scattering under a topic</b>	Across several tasks, participants who pass through a particular mid-level topic all pick different subtopics.	The subtopics are not clear and distinguishable.
<b>Backtracking at a topic</b>	Across several tasks, participants who click a particular topic (usually a mid-level topic) look at its subtopics and then back up a level.	They are misinterpreting the topic altogether. It could also mean that we are missing an important subtopic, but that’s less likely when the same thing happens across different tasks.
<b>A topic attracts clicks when it shouldn’t</b>	Across several tasks, participants keep choosing a topic even though it’s wrong for most of them. They may then scatter or backtrack as described above.	The topic may be an evil attractor – see <a href="#">Discovering evil attractors</a> later in this chapter.
<b>A topic doesn’t attract clicks when it should</b>	Participants keep avoiding a certain topic, and this happens across several tasks where it’s on a correct path.	They are interpreting that topic differently than we intended. We may need to do some in-person testing to probe this further, then reword the topic according.
<b>Giving up on a certain kind of task or in a certain section</b>	Several similar tasks have high skip rates.	The tree is not supporting those tasks as well as it should. If we notice that people are often giving up in a particular part of our tree, that indicates that the headings in that section are not what people expect. For more, see <a href="#">Where they gave up</a> earlier in this chapter.

**Next:** [Analyzing by branches](#)







## Analyzing by user group or other criteria

- Analyzing by user group
- Analyzing by other criteria

Splitting our analysis by user group or other criteria (such as region) is a common need, especially for larger websites.

### Analyzing by user group

If we have more than one user group visiting our site (and most of us do), analyzing a tree test is a lot easier if we analyze each user group separately.

If we run a separate tree test for each user group (as we discussed in [Different tasks for different user groups](#) in Chapter 7), then we're going to have a separate set of results for each test, so that's already taken care of.

**If, however, we run a single tree test across all of our user groups, we recommend doing a separate analysis for each group.** That way, we can see the differences in their performance, instead of having their respective highs and lows mashed together into middling scores.

Suppose, for example, that we were looking at the results for a single tree test of the Shimano website. In [Which part of the tree?](#) in Chapter 6, we saw that Shimano has 3 major types of users – cyclists, anglers, and rowers. And suppose that the cyclists and anglers had no problem with tasks that covered the *Corporate* section of the site, but rowers failed miserably at those. If we looked at the results of all users together, the high scores of the cyclists and anglers would get mixed in with the low scores of the rowers, and we would just see a middling composite score, with no easy way to find out what caused it.

If, however, we could separate the groups and do the same analysis for each group, we could see that it was the rowers who had problems – the other two groups were just fine.

In [Adding survey questions](#) in Chapter 8, we suggested **using survey questions to identify user groups**, so we could easily pull them apart later for analysis. Now that we're ready to analyze the data, we need to:

1. **Narrow the results to a single user group** by filtering the data to only include participants who chose that user group in the "who are you" survey question.
2. **Regenerate the results** (if this is not done automatically).

Here's an example where we want to see if there are differences between members, prospective members, and the general public.

To get the results for prospective members, we filter our data set using a "who are you" survey question:

## Your Participants ?

The screenshot shows a web interface for filtering participants. At the top, there is a search bar and a button labeled "Apply". Below that, there is a section for filtering participants. A red circle highlights the filter options: "Select participants who answered" and "Which describes you best? Not a member, but considering it". There is also a button labeled "Update" and "Done".

That selects 26 participants. We now we regenerate the results based on that subset of prospective members:

## Your Participants ?

26 Reload results using current selection Apply Search

Select participants who answered

Which describes you best? with

Not a member, but considering it Update Done

Once we've generated the results for each group, analysis proceeds as we've described elsewhere in this chapter. But at each step of the analysis (success rates, backtracking, first clicks, etc.), we can compare how the groups performed and look for any major discrepancies between them.

### Analyzing by other criteria

Just as we can split our analysis by user group, we can do the same for other factors such as:

- Region
- New vs. existing customers
- Age or life stage
- ...and so on.

In all cases, we need to either:

- **Run separate tree tests for each cohort.**

For example, if we're analyzing by region, we could run separate tests for region 1, region 2, and region 3.

If we intend to use different tasks for the different groups, then we should definitely run separate tests – see [Different tasks for different user groups](#) in Chapter 7.

- **Run a single test, using a survey question to identify each factor.**

If we're intending to use the same set of tasks for each cohort, we could add a "Which region do you live in?" survey question, then filter the results using the various values of this question (as described above for user groups).

---

**Next:** [Discovering evil attractors](#)

## Discovering evil attractors

- The false scent
- What makes an attractor “evil”?
- Spotting an evil attractor
- Why do they happen?
- How do we get rid of them?
- Evil attractors as waypoints

Usability guru Jared Spool has written extensively about the [scent of information](#) – how users hunt through a site, click by click, to find the content they’re looking for. Tree testing helps us deliver a strong scent by improving:

- **organization** (how we group headings and subheadings), and
- **labeling** (what we call each of them).

Anyone who’s watched a spy film knows that there are always **false scents and red herrings** to lead the hero astray. Anyone who’s run a few tree tests has probably seen the same thing – **headings that lure participants to the wrong answer**. We call these “**evil attractors**”. These are headings that lure participants down the wrong path – not just for one task, but for several different tasks.

### The false scent

One of our favorite examples of an “evil attractor” comes from a tree test we ran for consumer.org.nz, a NZ consumer-review website much like Consumer Reports in the USA. Their site lists a wide range of consumer products in a tree several levels deep, and they wanted to try out a few ideas to make things easier to find as the site grew bigger.

We ran the tests and got some useful answers, but we also noticed that there was one particular subheading (*Home > Appliances > Personal*) that got clicks from participants **looking for very different things** – mobile phones, vacuum cleaners, home-theater systems, and so on:

	Should I buy a Canon digital camera? Are they the best?	I want to find the best broadband plan for my needs.	I'm looking to buy a cordless drill. Which one is best?	Which brand of notebook (portable) computer is most reliable?	I need a new mobile phone - which one should I buy?
Home					
Appliances					
Heating		2			
Kitchen - large					
Kitchen - small			1		
Laundry and cleaning					
<b>Personal</b>	5		4	3	11
Prices and reliability	1		10		
Ventilation					
Cars					
Accessories					

The website intended this “personal appliance” category to be for products like electric shavers and curling irons, but apparently “Personal” **meant many things** to our participants, because they also went there for “personal” items like mobiles and cordless drills that actually lived somewhere else.

This is the **false scent**, the heading that “**attracts**” clicks when it shouldn’t, leading participants astray. Hence this definition:

Evil attractor: A heading that draws unwanted traffic in several unrelated tasks.

### What makes an attractor “evil”?

Attracting clicks isn't a bad thing in itself. After all, that's what a good heading does – it attracts clicks for the content it contains (and discourages clicks for everything else).

“Evil” attractors, on the other hand, **attract clicks for things they shouldn't**. They “lure” users down the wrong path, at which point the user either find themselves in the wrong place, backs up, and tries elsewhere (if they're patient), or gives up (if they're not). Because these attractor topics are magnets for the user's attention, **they make it less likely that our user will get to the place we intended**.

The other “evil” part of these attractors is the way they **hide in the shadows**. Most of the time, they don't get the lion's share of traffic for a given task; instead, they'll siphon off 5-10% of the responses, luring away a fraction of users who might otherwise have found the right answer.

## Spotting an evil attractor

The easiest attractors to spot are those at the “answer” end of our tree, where participants ended up for each task. If we can **look across tasks for similar wrong answers**, then we can see which of these might be evil attractors.

We use the same results view that we saw in the “Analyzing by user group” section above - a matrix that shows the tree down the left side and the tasks across the top. Here's part of the view from the consumer.org.nz study:

	Should I buy a Canon digital camera? Are they the best?	I want to find the best broadband plan for my needs.	I'm looking to buy a cordless drill. Which one is best?	Which brand of notebook (portable) computer is most reliable?	I need a new mobile phone - which one should I buy?	LCD or Plasma TVs - which one should I buy?	Which vacuum cleaner is best?
Home							
Appliances							
Heating		2					
Kitchen - large							
Kitchen - small			1				1
Laundry and cleaning							66
Personal	5		4	3	11	3	1
Prices and reliability	1		10			9	9
Ventilation							
Cars							
Accessories							

Normally, when we look at this view, we're looking down a column for big hits and misses for a specific task. To look for evil attractors, however, we look for **patterns across rows**. In other words, we're looking horizontally, not vertically.

If we do that here, we immediately notice the row for “Personal” (which we've highlighted yellow for this example). See all those hits along the row? Those indicate an attractor – steady traffic across many tasks that seem to have little in common.

But remember, traffic alone is not enough – we're looking for **unwanted traffic across unrelated tasks**. Do we see that here?

Well, it looks like the tasks (about cameras, drills, laptops, vacuums, etc.) are not closely related – we wouldn't expect users to go to the same topic for each of these. And the answer they chose – “Personal” – certainly isn't the destination we intended. While we can probably rationalise why they chose this answer, it is definitely **unwanted** from an IA perspective.

So yes, in this case, we seem to have caught an evil attractor red-handed. “Personal” is clearly a heading that's getting steady traffic when it shouldn't.

## Why do they happen?

It's usually not hard to figure out why an item in our tree is an evil attractor. In almost all cases, it's **because the item is vague or ambiguous** – something that could mean a lot of different things to a lot of different people.

Look at our example above. In the context of a product-review site, “Personal” is too general to be a good heading. It could mean products we wear, or carry, or use in the bathroom, or any number of things. So, when those participants come along clutching a task, and they see “Personal”, a few of them think “That looks like it might be what I'm looking for”, and they go that way.

Individually, those choices may be defensible, but as information architects, are we really going to group mobile phones with vacuum cleaners? The “personal” link between them is tenuous at best.

## How do we get rid of them?

Just as it's easy to see why most attractors attract, **it's usually easy to fix them**.

Evil attractors trade in vagueness and ambiguity, so the obvious remedy is to try to **make those headings more concrete and specific**.

In the consumer-site example, we looked at the actual content under the “Personal” heading. It turned out to be items like shavers, curling irons, and hair dryers. A quick discussion yielded “Personal care” as a promising replacement – one that should keep away people looking for mobile

phones and jewelry and the like.

In the second round of tree testing, among the other changes we made to the tree, we replaced “Personal” with “Personal care”. A few days later, the results confirmed our thinking – our former evil attractor was no longer luring participants away from the correct answers:

	Should I buy a Canon digital camera? Are they the best?	I want to find the best broadband plan for my needs.	I'm looking to buy a cordless drill. Which one is best?	Which brand of notebook (portable) computer is most reliable?	I need a new mobile phone - which one should I buy?	LCD or Plasma TVs - which one should I buy?	Which vacuum cleaner is best?
Home							
Appliances							
Heating							
Kitchen - large							
Kitchen - small							
Laundry and cleaning							29
Personal care							
Reliability & running costs			4		1	4	9
Ventilation							
Cars							
Accessories							

## Evil attractors as waypoints

Above, we learned how to spot evil attractors that were final endpoints in our tree. However, topics higher in the tree can also be evil attractors. They may be top-level headings, or they may be in lower levels. But in all cases, they are **waypoints that are attracting traffic when they shouldn't**, across several unrelated tasks.

- **Top-level headings are usually easy to check for evil attractors**, if the tool we're using has a way of highlighting “first clicks”. If we see a level-1 topic getting lots of incorrect clicks across several tasks, chances are that it's an evil attractor. A very common culprit is an “other stuff” heading like *Resources*. It's so general that it will get all kinds of traffic for all kinds of reasons.
- **Mid-level headings typically need a bit more detective work** to see if they're evil attractors. Usually this means eyeballing all of our task results to see if certain mid-level topics are luring clicks when they shouldn't. For more, see [Where they went](#) earlier in this chapter.

---

**Next:** Chapter 12 - key points

## Chapter 12 - key points

---

If we're using an online testing tool, we'll probably need to do some **data clean-up** before analyzing the results.

**Overall scores** (based on factors such as success rate, directness, and speed) give us a quick idea of our tree's effectiveness and how it compares to other trees.

**Task scores** will show us where the tree is performing well or poorly.

**Low task success rates** show us where participants got confused, disagreed about where the answer would be found, or were lured down the wrong path by a promising (but wrong) intermediate heading.

**First clicks** are especially important to analyze, because they greatly affect success rates.

Finding **where people backtracked** helps us find **evil attractors** and dead ends.

Spotting areas of our tree that **slowed participants** shows us where subheadings need to be clarified.

Look for **patterns** of behavior across **several related tasks**.

Look for **patterns** of success or failure **across different sections** of the tree.

We can **compare the results** of different user groups (or other criteria) if we run separate tests for each, or if we run a single test and filter results by a corresponding survey question.

---

**Next:** [Chapter 13 - Communicating results](#)

## 13 - Communicating results

*"Two roads diverged in a wood, and I -  
I took the one less traveled by,  
And that has made all the difference." - Robert Frost*

In the previous chapter, we analyzed the results of our tree tests. We should now have a good idea of how each tree performed, which elements worked well, and which didn't.

That's great, but those findings are currently stored in our heads and in a mixture of paper notes, spreadsheet entries, and hard-copy markups.

- If we're working with a team, we'll need to convert these into **something that our colleagues can easily digest**, understand, and act on.
- Even if we are working alone, it's good to clean up our analysis and summarize our findings so that we can easily **refer back to them later**.

In this chapter, then, we'll cover:

- How to record our findings in a format that can be easily understood by anyone
- How to turn findings into concrete recommendations for our website
- How to report our results, either as a quick summary or a full report
- How to pass additional participant feedback along to the project team

---

### Recording findings

Examples, marking up hardcopy, using a matrix, etc.

### Turning findings into actions

Obvious vs. considered actions, recording them

### Summing up the basics

For a single tree or for several trees

### Reporting in more depth

What to cover, templates for slides and documents

### Passing along participant feedback

Collecting, cleaning up, and forwarding to the right people

### Key points

---

**Next:** [Chapter 14 - Revising and retesting](#)



## Recording findings

- What makes an effective "finding"?
  - Examples of findings
  - Marking up the hardcopy
  - Using a tree/task matrix
  - Adding findings to the tree spreadsheet
- 

When it comes to taking notes, everyone seems to work out their own favorite method over time. Some take longhand notes in a journal, some prefer to red-pen hard-copy output, and so on. Below, we describe some methods that have worked well for us in past studies. Feel free to mix and adapt them as needed.

### What makes an effective "finding"?

A finding is a conclusion we come to after observing patterns in our data. When compiling our findings, we must keep in mind that:

- Each finding comes **from the results of one or more tasks**, not our own personal opinion of a given tree.
- Each finding that we record describes a success or an issue, but **does not jump to a solution** just yet. (Those come later – see below.)
- Each finding **applies to a specific part or characteristic of the tree**.

For example, findings such as "This tree confused users" are too vague to be useful. We should be specific, as in "New customers had trouble navigating the top-level headings".

### Examples of findings

Here are some examples of effective findings, to give us an idea of what we should be aiming for:

*"Although the Meetings section attracted traffic for the right tasks, once there, many users backtracked (presumably because the Calendar subtopic wasn't specific enough)."*

*"The Our Work section was too general (attracting clicks on almost every task) and also too vague (not attracting traffic for some of its subtopics)."*

*"The audience-based sections (Special Education, Maori, and Pasifika) performed well overall, attracting traffic only when relevant to the task."*

*"The Boards/Principals section and the Admin section were continually confused by participants."*

### Marking up the hardcopy

The most obvious way to record findings is to print the results of each task and mark them up with a red pen, circling items of interest and jotting down findings (and follow-up questions) beside them. This method is simple and direct, and can be cleaned up for a slide presentation later.

However, because the findings are scattered across disparate tasks, it's harder to pull together the big picture – patterns that recur across tasks or across several trees. So, we recommend:

- **Doing a first pass on a given tree**, examining each task separately.
- **Doing a second pass on that tree** to look for insights across tasks, and record these on a separate sheet.
- **If we're testing several trees, doing a third pass** to look for similarities and differences across all the trees.

### Using a tree/task matrix

Another way to recording findings is to list the tasks down a column of a spreadsheet, then list our findings beside each task. (The example below also uses color coding to indicate the task's success rate.)

residential tasks		
4	You're looking for testimonials from other home-owners who use Meridian.	Why Join got traffic (correct). About Us got traffic, then scattered. Lots of leaks at L1 (including FAQ).
5	You live in Tauranga and you're wondering how much your monthly bill would be if you were with Meridian.	Why Join got traffic (correct). SEM got traffic (misinterpreted). FAQ got some traffic.
6	You just got a new meter installed at your house. You heard there are special deals for people with these types of meters. Find out more.	SEM got traffic (misinterpreted). Why Join got traffic (correct). My Account got traffic (existing customer), but not there. FAQ got traffic, but scattered.
7	You've decided to choose Meridian for your home. How would you apply?	Why Join got traffic (correct).
8	You'd like to settle your house's electricity charges using the web.	Most went to My Account > Pay (correct). Many went to MA > My bill (reasonable).
9	How much electricity did your household use last week?	Most went to My Account (correct). Many went to MA > My bill (reasonable). Minor traffic to FAQ, SEM.
10	Your new house has solar panels. Will Meridian buy back the excess power you produce?	SEM got traffic (correct), but scattered. SEM > Resi got traffic, but not there. FAQ got traffic. More leaks at L1.

This method really comes into its own when we **compare several trees** that we tested with the same (or very similar) tasks:

A	B	C	D	E
		tree 1 (audience C)	tree 2 (actions)	across trees
#	<b>General tasks (not audience-specific)</b>			
1	You're not sure why your power costs so much. How would you find out what makes up a typical invoice?	MA got traffic (correct), then scattered. Resi got traffic, then scattered. About Us got minor traffic (correct).	Most went to YA (correct). Plans got traffic, then scattered. No one thought generally enough to go to WWD.	Anything related to a pragmatic task (e.g. bill breakdown) needs to be in on the main sections, or cross-linked from there.
2	You'd like to let Meridian know exactly how much electricity you used this month, so they don't send you an estimated bill. Find out how.	MA got traffic (correct). Resi got traffic (not sales), then scattered.	Most went to YA (correct). A few went to Login and FAQ (correct).	
3	Does Meridian supply natural gas?	About Us got traffic (correct), then scattered. Resi got traffic, then scattered.	No agreement on where this lives. WWD got traffic (reasonable).	About Us and WWD should also be clear on what we don't do.
4	You're looking for testimonials from other home-owners who use Meridian.	About Us got traffic, then scattered. Resi got traffic (correct).	Better got big traffic (correct). About Us got traffic. Some L1 leaks.	Should About Us include (or cross-link to) customer stories?
5	You live in Tauranga and you're wondering how much your monthly bill would be if you were with Meridian.	Resi got traffic (correct). Minor leaks elsewhere.	Plans got big traffic (correct). Better got some traffic.	
6	You just got a new meter installed at your house. You heard there are special deals for people with these types of meters. Find out more.	Resi got traffic (correct), then scattered at 2 levels (e.g. Offers, plans). MA got traffic (reasonable).	Better got traffic, mostly Offers (reasonable). Plans got traffic (correct). Lots of L1 leaks	My Account should cross-link to switching plans/meters.
7	You've decided to choose Meridian for your home. How would you apply?	Resi got traffic (correct).	Join got big traffic (correct).	Join doesn't have to be at L1, but doesn't hurt either.
8	You'd like to settle your house's electricity charges using the web.	MA got traffic (correct), Login will work in future. Resi got traffic (not sales).	YA got big traffic (correct). Some chose eBills (reasonable). Login got traffic (correct).	
9	How much electricity did your household use last week?	MA got traffic (correct), then scattered. Resi got traffic (not sales).	YA got big traffic (reasonable). Login got minor traffic (correct).	
10	Your new house has solar panels. Will Meridian buy back the excess power you produce?	Resi got traffic (correct), but Offers stole some. About Us got traffic (reasonable). Some L1 leaks	Better got traffic (reasonable). Plans got traffic (correct). Leaks at L1.	
	<b>across tasks</b>	<b>43%</b>	<b>47%</b>	
		L1 segments did well for sales tasks.		
		Segments were browsed for non-sales content.		
		Plans vs. Offers - users were not good at choosing the right one.	Plans vs. Offers - users were not good at choosing the right one.	
			FAQ attracted some traffic across most tasks - an attractor, mostly evil.	



- We added a **column for the findings of each tree**.
- We added a **summary column on the right**.  
For each row (i.e. for each task), this lets us adding common findings across all trees.
- We add **summary rows at the bottom**.  
For each column (i.e. for each tree), we can add common findings across all tasks for that tree.

This gives us a matrix of trees vs. tasks, where **all of our findings are in a single place**, making it easier to see the bigger patterns. It's also a compact way of coming back to our findings later.

Here's a sample Excel spreadsheet of findings, using the method shown above:



Note that we've used a spreadsheet here, but we've also done this matrix using a whiteboard, adding each finding as a color-coded sticky note (where green is good, pink is bad, and yellow is neutral).

We may find the whiteboard method better for on-site collaboration with our team, whereas the spreadsheet may work better for remote collaboration (assuming we're using an online spreadsheet like Google Sheets or Office 365) or for cases where the results need to be portable and easily accessed later in the project.

### Adding findings to the tree spreadsheet

Another simple way of recording findings is to add a "Findings" column to the spreadsheet where we created our tree(s):

A	B	C	D	E	F	G
Level 0	Level 1	Level 2	Level 3	Level 4		analysis
About this tree:	Prioritised main tabs, plus less important topics under a More tab. Other minor regroupings and rewording. Landing pages have overviews.					TT score = 60%
MfE Home						
	Air					
		The quality of our air				
			Air quality research			
			Carbon monoxide			
			Dioxins, furans and PCBs			
			Greenhouse gases			Most went to CC, but confusion from there. Some went to Air, but confusion there too. ER is correct? More > ER was slim. Some leaks at L1.
			Hazardous air pollutants			
			Nitrogen dioxide			
			Ozone			
			Particles			
			Sulphur dioxide			Most went to Air. Quality and ER were strong. Quality - some confusion in subtopics. ER>Monitoring - correct? Minor leaks elsewhere.
		Clean Air programme				

This works well when the findings map directly to specific items in our trees. For findings that need action (see below), we may want to highlight these according (e.g. red cell background).

The benefit of this method is **context** – when we come back to revise this tree (or extract the best ideas from it for a new tree), we'll see the specific items that performed well or poorly.

The main drawback of this method is the same for hardcopy markup above – it makes it a bit harder to see the big picture across tasks and across different trees.

Finally, for more general findings, we may want to create an area at the bottom of the spreadsheet to document these:

Summary
Some topics need work - e.g. Air, CC, Waste, RMA
Anything important in ER should be cross-linked from other main-topic sections.
Hazards attracts a lot of semi-related clicks. May need some redirects.
The topics under More are definitely less visible, but the mega menus should improve this.
News & Events got consistent traffic when the task concerned something time-based.

---

**Next:** [Turning findings into actions](#)

## Turning findings into actions

- Obvious or not?
- Recording actions

Once we've recorded our findings, it's time to turn them into actions. These are usually revisions that we need to make to our site tree, but sometimes they go further - checking terminology with the Legal department, filling content gaps we've discovered, checking technical limitations with developers, and so on.

### Obvious or not?

Some findings are simple (e.g. "users don't understand the term *Contingency planning*") and their solution appears obvious ("rename it *Planning for emergencies*"), so it's pretty safe in these cases to just get on with it.

But **be careful of knee-jerk solutions**. Earlier we warned against jumping to solutions too soon, and this is good advice for any kind of user testing. It's best to do a first pass to write down all our findings before we start solving them, because we often need a wider view to understand what is really going on – what the real causes of the problems are.

We'll often find that we've jotted down solutions for seemingly easy findings, only to encounter some later tasks that either contradict or modify the earlier solutions. It's best to **think of all solutions as tentative until we've gone at least once through all the findings**.

### Recording actions

If we're marking up hardcopy, it's easy to jot down our solutions beside the related findings, and highlight them for later action (e.g. by circling or starring them).

If we're using a tree/task matrix (as described earlier in [Recording findings](#)), we can add another column for solutions across trees, or another row at the bottom for solutions across tasks:

		tree 1 (audience C)	tree 2 (actions)	across trees
#	<b>General tasks (not audience-specific)</b>			
1	You're not sure why your power costs so much. How would you find out what makes up a typical invoice?	MA got traffic (correct), then scattered. Resi got traffic, then scattered. About Us got minor traffic (correct).	Most went to YA (correct). Plans got traffic, then scattered. No one thought generally enough to go to WWD.	Anything related to a pragmatic task (e.g. bill breakdown) needs to be in on the main sections, or cross-linked from there.
2	You'd like to let Meridian know exactly how much electricity you used this month, so they don't send you an estimated bill. Find out how.	MA got traffic (correct). Resi got traffic (not sales), then scattered.	Most went to YA (correct). A few went to Login and FAQ (correct).	
3	Does Meridian supply natural gas?	About Us got traffic (correct), then scattered. Resi got traffic, then scattered.	No agreement on where this lives. WWD got traffic (reasonable).	About Us and WWD should also be clear on what we don't do.
4	You're looking for testimonials from other home-owners who use Meridian.	About Us got traffic, then scattered. Resi got traffic (correct).	Better got big traffic (correct). About Us got traffic. Some L1 leaks.	Should About Us include (or cross-link to) customer stories?
	You live in Tauranga and you're wondering how			

Similarly, if we're adding findings and solutions to our tree spreadsheets, another column does the trick for specific items, and more rows at the bottom can handle the more general actions we want to take.

**Next:** [Summing up the basics](#)

## Summing up the basics

- [Summary for a single tree](#)
- [Summary for several trees](#)

In most cases, we work with others – members of the project team, project owners and sponsors, clients, and vendors. While it's important that we (as designers and information architects) clearly understand our tree-test results, it's equally important that we **clearly communicate these results to our colleagues** and anyone else with a stake in the project.

If we're working in a fast-paced environment (e.g. in an Agile team or a small organization), the simplest and fastest way to communicate results is an **email summary of what we learned and what we do next**. In our experience, most stakeholders will not read a long-form report attachment (no matter how lovingly we assembled it), but they will read (or at least skim) a short email of bullet items. **Whether we end up writing a full report or not, sending an email summary first is always a good idea.**

The summary should include the bare minimum needed to communicate the most important results of the study. This normally is limited to:

- A sentence or two describing the study (mainly for those who were not close to it).
- Bullet lists (or a table) of concisely worded findings and actions
- A sentence or two outlining what happens next
- Links to fuller results (results from our online testing tools, more detailed findings, tree spreadsheets, etc.)
- An invitation for people with more questions/comments/suggestions to contact us.

Note that **this summary should be in the body of the email**, readable without having to open an attachment or click a link. We must make it as easy as possible for email skimmers (that means most of our audience) to get the gist of the results without any extra effort.

## Summary for a single tree

If we tested a single site tree, we may find it useful to group our findings by:

- Items/ideas that worked well
- Items/ideas that didn't work well
- What will happen next

*Subject: tree test of existing site - summary of results*

*Hi team,*

*This week, we ran an online tree test of our existing website, to see if site visitors could find typical items by browsing the headings and subheadings. 87 customers and 92 non-customers participated.*

### **What worked well**

- *My Account and About Us were clear, with all participants easily finding relevant items there and not visiting those sections otherwise.*
- *The Downloads page in the Support section was easily found by participants looking for our latest software patches.*

### **What needs work**

- *The separate Products and Solutions sections were continually confused by both user groups.*
- *Existing customers found the SuperWidget product page easily, but non-customers confused it with our Widget Packs and went to those pages instead.*
- *The What's New page (where we list our upcoming events) was not found by most participants looking for the next webinar.*
- *The FAQ page attracted heavy (and unwanted) traffic even for basic product information.*

### **Next steps**

- *Susan will run an online card sort next week to help decide how to reorganize the Products and Solutions content.*
- *Lloyd will investigate the FAQ content and see if we can merge it into the main content areas of the site.*

### **Full data and analysis**

- The tree test data and more detailed results are at <http://acme.com/studies/tree-test-2016-01-15>

As always, contact me if you have any questions, comments, suggestions, or ideas for future studies!

## Summary for several trees

If we tested several site trees, we may find it better to group our findings by tree, to keep from confusing our readers:

*Subject: tree tests - round 1 - summary of results*

*Hi team,*

*This week, we ran online tree tests of two new proposed structures for our website, to see if site visitors could find typical items by browsing the headings and subheadings.*

*We got a pretty good turn-out of existing customers and non-customers, and a **clear winner** to use as the skeleton of our site redesign:*

	<b>Tree 0 (existing site)</b>	<b>Tree 1 (task-based)</b>	<b>Tree 2 (audience-based)</b>
# of Customers	87	91	81
# of non-customers	92	75	65
Success rate	44%	61%	<b>74%</b>

### **What worked well**

<b>Tree 1 (task-based)</b>	<b>Tree 2 (audience-based)</b>
<i>My Account and About Us were clear, with all participants easily finding relevant items there and not visiting those sections otherwise.</i>	<i>My Account was clear.</i>
<i>The Downloads page in the Support section was found by most participants looking for our latest software patches.</i>	<i>The Downloads page (moved to the top level) was found by ALL participants.</i>

### **What needs work**

<b>Tree 1 (task-based)</b>	<b>Tree 2 (audience-based)</b>
<i>The separate Products and Solutions sections were still confused by both user groups, as they were in the original site.</i>	<i>The unified Solutions page performed well, attracting the right clicks from both user groups.</i>
<i>Non-customers confused the SuperWidget product page with our Widget Packs and went to those pages instead.</i>	<i>Moving the Pack pages under their corresponding products increased the success rate of the product tasks.</i>

### **Next steps**

- Susan will run a follow-up tree test combining a revision of Tree 2 with the best elements of Tree 1.
- Lloyd will start updating content for a Tree-2-based site.

### **Full data and analysis**

- The tree test data and more detailed results are at <http://acme.com/studies/tree-test-2016-01-15>

As always, contact me if you have any questions, comments, suggestions, or ideas for future studies!



---

**Next:** [Reporting in more depth](#)

## Reporting in more depth

- [Slide decks](#)
  - [Long-form documents](#)
- 

In many cases, the email summary described above may be enough for the project team to get on to the next steps in site design. In this (happy) case, we can move right on to [Chapter 14 - Revising and retesting](#).

However, there are some situations where we must produce a more formal, substantial report on our tree-test results:

- We are working for a client who wants a formal deliverable.
- We are presenting to upper management.
- Our results will be handed off to another team who we won't have a chance to brief.

In these cases, we will need to create a report that:

- Is clear to people who have not been involved in our study, and who may not know anything about tree testing (or even information architecture)
- Is detailed enough so that it can stand alone as documentation of our study.

### Slide decks

If we're *presenting* our results, a slide deck (using PowerPoint, Keynote, etc.) is the obvious way to go.

In our experience, slide decks may also be the best choice for reports that are distributed to stakeholders, because **people are often more willing to flip through slides** than they are to read a "document-style" report.

We also like slide decks because they **force us to fit our results on a single slide at a time**, which encourages graphics, simple tables, and bullets over long-form text. This can help stakeholders quickly understand the highlights instead of getting bogged down in details.

We typically include the following in our results slide deck:

- **Background**  
A slide or two about the study – why we ran it, when, how many participants, etc.
- **About tree testing**  
A slide or two about how tree testing works, from the participant's point of view, and later during analysis.
- **The tree(s)**  
We show an abbreviated summary of the tree(s) we tested (usually just the first and second levels) to give an idea of the various schemes we tried.
- **What we looked for**  
A short list of the questions we asked ourselves while creating the trees and writing the tasks, and the issues that could be resolved by testing.
- **Our findings and actions/recommendations**  
This is similar to what we sent earlier in our email summary, but we now have the room to add visuals (e.g. graphs from our tree-testing tool) and more details. Each task is typically a slide, with slides at the end for more general findings.
- **Next steps**  
What we plan to do next, when it will happen, who will be involved, and when we plan to be "done" with the site tree.
- **How to find out more**  
We wrap up by linking to more resources (e.g. the results in the tree-testing tool, our detailed working documents, etc.) and who to contact for more info.

Below is a PowerPoint deck with sample content. Feel free to customize it as needed:



## Long-form documents

As consultants doing IA work for corporate and government clients, we traditionally wrote a formal report as part of our deliverables. Originally, this was a long-form document created in Word, InDesign, or other “document” software, and usually delivered as a PDF.

While our deliverables these days tend to be lighter-weight (such as the email summaries and slide decks described above), some clients still do ask for the “full Monty” report.

For tree testing, long-form reports contain everything a slide deck would (see above), but are also likely to add:

- More detailed discussion of each finding, and more detailed rationale for recommended actions.
- Summaries of related activities (e.g. card sorts and content audits we did beforehand, other up-front customer research), to add context.
- Anything else we think the reader will need to understand the tree-test results at both high and low levels.

---

**Next:** [Passing along participant feedback](#)

## Passing along participant feedback

---

The main value we get from tree-test participants is the data we collect as they try to complete the tasks we've set. That helps us see where our trees are succeeding and failing, and how we can improve them.

A secondary benefit is that we're getting real users to engage with us, and this is a great opportunity to get a bit of extra feedback from them. Normally we do this by [adding a few survey questions to the study](#).

While some of these questions may be aimed at helping us analyze the results by group (e.g. asking their age, then checking to see if this affects their success rates), some questions may simply be there for gathering some knowledge about our audience. For example, we might have asked which region they live in, which may not make much difference to the tree test itself, but which may be interesting to the Marketing folks.

The most common kind of item to include in online studies like this is an **open-ended catch-all question** such as "Did you have any other feedback about our website?", where the user can type away with questions, comments, suggestions, rants (coherent and otherwise), and so on.

We're always surprised by how many participants take the time to type a response to this kind of question. It may be that site visitors are more likely to do an online study (such as a tree test) when they already have something they want to tell us.

In any case, **it's important that their feedback is collected, cleaned up, and forwarded to the right people in the organization**. Typically this involves:

1. Copying the feedback from the online tool to an editable medium (such as a spreadsheet)
2. Deleting obvious garbage responses such as "nothing I can think of" and phrases that don't have any clear meaning
3. Deleting anything that identifies a participant (unless they explicitly ask to be contacted about their feedback)
4. Grouping or tagging the responses according to an appropriate scheme (e.g. positive vs. negative comments, buying vs. billing vs. support, etc.)
5. Sending the various items to the right people in the organization (e.g. billing responses to the Billing department, support questions to Support, etc.), and asking them to reply directly to the participant where requested.

---

**Next:** [Chapter 13 - key points](#)

## Chapter 13 - key points

---

We need to translate our detailed analysis into **findings that colleagues can easily understand** (or that we ourselves can understand if we come back later).

To get the big picture, we should **do a pass to create findings before we move on to solutions**.

**Findings should be specific, and emerge from the data**, not our personal opinions.

For single-tree tests, **marking up the hardcopy** may be the easiest way to record our findings.

For multi-tree tests, **creating a tree/task matrix of findings** will help us see patterns more easily.

Some findings may contradict each other. Because of this, it's best to **mark our solutions as tentative** until we've gone at least once through all the findings.

In most cases, the quickest and easiest way to report our findings is a **brief email summary**, which is more likely to be read (or at least skimmed) than a long-form report attachment.

Need to produce more formal deliverables? **Download and customize our PowerPoint or Word templates**.

If we receive feedback from participants that is outside the scope of your study, **we should clean it up and pass it along** to other parts of the organization.

---

**Next:** [Chapter 14 - Revising and retesting](#)

## 14 - Revising and retesting

*"I may not have gone where I intended to go, but I think I have ended up where I needed to be." Douglas Adams*

Now that we've completed a round of tree testing, we should have lots to act on – revisions to trees that performed well, discarding of trees that just didn't work, identifying which elements to pursue and which to leave behind, tweaking of terms that participants had trouble with, and so on.

If we found a tree that tested extremely well, and only needs very minor tweaks, we may be "done" as far as tree testing is concerned. More likely, though, we'll want to make some major decisions and revisions, and those will need retesting to make sure our changes were the right ones.

Retesting also provides two big wins to the organization, beyond a better site tree:

- It shows that **the organization is willing to try out ideas**, have some of them fail early (before they reach the production site), and use that feedback to come up with something better. This "room to fail" is a key to real innovation.
- It lets the project team "**prove**" that their final design is a good one, not just by hand-waving, but by showing objective before-and-after data.

---

### Revising trees

Single tree vs. competing trees, keeping a record, no pandering

### Cherrypicking and hybrid trees

Selecting the best from each tree

### Rewording and replacing tasks

Fixing misunderstandings, retiring high-success tasks, updating answers

### Tuning survey questions

If we didn't get them quite right the first time

### Using fresh participants

Avoid reusing participants unless the pool is small

### When are we done?

70% success and no more substantial revisions

### Key points

---

**Next:** [Chapter 15 - Special considerations](#)

## Revising trees

- Results from a single tree
- Results from competing trees
- Keeping a record
- Updating correct answers accordingly
- Pandering to the task

### Results from a single tree

If we only tested a single tree in our first round, then there are two likely outcomes from that testing:

- **The tree did reasonably well and just needs refining.**  
This means that the tree's main organizing scheme performed reasonably well (indicated by an overall success rate around 60%), but was hurt by specific weaknesses that we think we can fix.
- **The tree performed poorly and needs major rethinking.**  
In this case, we decide that the tree's approach is just not the right one, and trying to revise it would just result in a lot of work to get to "mediocre". Best to try another tree (which means starting over). This is why we recommend testing several trees instead of one.

### Results from competing trees

If we tested several alternative trees (as we recommended in [The design phase: creating new trees](#) in Chapter 3), we're obviously keen to see which performed best (so we can pursue them) and which performed worst (so we can discard them).

If the overall success rates point to a clear winner, we should go with that tree and then revise it to be even better.

If there is more than one "winner", we can still **discard the poor trees and narrow our field**:

- Ideally we would revise the best trees, and pit them against each other in the next round.
- If we're tight on time or budget, we may have to pick one of the winning trees (based on other criteria such as effort to implement) and perhaps incorporate some of the elements that made the other winners perform well (see below).

### Keeping a record

As we select and discard trees and elements in them, and make changes to them, it's a good idea to keep a record of:

- **Previous versions of the trees**  
This is easily done by saving our old versions and making our changes to a new "current" copy of them. If we ever need to go back for an idea we previously discarded, it will be there waiting for us.
- **Decisions and our rationale for them**  
As we revise and reword individual items or entire sections of a tree, we can annotate them with our decisions and a brief rationale (where needed). We can also use color-coding or text styling to distinguish items that were deleted, changed, or are placeholders for future content.

In the example below, we've color-coded deleted item as red (some marked as "future" for later releases), and added rationale and instructions for other items:

Level 2	Level 3	Level 4	notes	comments
	Learning tools and resources		<i>move to here</i>	
		Māori resources		
		Pasifika resources		
		National Library resources		
	Special education		<i>bumped up a level</i>	
	Employment and pay		<i>Redirect to Run&gt;Employ (full list)</i>	
	Laptops for teachers		<i>from the teacher's PoV</i>	<i>future</i>
	All school forms		<i>future as needed</i>	
Running a school				
	Term dates and holidays		<i>removed as in the mega menu</i>	
	Enrolment and attendance			
		About ENROL	<i>have changed this title to reflect what is actually in the section</i>	
		National Student Number		
		<i>Paetoral rare for international students</i>		

### Updating correct answers accordingly

If we make any substantial revisions to our tree(s), we should re-test those revisions, to make sure our changes work.

As we make our revisions, we must also remember that these changes may affect the correct answers we've marked for our tasks.

- **If we change the tree, the location of correct answers may move** (for example, *FAQ* was under *Contact us*, but is now under *Support* ).  
Or, some correct answers may disappear (or new ones appear) because of our reshuffling of topics in the tree.
- **If we change a task, the correct answers for it may also change.**  
Minor tweaks of phrasing are unlikely to matter, but if we change the meaning or purpose of a task, we need to check that its marked answers are still correct and complete.

## Pandering to the task

When we analyze individual tasks, especially low-scoring ones, it's natural to want to fix the problems we discover. This usually means shuffling or rewording topics.

That's all well and good, but we need to make sure that we're **making a change that will help the tree perform better in real-world use, not just for this single task.**

If we're considering a change to our tree to fix a low-scoring task, we should make sure to:

- **Review other tasks that visit that part of the tree**, look at their results, and check that our change won't cause problems for them.
- **Review other reasons that users may visit that part of the site**; they may not have been important enough to become tasks in our study, but they should still be supported by the tree.

---

**Next:** [Cherrypicking and hybrid trees](#)



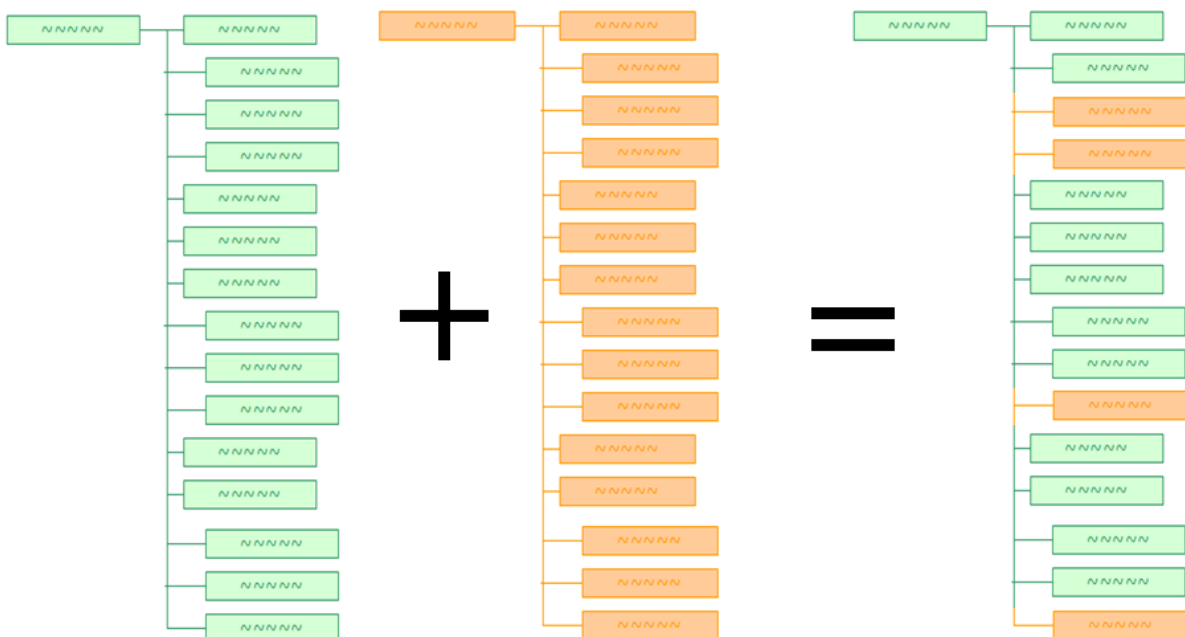
## Cherry-picking and hybrid trees

In our experience of testing several trees against each other, it's rare that one tree gets it all right, while the other trees get it all wrong. More often, we get one of the following situations:

- **One tree is the clear winner**, substantially outperforming the other trees. But there are a few elements of the tree that didn't work well, and need to be rethought. The other trees lost out, but our analysis may have identified **certain elements in those trees that actually performed better** than their counterparts in the winning tree.
- **No tree is the clear winner** – they all had problems that kept their overall scores down. Like the case above, though, we were able to identify certain parts of each tree that performed well and should be pursued if possible.

In both cases above, we end up **cherry-picking the best ideas from our candidate trees** and **creating a hybrid tree** from those pieces:

- If we have a winning tree, we keep its overall structure and try replacing its weak areas with the best ideas from the other trees:



- If we don't have a winner yet, we may need to create two or 3 hybrids to test in the second round, to see if we can come up with a winning structure.

When the New Zealand Ministry for the Environment redesigned their website, that's exactly what they did. They tested the existing tree, then developed 2 new ideas for their site structure. While both of these ideas tested better than the original site, their final tree was actually a hybrid of the two ideas:

## Idea 1

- **Topics A-Z**
  - Air
  - Biodiversity
  - Climate change
  - Energy
  - Environmental reporting
  - Fresh water
  - Hazards & risks
  - etc.
- **Publications**
- **Legislation**
- **Funding & awards**
- **News & events**
- **About us**
- **etc.**

## Idea 2

- **Air**
- **Water**
- **Land**
- **Waste**
- **Hazards**
- **Climate change**
- **RMA**
- **More topics...**
  - Acts & standards
  - Awards
  - etc.
- **Publications**
- **News & events**
- **etc.**

## Final tree

- **Air**
- **Fresh water**
- **Marine**
- **Land**
- **Waste**
- **Hazards**
- **Climate change**
- **RMA**
- **More topics...**
  - Acts & standards
  - Awards
  - etc.
- **Publications**
- **News & events**
- **etc.**

Note that this is not always possible, because some ideas are just not compatible with others. If tree A won the first round, but there are parts of tree B that did really better than their counterparts in tree A, it may not just be an easy copy-and-paste from B to A, because what worked in B may not work with the different approach that tree A takes. Which is why we heartily recommend retesting the hybrid tree to make sure everything works as intended.

---

**Next:** [Rewording and replacing tasks](#)

## Rewording and replacing tasks

- Fixing misunderstood tasks
  - Tasks with very high success rates
  - Updating correct answers accordingly
- 

It's not just the tree that may need revising after we get our first round of results. Sometimes the tasks themselves need rework.

### Fixing misunderstood tasks

Even though we work hard to write clear and unambiguous tasks, the results sometimes make it plain that a task was not clearly understood (or was clearly misunderstood) by our participants. Because most tree testing is done unmoderated, participants can't ask for clarification when they have trouble understanding a task.

For example, when we helped a bank run a tree test on their self-service website, we included this task:

| *How would you get automatically notified when your cheque-account balance goes below \$100?*

In the results, we saw that:

- Some participants went to the *Account Status* section to see if there were any alerts.
- Other participants went to the *Preferences* section to set up a low-balance alert.

When we re-read the task, we saw that it could have interpreted as either getting the alert, or setting it up. (We actually meant the latter.)

If we're doing another round of testing, **we should revise these murky tasks so we can get higher-quality results**. This makes before-and-after comparisons harder, but comparisons are less important than clear primary findings about the tree.

In the example above, we clarified the task wording so that we could judge the results better:

| *How would you set up an automatic notification for when your balance goes below \$100?*

### Tasks with very high success rates

**We should also reconsider tasks that (almost) everyone got right.** It's great that participants easily found what they were looking for (after all, that's our Big Goal), but first let's make sure those tasks weren't "gimme's". Did we give away the answer by careless word matching, or by phrasing the question in the same browsing sequence that the participant would follow in the tree? (For more on these pitfalls, see [Writing a good task](#) in Chapter 7.)

Even if the task passes these tests, we still may want to replace it in the next round. Why? Because we won't learn much from it in later testing. **It works, and we should move on.**

This is especially true if we find that another part of the tree is either not performing well or is not being tested enough; it may be better to replace our "golden" task with one that tells us more about what needs improving.

Note that there are **two major downsides with replacing high-scoring tasks** in a later round of testing:

- **The overall success rate may go down.**  
If we replace a task that scored 90%, the task that replaced it is unlikely to score that high, so our average score will go down. This makes the results harder to explain to the project team and management.
- **Before/after comparisons will be harder to make.**  
When we keep the tasks the same between tests, we can make apples-to-apples comparisons. If we start replacing tasks, we can no longer make broad comparisons across tests.

If these factors are important to our study, we may want to avoid replacing high-scoring tasks. If they're not so important (that is, if our need for specific answers to specific questions outweighs these broader considerations), then replacing high-scoring tasks becomes a way of re-focusing the study on the parts of the tree that need more testing.

## Updating correct answers accordingly

If we revise tasks, we must also remember to check the correct answers for those tasks. Our revisions may have changed what we should accept as correct. This is especially true if our revisions were not just minor rewording, but actually changed the meaning or purpose of the task itself.

For more, see [Identifying correct answers](#) in Chapter 7.

---

**Next:** [Tuning survey questions](#)

## Tuning survey questions

---

In addition to tasks that need revising, we may find that some of our survey questions didn't quite hit the mark the first time around.

For example, we may have given participants a multiple-choice question that didn't include the answers they needed, so we got a lot of "other" responses. For example:

*When you first got interested in electric vehicles, where did you go for information?*

*48 - Google or other web searches*

*8 - Facebook groups*

*5 - Car manufacturers/dealers*

*46 - Other (please describe)*

In this example, half of our participants chose "Other", and many of them wrote in the same answers ("my friend who owns an EV", "government transportation website", etc.). We realised that we had missed several popular answers in our initial question.

This was a useful result, because not only did we learn something for our research, but we were also able to improve this question in our next round of testing, turning that big blob of "other" into more useful specifics:

*When you first got interested in electric vehicles, where did you go for information?*

*- Google or other web searches*

*- Facebook groups*

*- Car manufacturers/dealers*

*- **Government websites***

*- **Word of mouth (neighbors, friends, colleagues, etc.)***

*- Other (please describe)*

Again, by revising our questions between rounds, we lose some of our ability to compare the results against each other, but given the choice of analyzing two sets of cloudy data vs. 1 set of clear data, we'll always take the latter. 😊

---

**Next:** [Using fresh participants](#)

## Using fresh participants

---

For subsequent rounds of testing, a common question is “Do we need to get fresh participants, or should we use people who have already done an earlier test?”

The answer here is the same as for most other forms of user testing: **ideally, we prefer fresh participants** because they haven't seen the tree (in an earlier revision) or the tasks before, so they should be untainted (so to speak).

If we have a large audience to draw from, we can certainly do some screening to make sure we're not accepting people who participated in an earlier round of tree testing. (For more on screening, see [Screening for specific participants](#) in Chapter 9.)

In the real world, however, we often do not have a huge pool for testing, so we often forego this screening just to get the minimum numbers we need to produce clear results. We think this is acceptable because the “experienced” participants don't really have a big advantage over the fresh participants – the tree may have changed substantially since the last test, the tasks may have been revised, and enough time has probably passed (say, a week or two) that their recall is only partial.

If we're concerned about re-using participants, we can also add a survey question to the second test, asking if they participated in the first test. Later, during analysis, we can filter the results to see if the “veterans” did significantly better than the first-timers. We suspect the difference (if any) would be small enough to ignore.

---

**Next:** [When are we done?](#)

## When are we done?

---

In [The design phase: creating new trees](#) in Chapter 3, we strongly recommended doing more than one round of tree testing to get a site tree performing well:

- **Two rounds is very common** – round 1 for the original tree and the proposed replacement, and round 2 to test revisions to make sure our changes actually solve the problems we found.
- **Three rounds is ideal** – round 1 for the original tree (along with an open card sort to generate ideas), round 2 for a few alternative trees to compete against each other, then round 3 to test revisions to the tree that won out in round 2.
- If parts of our chosen tree are still not performing well, we can easily do further rounds to keep improving our results.

At some point, however, we need to be done:

- In the best case, we're done when our tree is performing well (typically a **success rate of 70% or more**) and there are **no more useful revisions we can make** to the tree – or at least, no revisions that are going to make a real difference to our tree-test results. We may still tweak the tree in later stages of the project (e.g. after usability testing), but for now, we have a solid foundation to build on.
- If time or budget (or motivation) run out before we achieve a high-performing tree, that's not optimal, but it's still **much better than doing no testing at all**. We have made substantial revisions, and we've identified where the trouble spots remain, so we can keep an eye on them (through usability testing and analytics) and fix them later when circumstances allow.

---

**Next:** [Chapter 14 - key points](#)

## Chapter 14 - key points

---

**Most trees will need revisions after testing.** If these are more than minor tweaks, they should be re-tested to confirm that they fix the observed problems.

**Re-testing** results in a better structure, shows that the organization is willing to try out ideas, and lets the project team “prove” that their final design is better than previous ones.

In general, we **discard trees (or elements) that didn't perform well, and pursue those that did**, documenting our rationale as we go.

**Testing alternative trees** makes this iterative approach much more powerful.

When re-testing, we should also **revise tasks** that were not clear in the first round, or which needs rewording (or different correct answers) because of our tree changes.

We ideally want **fresh participants for the follow-up test**, but may have to settle for previous participants if our pool is small.

**We're “done”** when our success rate is high (>70%) and there are no more useful revisions we can make to the tree. But if time or budget run out before this point, we can console ourselves with the fact that **some testing is better than none at all**.

---

**Next:** [Chapter 15 - Special considerations](#)



## 15 - Special considerations

*Notice that the stiffest tree is most easily cracked, while the bamboo or willow survives by bending with the wind. - Bruce Lee*

Going slightly off the beaten path? These topics may help you adapt tree testing to your particular needs.

---

### Testing trees for mobile apps

summary text here

### Tree testing on paper

summary text here

### Multi-language testing

summary text here

### Key points

summary text here

### Tree testing in Agile/Lean projects

summary text here

---

**Next:** Chapter 16 - Beyond tree testing

## Testing trees for mobile apps

---

need content

---

**Next:** [Multi-language testing](#)

## Multi-language testing

---

- [separate tests?](#)
- [How to translate?](#)
- [Cultural differences?](#)
- [who to ask about this?](#)
- <http://digitalworkplacegroup.com/2014/07/14/intranet-navigation-tree-testing-7-tips/>

---

**Next:** Tree testing in Agile/Lean projects

## Tree testing in Agile/Lean projects

---

need content

---

**Next:** [Tree testing on paper](#)

## Tree testing on paper

---

The tree-testing method was created by well-known information architect [Donna Spencer](#) while at [Step Two Designs](#) and the [Australian Bureau of Statistics](#). She ran tests manually, using index cards for tree branches and tasks, and compiled the results manually in a spreadsheet.

- For a summary of the original technique, see her [Boxes and Arrows](#) article.
- For more details about running a tree test on paper, see Donna's [Testing Information Architecture](#) article at [UX Mastery](#).

---

**Next:** [Chapter 15 - key points](#)

## Chapter 15 - key points

---

need content

---

**Next:** [Chapter 16 - Beyond tree testing](#)

## 16 - Beyond tree testing

| *"To infinity, and beyond!" - Buzz Lightyear*

Tree testing is a narrow, focused method; it only tests the **grouping of headings** in your site, and their **labelling**.

But there is more to Information Architecture than grouping and labelling. We must also consider **navigation**, **search**, **visual design**, and the **content** itself.

In this chapter, we briefly discuss how we can test these remaining IA elements.

---

### Testing navigation

Menu testing and first-click testing

### Testing search

summary text here

### Testing visual design

summary text here

### Testing content

summary text here

### Testing overall usability

summary text here

### Key points

---

## Testing navigation

- [Menu testing](#)
  - [First-click testing](#)
- 

It's easy to confuse tree testing with navigation testing; both are dealing with how users move through a site by clicking on headings and menus.

But there's an important difference: tree testing evaluates the effectiveness of grouping and labelling, but without any spatial design or interaction design - just a plain drill-down textual tree.

**Navigation testing**, on the other hand, presents an actual (or simulated) user interface to our participants, which may include things like working menu bars, visually designed controls, and so on. In this case, we're testing not just the grouping and labelling, but also the menu design (in menu testing, below) or visual layout (in first-click testing).

### Menu testing

**Menu testing** is similar to tree testing, in that we are typically modeling a top-down structure and using representative tasks to test its effectiveness with users.

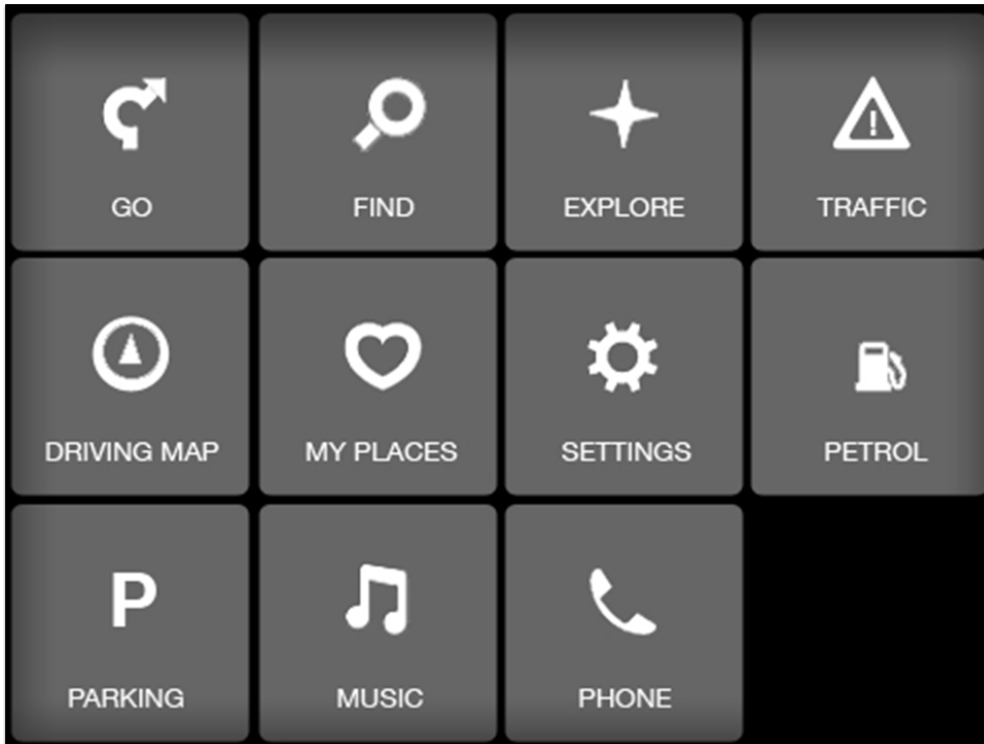
[Naview](#) is a menu-testing tool that lets us "**design and build navigation prototypes** and **test navigation usability**".

### First-click testing

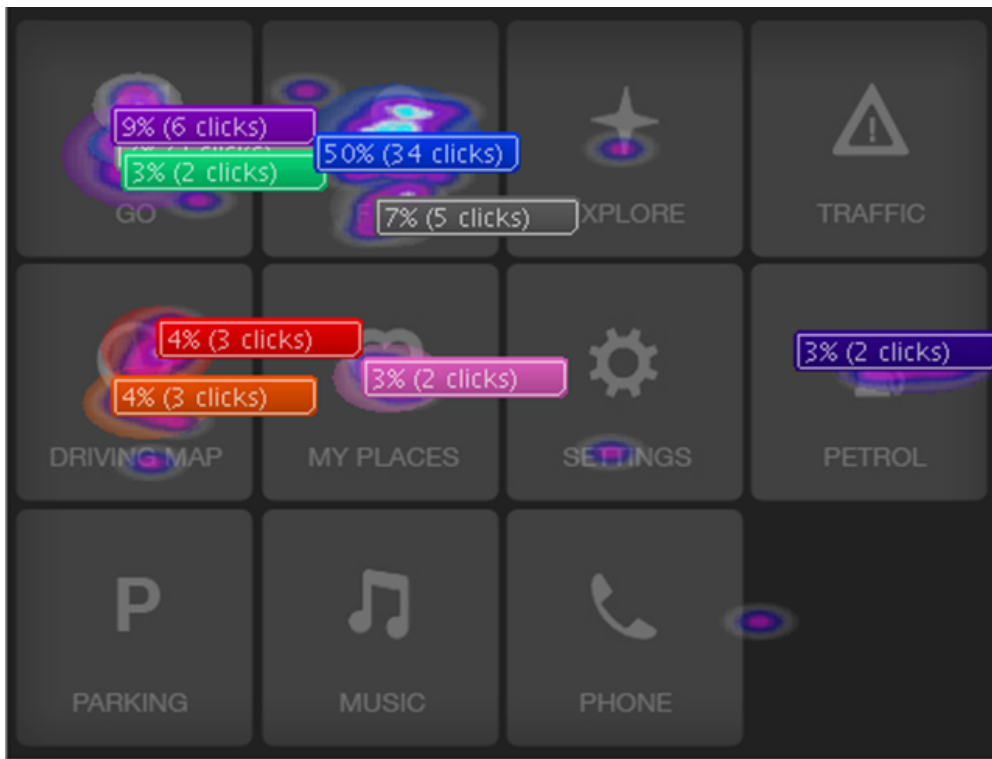
**First-click testing** (sometimes just called "click testing") is a popular way to see where users click, given a screenshot and a specific task.

For example, here's a prototype sat-nav UI that underwent first-click testing:





When we asked participants to "visit friends at their new house on 23 Napier St for the first time", the "heat map" generated by the testing tool showed us where they clicked:



This is a quick way of testing various navigation elements - navigation bars, utility links, featured items, links in body content, and so on.

However, because we're just capturing a single click on a screenshot, we can't easily test more complicated interactions like multi-level menus.

### Some first-click testing tools

- [Chalkmark](#) by Optimal Workshop
- [Click Test](#) and [Navigation Test](#) by UsabilityHub
- [Screenshot testing](#) by UserZoom
- [Usaura](#)

---

**Next:** [Testing search](#)

## Testing search

---

e.g. meta data, breadcrumbs, page titles vs nav titles, etc.

---

**Next:** [Testing visual design](#)

## Testing visual design

- [First-click testing](#)
  - [Multi-click testing](#)
- 

To test visual design (page layout, color, graphics, and so on), we typically use **tools that test an actual working site** with production graphics or **tools that test a screenshot** of a given page (often mocked up in something like Photoshop).

### First-click testing

If we want to find out if a particular placement of a control (or its size or shape or color) is effective or not, we can use the first-click tools described in [Testing navigation](#).

### Multi-click testing

If we're interested in testing more complex interactions (up to and including actual use of the working site), then we can use the "full-on" usability testing tools described in [Testing overall usability](#).

---

**Next:** [Testing content](#)

## Testing content

---

need content

---

**Next:** [Testing overall usability](#)

## Testing overall usability

---

need content

---

**Next:** [Chapter 16 - key points](#)

## Chapter 16 - key points

---

need content

---

## List of all templates

Template	Format	Description	Explained in...
<a href="#">tree test - planning questions v3.doc</a>	MS Word	A list of questions to ask when first scoping and planning a tree-test study	<a href="#">Documenting our plan</a>
<a href="#">Tree test - checklist.xls</a>	MS Excel	A detailed checklist of everything we do in a tree-testing project	<a href="#">Documenting our plan</a>
<a href="#">sample email invitation.docx</a>	MS Word	An effective email invitation for recruiting participants	<a href="#">Writing a good invitation</a>
<a href="#">sample explanation page.docx</a>	MS Word	A typical explanation page to explains the study and links to it	<a href="#">Using web ads</a>
<a href="#">sample terms and conditions.docx</a>	MS Word	A template for the terms & conditions of a study with a prize draw	<a href="#">Writing supporting text</a>
Interim update email	text	An email template for keeping stakeholders informed about the progress of the study	<a href="#">Keeping stakeholders informed</a>
<a href="#">audiences and channels.xlsx</a>	MS Excel	A spreadsheet for tracking which media channels we'll use to recruit which audiences	<a href="#">Coordinating audiences and channels</a>
<a href="#">sample findings matrix.xlsx</a>	MS Excel	A spreadsheet for recording findings across tasks and across trees	<a href="#">Recording findings</a>
<a href="#">sample report deck.pptx</a>	PowerPoint	A slide deck with sample content for reporting results	<a href="#">Reporting in more depth</a>



## About this guide

This guide is about **tree testing**, described by [Wikipedia](#) as:

| *a usability technique for evaluating the findability of topics in a website*

Tree testing originated as a [paper-based method](#) created by [Donna Spencer](#). It is now mostly done online using [commercial tools](#) such as **Treejack** and **UserZoom**.

This guide is based on lessons learned from hundreds of tree tests and dozens of Information Architecture (IA) projects run for small, medium, and large organizations.

Originally conceived as a book, *Tree Testing for Websites* has been released as a **community wiki** so that:

- It can reach the widest audience possible (by Internet searching and word-of-mouth linking).
- The community can update it to match the latest developments in IA methods and tree-testing software.
- Readers can comment on articles (no login required) and even add their own content (get a login below).

## Contributing to this guide

Have some hard-earned wisdom to share with the tree-testing community? [Contact us](#) to get a login so you can contribute to this wiki.

## Acknowledgements

Thanks to [Andrew Mayfield](#) and the folks at [Optimal Workshop](#) for the idea of writing a book about tree testing, and for supporting it along the way.

Thanks to **Donna Spencer** for her IA wisdom and early outlines for this guide.

## Credits

Emoji provided free by [Emoji One](#)