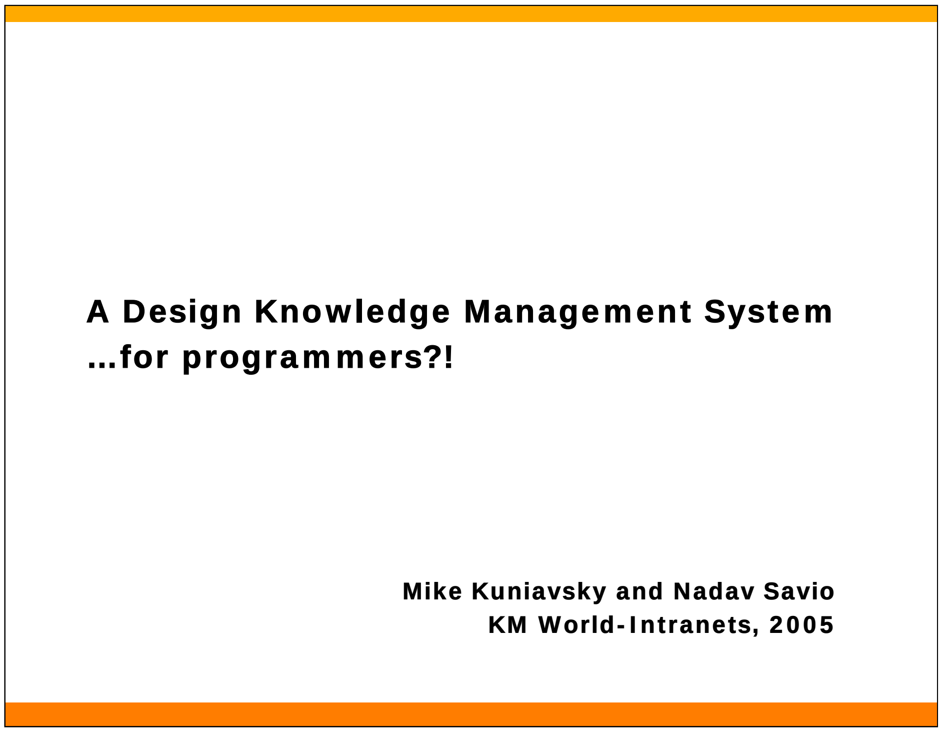
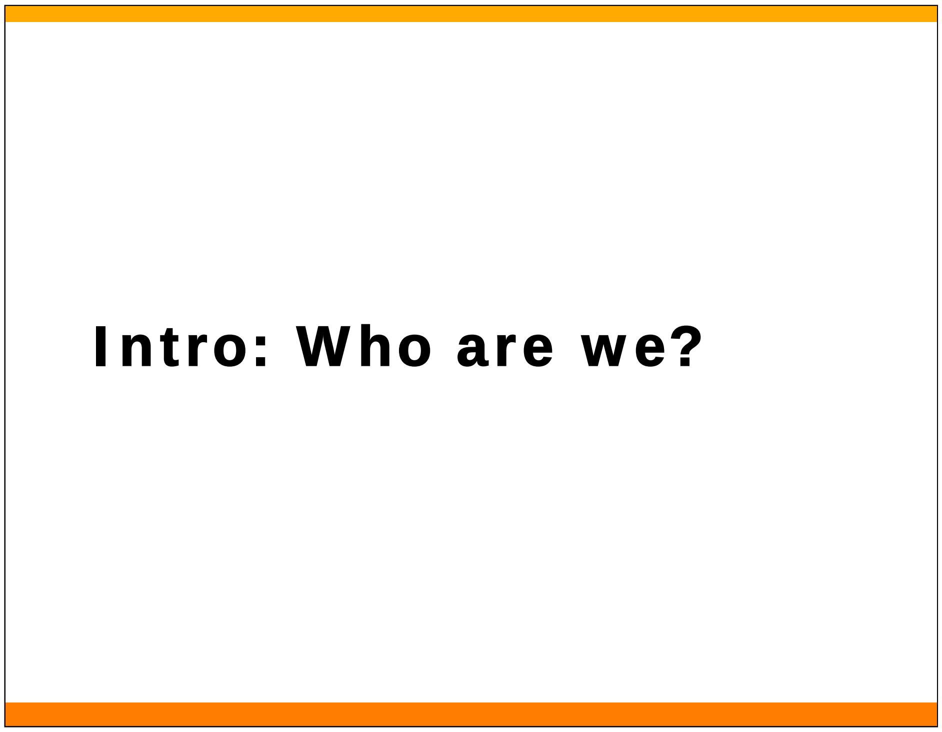


# A Design Knowledge Management System …for programmers?!

Mike Kuniavsky and Nadav Savio
KM World-Intranets, 2005



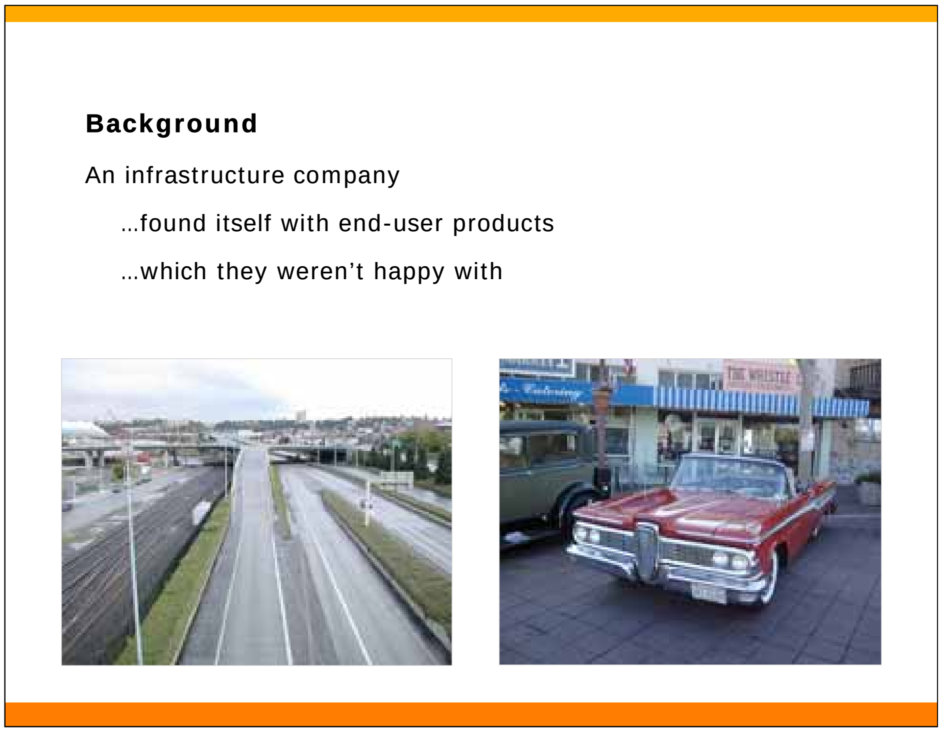
## Intro: Who are we?

Hi, I'm Mike Kuniavsky, I'm a user experience designer and consultant specializing in the strategic introduction of user-centered methods into organizations.

I'm Nadav Savio, I'm an interface designer and co-founder of the research and design firm Giant

Ant.

Today we’re going to talk about how we went about solving a problem, but first some background.



## Background

An infrastructure company

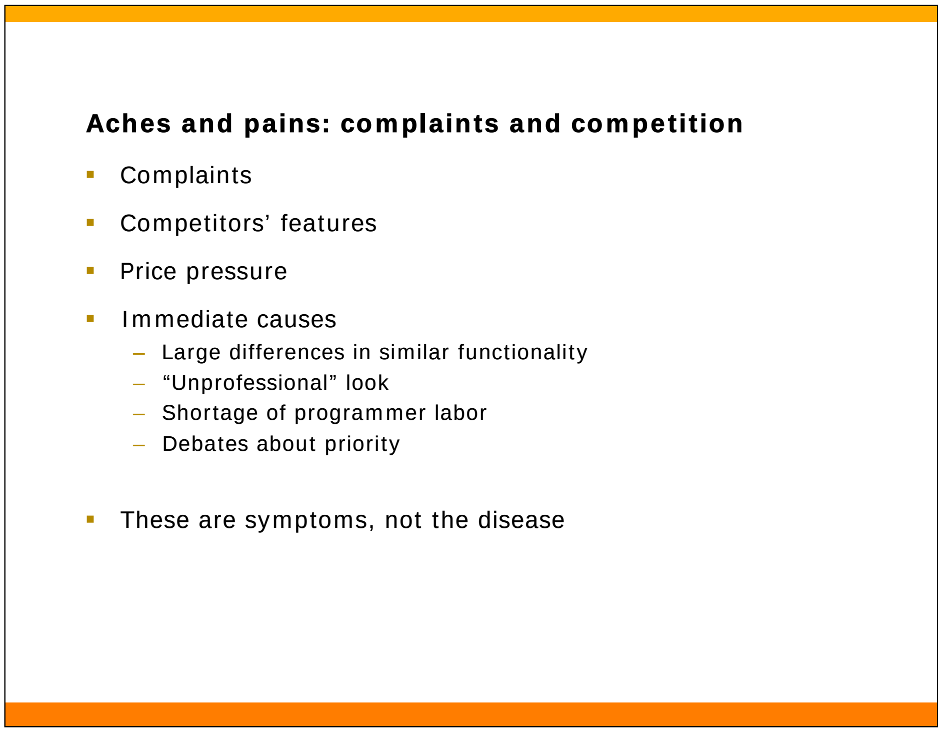
…found itself with end-user products

…which they weren’t happy with

About a year ago, a Fortune 500 company that traditionally makes communication infrastructure products approached me with a problem. For a number reasons, they found themselves making and selling client software in addition to just the infrastructure they had traditionally sold. However, they weren’t happy with these new products, and they called me to help their development team.

I started by asking them where they felt their problems lay.

Photos by: romulusnr, Morven, Flickr



## Aches and pains: complaints and competition

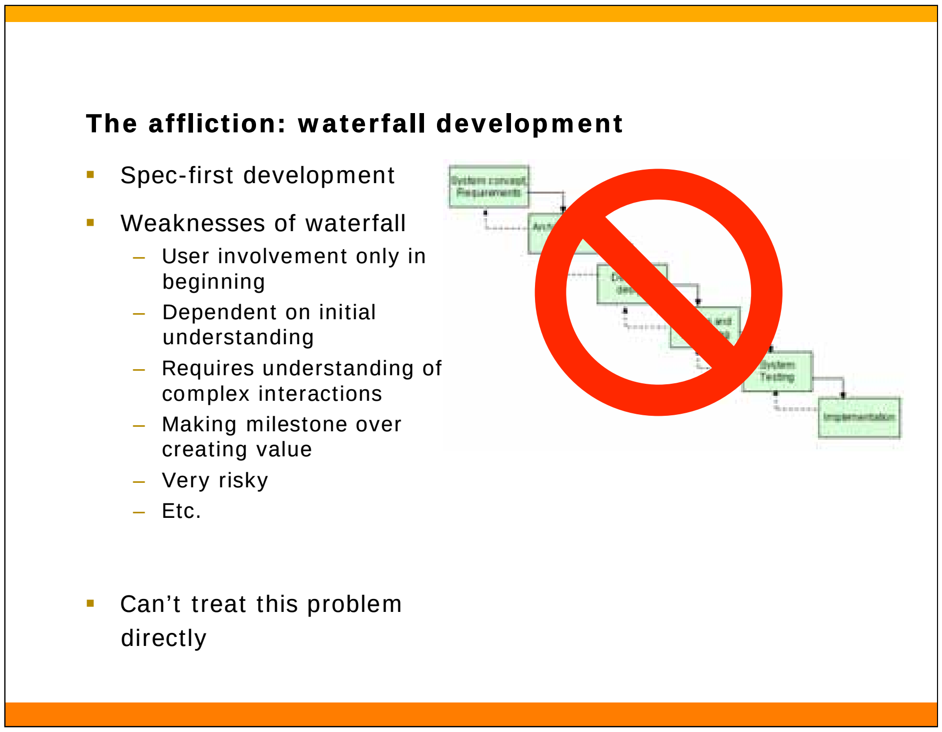
- Complaints
- Competitors’ features
- Price pressure
- Immediate causes
  - Large differences in similar functionality
  - “Unprofessional” look
  - Shortage of programmer labor
  - Debates about priority
- These are symptoms, not the disease

The problems they identified were: customer and management complaints about product functionality, that they were unable to deliver features that their competitors have because they’re addressing these complaints, and felt pressure from large customers to lower their prices because they could not deliver competitive features. A vicious cycle.

In addition, they cited big differences between applications in terms of how similar data was displayed or manipulated, a look that’s not up the standards of the rest of their products, they felt they didn’t have enough programmers to work on all of these problems, and they were constantly locked into debate about functionality.

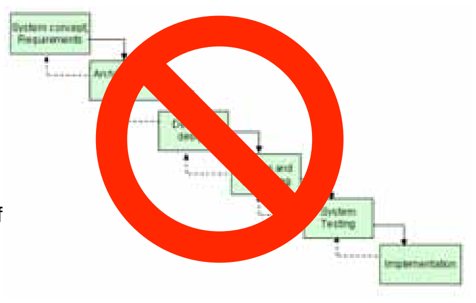
The sum total was that the applications were difficult to use, difficult to build and made them look bad.

However, these are all just symptoms of a deeper problem…



## The affliction: waterfall development

- Spec-first development
- Weaknesses of waterfall
  - User involvement only in beginning
  - Dependent on initial understanding
  - Requires understanding of complex interactions
  - Making milestone over creating value
  - Very risky
  - Etc.

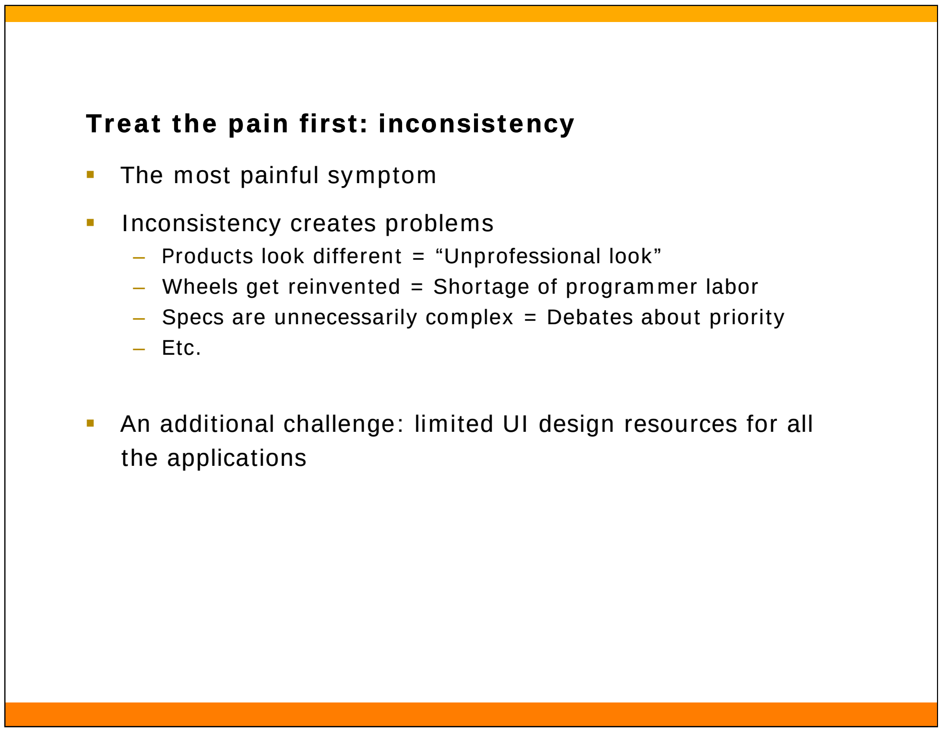


- Can't treat this problem directly

Basically, the core problem was they were stuck in a waterfall model of development. Extensive (600 page) specifications determine everything down to the labels on the buttons before any code is written.  Users are only involved in the creation of the specifications.

I believe that the waterfall model is fundamentally broken, and there are lots of problems, but it's a hard habit to quit: it makes everything seem like it's going great and the project is meeting its milestones, when in in fact it's going badly in all ways other than the Gantt chart.  This is, however, a deeply rooted methodology and can't be changed instantly.

Our decision was that in order to treat the deep problem, we needed to start with something simpler…



## Treat the pain first: inconsistency

- The most painful symptom
- Inconsistency creates problems
  - Products look different = “Unprofessional look”
  - Wheels get reinvented = Shortage of programmer labor
  - Specs are unnecessarily complex = Debates about priority
  - Etc.
- An additional challenge: limited UI design resources for all the applications

We decided to stabilize the situation so that the improvement can be introduced incrementally and consistently.

In addition, the company had very few UI designers available to design all of the applications, and that wasn’t going to change.

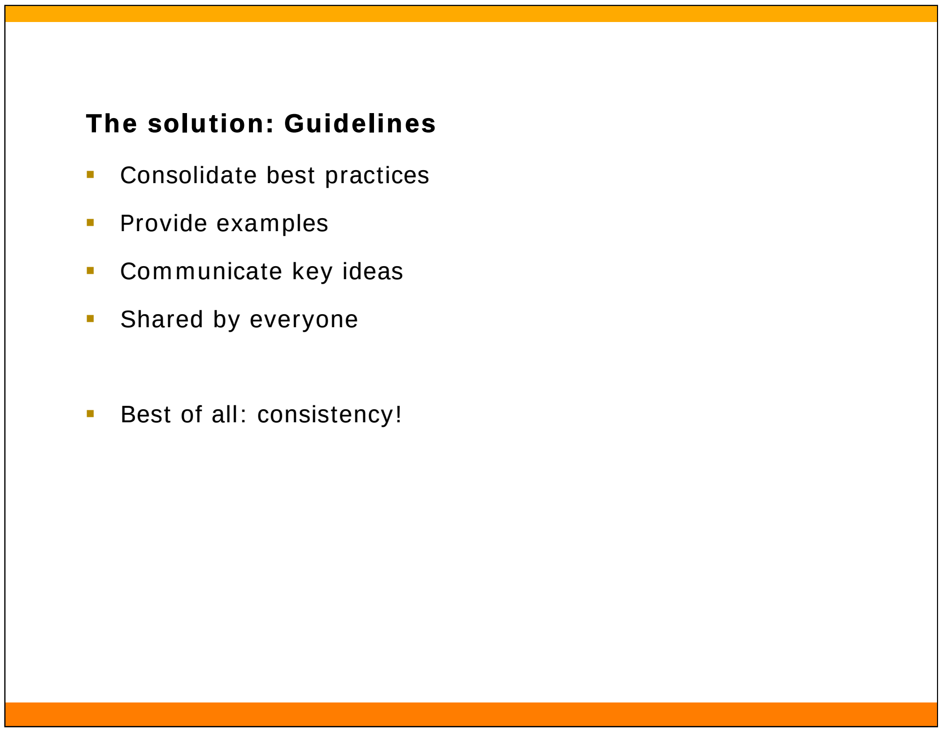
The led us to the conclusion that several hundred engineers in the company were going to have to be doing nearly all of the UI design.

## Creating consistency: Programmers as designers?!?

The “engineer vs. designer” myth

I know you’re thinking that designers and programmers are from two different tribes, but that’s not true. Programmers aren’t necessarily bad UI designers, just as they aren’t (necessarily) bad shortstops or chefs.  It’s just not what they’re trained to do.

Without formally training all of the developers, we needed to come up with some way to create consistency in the UI designs they were creating.



## The solution: Guidelines

- Consolidate best practices
- Provide examples
- Communicate key ideas
- Shared by everyone

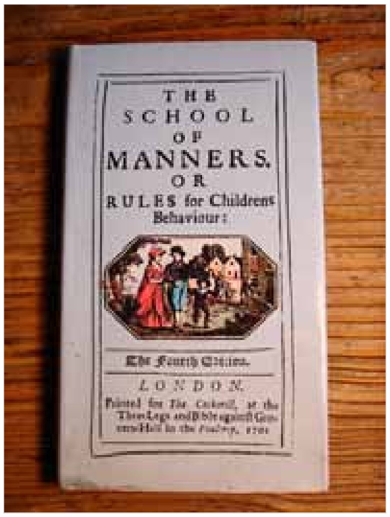
- Best of all: consistency!

So here was the big idea: guidelines.  Guidelines provide all kinds of good things.

But there’s a problem…

## The problem with guidelines

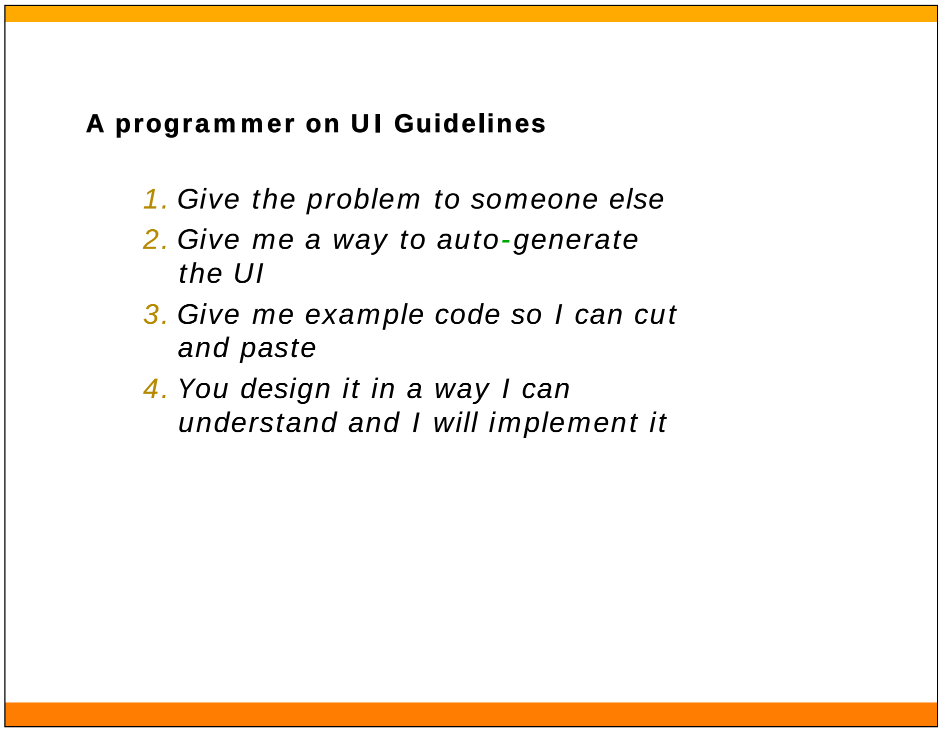
- People's don't use guidelines. Why?
  - Presented as "rules"
  - Rules = extra work
  - Not written for audience
  - Long
  - Lecture rather than solve problems
  - Little incentive to comply
  - Laws in disguise?
  - One way communication
- Good guidelines aren't enough
- They're not designed for their primary audience: developers



People don't use guidelines.

[read guideline problems]

In other words, if we think of guidelines as a product, they're generally missing the needs of their audience. Programmers aren't interested in extra work. To make sure that we were designing guidelines for developers, we talked to them



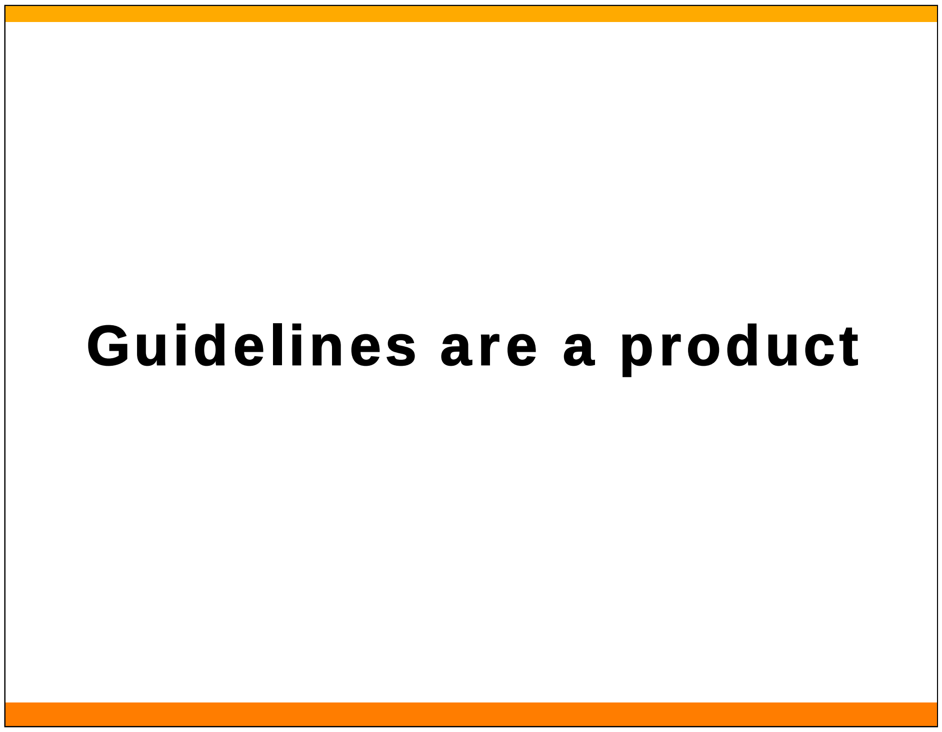
**A programmer on UI Guidelines**

1. *Give the problem to someone else*
2. *Give me a way to auto-generate the UI*
3. *Give me example code so I can cut and paste*
4. *You design it in a way I can understand and I will implement it*

Here's how one programmer put it when I asked him about how he'd like to use guidelines.

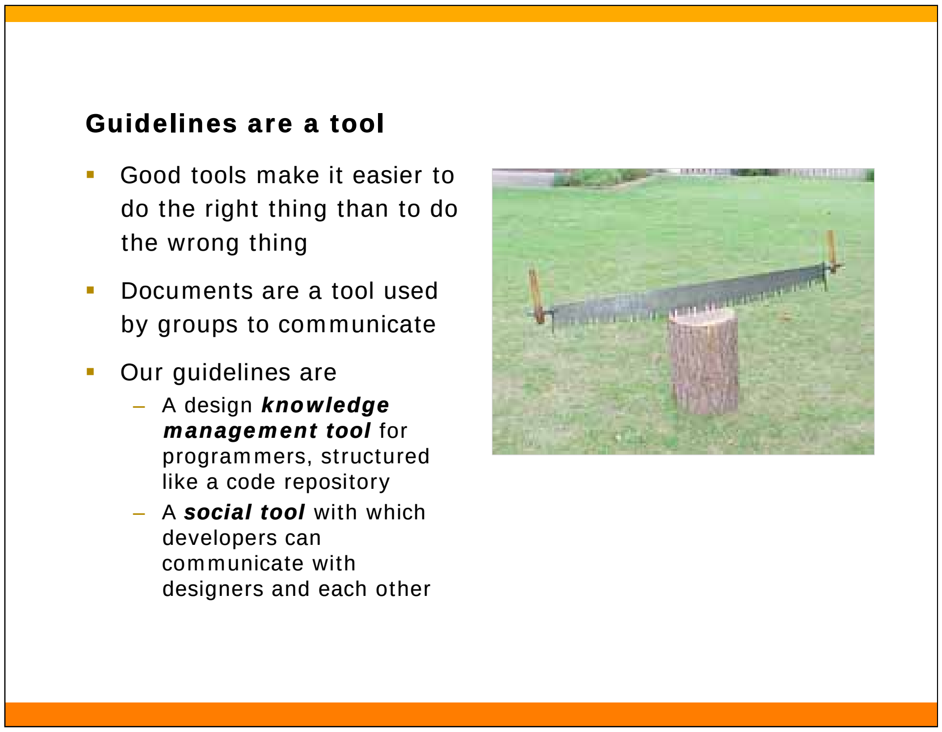
We didn't have the options to do number 1 (no resources) or 2 (magic), so we decided to tackle a combination of 3 and 4.

Having made that decision, we made another decision that we felt was pretty important…



## Guidelines are a product

Taking this into account, we decided to treat the guidelines like a product, and approach it exactly like we were developing a piece of software, with users and competitors.



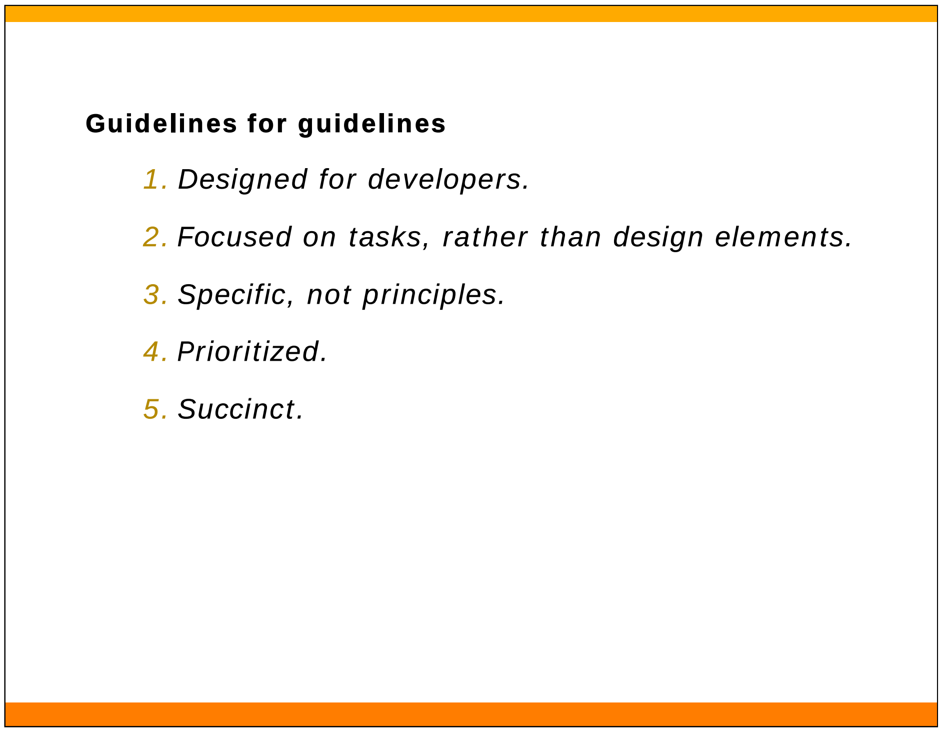
## Guidelines are a tool

- Good tools make it easier to do the right thing than to do the wrong thing
- Documents are a tool used by groups to communicate
- Our guidelines are
  - A design ***knowledge management tool*** for programmers, structured like a code repository
  - A ***social tool*** with which developers can communicate with designers and each other

Approached this way, guidelines are a tool

Tools embody a set of constraints with them.  Structuring the guidelines as a tool with the right constraints, would inherently make it easier to follow the guidelines.

They're also a communication tool.  Normally, they're a one way communication tool, but there's no reason they can't be a two-way communication tool.

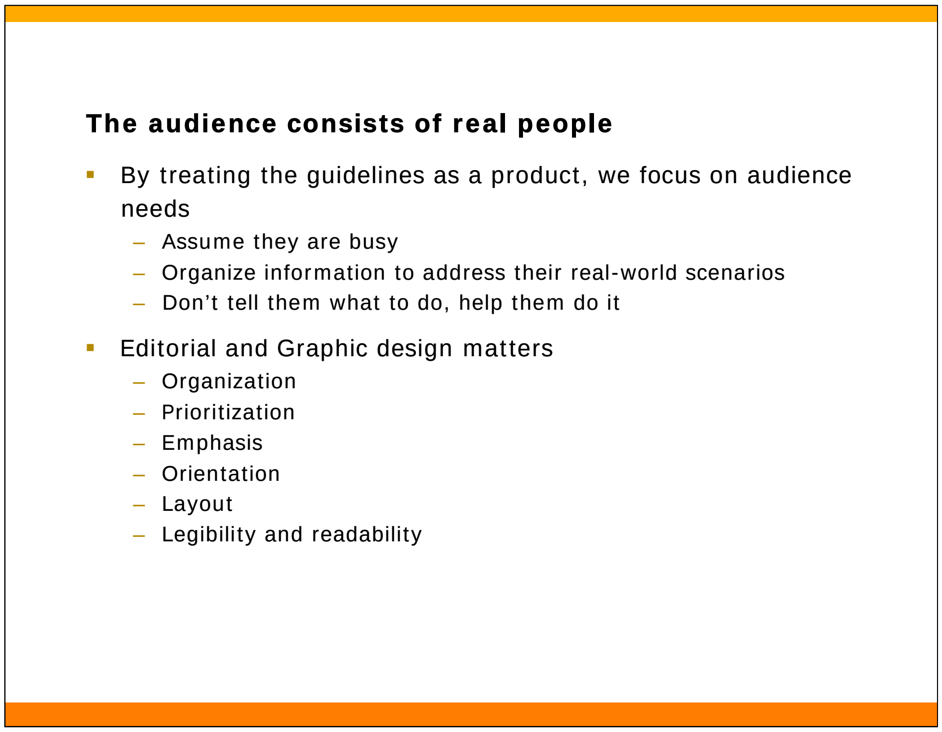


**Guidelines for guidelines**

1. *Designed for developers.*
2. *Focused on tasks, rather than design elements.*
3. *Specific, not principles.*
4. *Prioritized.*
5. *Succinct.*

What kind of product is it?

Based on user and competitive research we produced a set of Guidelines for Guidelines. 1. The structure of a tool for them should be built around programmer needs. 2. *Focused on tasks*. *3.* specifics address developers' needs better than theories. 4. *Prioritized:* explicit prioritization helps make choices. Not all guidelines have equal impact. 5. *Succinct: programmers are always in a hurry.* Extra words won't get read and diagrams will not get examined.



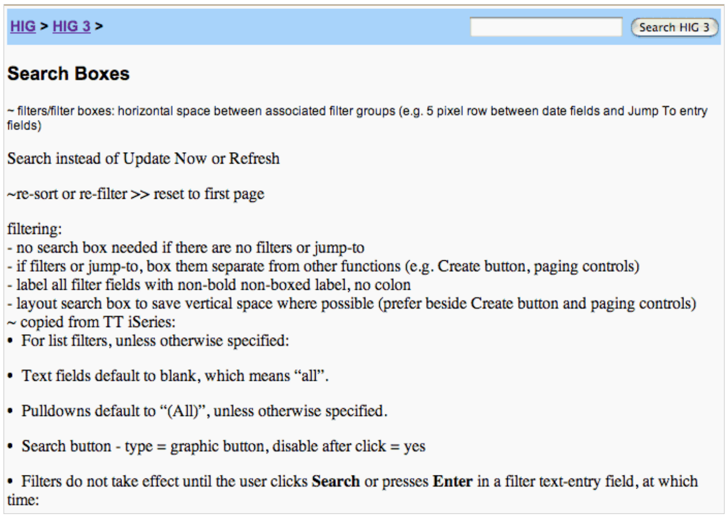
## The audience consists of real people

- By treating the guidelines as a product, we focus on audience needs
  - Assume they are busy
  - Organize information to address their real-world scenarios
  - Don’t tell them what to do, help them do it
- Editorial and Graphic design matters
  - Organization
  - Prioritization
  - Emphasis
  - Orientation
  - Layout
  - Legibility and readability

How does that play out in actually designing a guidelines document?

When creating documentation and just trying to get it all down on paper, it’s easy to forget that your goal is to help your audience implement the guidelines you are writing. The antidote is to remember that there is a real human being on the other side of the document. They want to do the right thing, but they are also busy and they were probably just in the middle of doing something else. By keeping this in mind, you move beyond merely documenting a system to supporting the system’s developers, with the result that your guidelines will be used and not just filed away on a shelf next to the Employee Handbook.

## Before



HIG > HIG 3 >

Search HIG 3

**Search Boxes**

~ filters/filter boxes: horizontal space between associated filter groups (e.g. 5 pixel row between date fields and Jump To entry fields)

Search instead of Update Now or Refresh

~re-sort or re-filter >> reset to first page

filtering:
- no search box needed if there are no filters or jump-to
- if filters or jump-to, box them separate from other functions (e.g. Create button, paging controls)
- label all filter fields with non-bold non-boxed label, no colon
- layout search box to save vertical space where possible (prefer beside Create button and paging controls)
~ copied from TT iSeries:
• For list filters, unless otherwise specified:

• Text fields default to blank, which means “all”.

• Pulldowns default to “(All)”, unless otherwise specified.

• Search button - type = graphic button, disable after click = yes

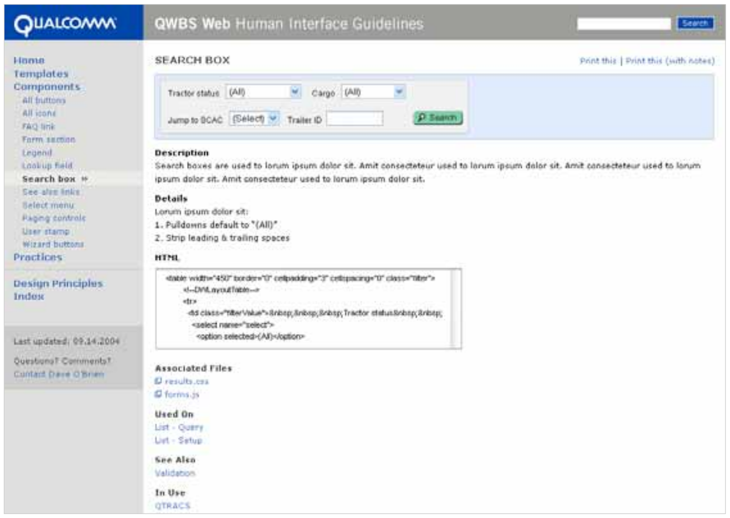
• Filters do not take effect until the user clicks **Search** or presses **Enter** in a filter text-entry field, at which time:

Because we forget that there is an actual human being there, it is all-too common to think of guidelines as the responsibility of the reader. In other words, if all the information is written down, then it’s developers’ fault if they get something wrong.

Here’s what the styleguide looked like when we started. There’s no structured navigation (just embedded links). No orientation beyond the page title. No organization or prioritization of the information. And no thought given to what developers might actually be trying to do when they come to this page.

The assumption here is that people will read and digest the entire document. But no one wants to curl up with an application styleguide by a cozy fire and read it cover to cover (well, maybe someone like me or you, but that’s a different story).

## After



Last updated: 09.14.2004

Questions? Comments?
Contact Dave O'Brien

SEARCH BOX

**Description**
Search boxes are used to lorum ipsum dolor sit. Amit consecteteur used to lorum ipsum dolor sit. Amit consecteteur used to lorum ipsum dolor sit. Amit consecteteur used to lorum ipsum dolor sit.

**Details**
Lorum ipsum dolor sit:
1. Pulldowns default to "(All)"
2. Strip leading & trailing spaces

**HTML**

```
<table width="450" border="0" cellpadding="3" cellspacing="0" class="filter">
      <!--DWLayoutTable-->
      <tr>
        <td class="filterValue">&nbsp;&nbsp;&nbsp;Tractor status&nbsp;&nbsp;
          <select name="select">
            <option selected>(All)</option>
```

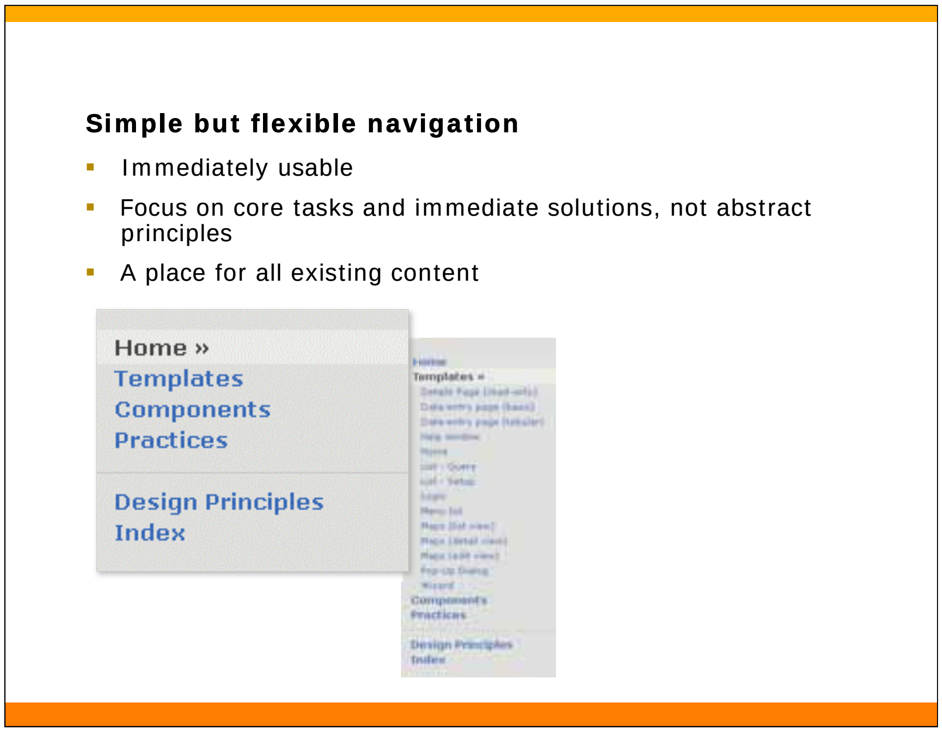
**Associated Files**
results.css
forms.js

**Used On**
List - Query
List - Setup

**See Also**
Validation

**In Use**
QTRACS

As you probably can’t quite see, we took the information, organized and prioritized it, and provided a straightforward, but flexible structure to give it form.



**Simple but flexible navigation**

- Immediately usable
- Focus on core tasks and immediate solutions, not abstract principles
- A place for all existing content

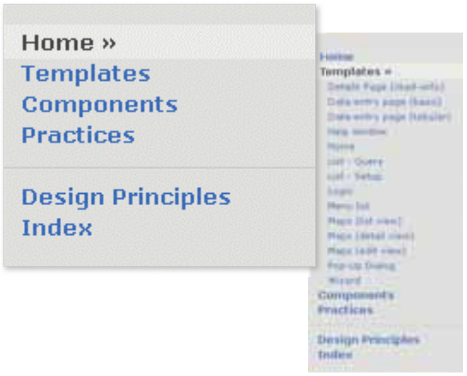
I want to focus on a few of the principles that guided our design decisions.

Overall, we wanted a structure that was simple enough to stay in the background and flexible enough to accommodate varied content but with enough backbone to help people dip in and find what they need quickly.

We divided the content into two primary sections: “Templates” (page-sized design patterns) and “Components” (reusable objects smaller than a page but larger than an individual element like a pulldown menu). We also included a “Practices” section for design guidelines that work across multiple Templates and Components, such as the general use of icons or buttons or colors.

## Emphasis on functionality

- Function over philosophy
- Specifics over generalizations



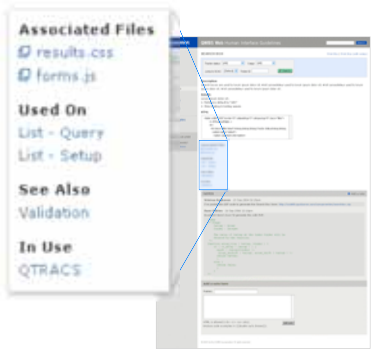
Along with the specific guidelines, the existing content included a fair amount of discussion of underlying principles and the reasoning behind the design decisions.

We wanted to provide a nice home for this more philosophical information (with the idea that some developers would want to understand the "why" of what they were being asked to do), but we also wanted to keep it out of the way of people merely looking for the "what."

So we included a secondary section called "Design Principles".

## Structured lateral navigation

- Cluster similar objects
- Move between related objects
- Selective linking to maximize relevance

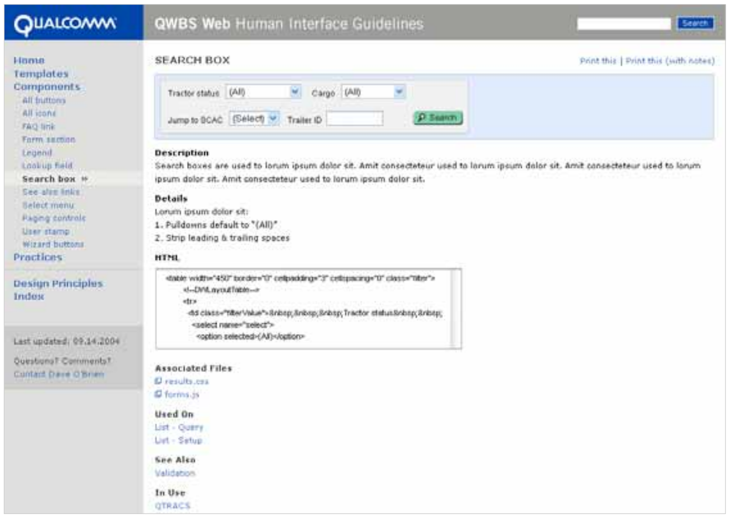


Our simple primary sections worked nicely for a top-down navigation path, but didn’t help much with lateral relationships or relevant offsite links.

The original guidelines we were working from followed a wiki-esque approach, where embedded links could appear anywhere and link to anywhere. But, while this approach is *infinitely* flexible, it (again) assumes people will read through the entire document and inevitably leads to a lot of navigational thrashing around.

Instead, we provided a structured set of lateral navigation options, appearing at the same place on every page. Here, we offered clearly-labeled links to associated files (such as CSS or Javascript), to parents and children, to related topics, and finally to working examples of the item in action.

## Graphic Design



SEARCH BOX

**Description**
Search boxes are used to lorum ipsum dolor sit. Amit consecteteur used to lorum ipsum dolor sit. Amit consecteteur used to lorum ipsum dolor sit. Amit consecteteur used to lorum ipsum dolor sit.

**Details**
Lorum ipsum dolor sit:
1. Pulldowns default to "(All)"
2. Strip leading & trailing spaces

**HTML**

```
<table width="450" border="0" cellpadding="3" cellspacing="0" class="filter">
        <!--DWLayoutTable-->
        <tr>
          <td class="filterValue">&nbsp;&nbsp;&nbsp;Tractor status&nbsp;&nbsp;
            <select name="select">
              <option selected>(All)</option>
```

**Associated Files**
results.css
forms.js

**Used On**
List - Query
List - Setup

**See Also**
Validation

**In Use**
QTRACS

Finally, I just wanted to say a word about graphic design. Too often, graphic design is thought of as window-dressing, as empty style, as a "skin." In fact, graphic design is the practice of effective visual communication. Since guidelines are at their core about communication, it would obviously be a mistake to ignore their graphic design.

For example,

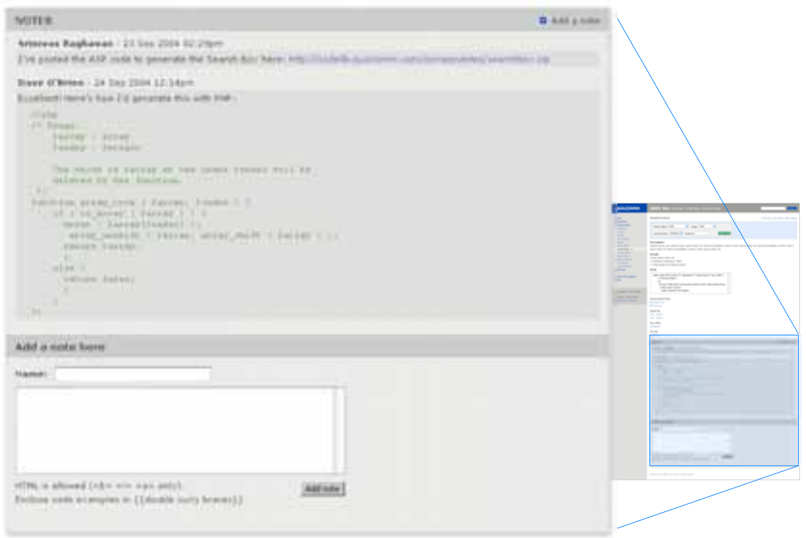
-We used color and typography to highlight the most important elements throughout the guidelines..

-We used visual design to reinforce and support the site's organization. So, Templates always look like Templates, Components always look like Components and so on

-We used pictures rather than only text to represent visual objects. Since developers would be searching for things based on what they look like, we wanted to show them rather than describing them wherever possible.

I also just wanted to note that, unlike say a product marketing site, the graphic design here is minimal and purely functional. It is not intended ever to be noticed, but is there to support user tasks and goals.

## Easy, natural collaboration


NOTES

Add a note

Srinivas Raghavan : 23 Sep 2004 02:29pm

I've posted the ASP code to generate the Search Box here: [illegible]

Dave O'Brien : 24 Sep 2004 12:14pm

Excellent! Here's how I'd generate this with PHP:

```
[illegible]
```

Add a note here

Name:

HTML is allowed (<b> <i> <a> only)

Enclose code examples in {{double curly braces}}

Add note

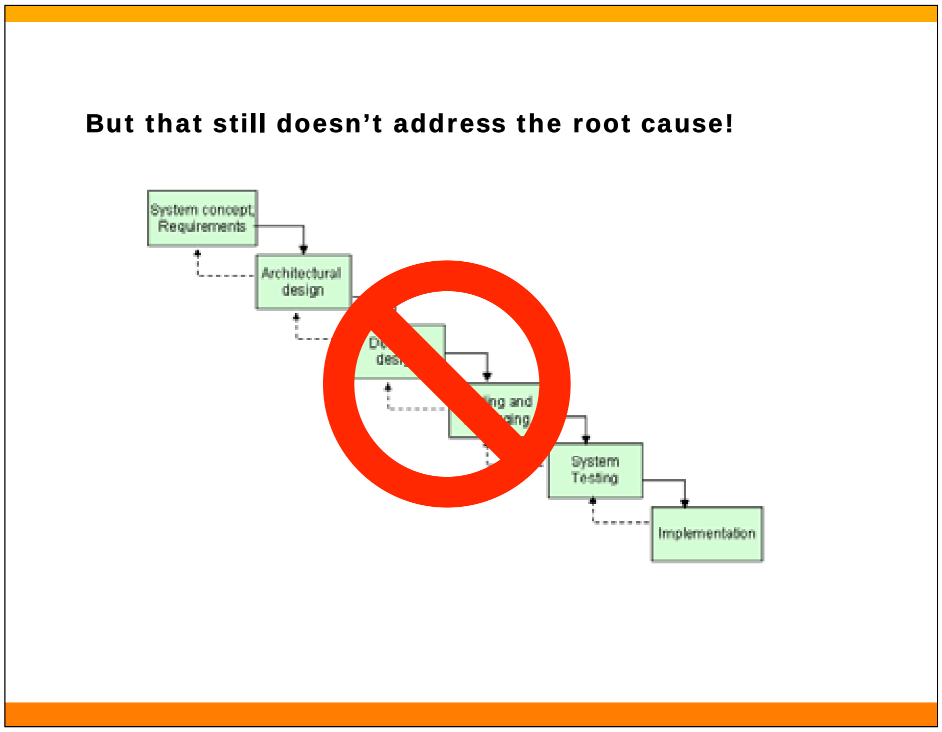
So that covers the documentation aspect to this. There is also a collaboration aspect, which was guided by three principles:

1. We wanted the guidelines to be a living document
2. We wanted developers to feel a degree of ownership over the guidelines
3. We wanted the tool to foster collaboration between developers and designers.

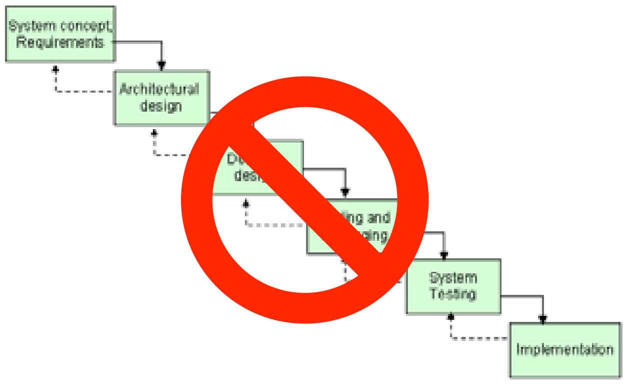
From these principles came a simple but effective solution: We built the tool on the MovableType weblog publishing platform which allowed us to include straightforward blog-style comments on every page of the guidelines.

So, when you're looking at the page for a given item, you see another developer's suggestion for modifications or a link to relevant code or a request for clarification. By including the ability to comment right on each page (rather than in a separate forum, or, worse still, leaving them to languish in email), we opened the document itself to modification by the people using it.

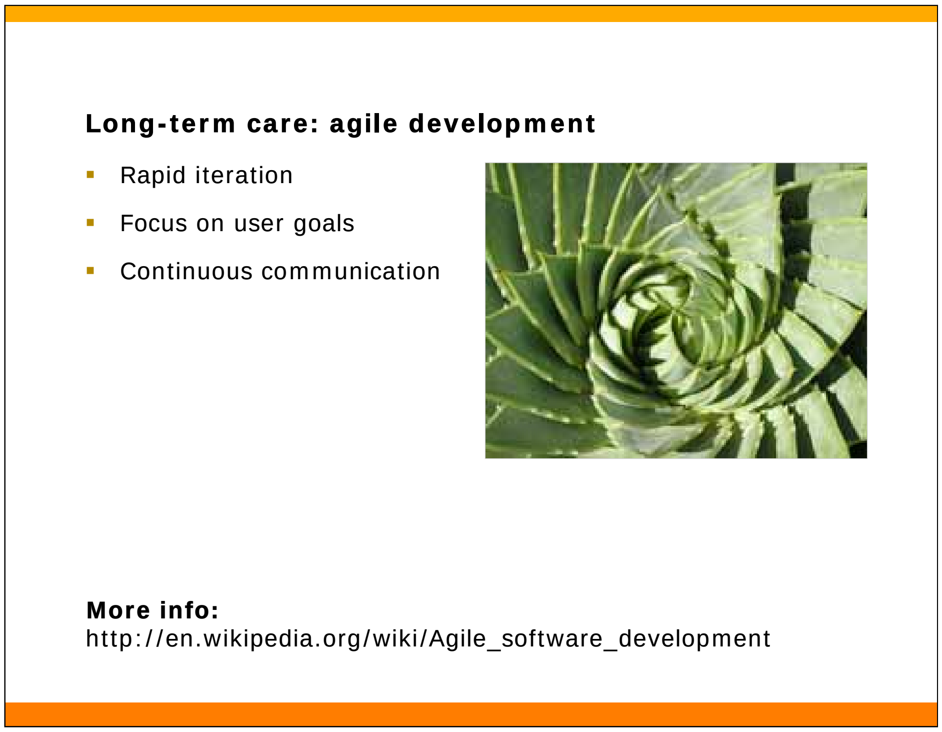
To further support this idea of a living guidelines document, whenever a comment is posted, an email goes out to a distribution list of the designers and managers responsible for the guidelines. This means that the



**But that still doesn't address the root cause!**



All this work is still only part of the story. Creating consistency is a starting point to organizational change.  The real goal is to change the development culture so that it's more flexible and better at managing risk…



## Long-term care: agile development

- Rapid iteration
- Focus on user goals
- Continuous communication

**More info:**
http://en.wikipedia.org/wiki/Agile_software_development

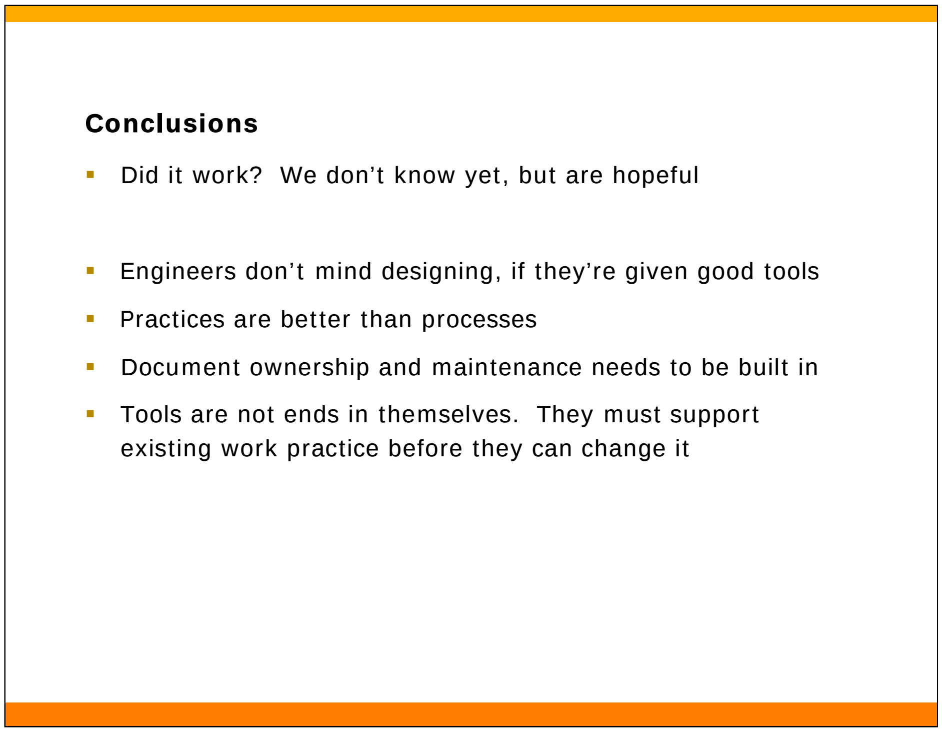
We did this by example, and built the system in a way that introduces ideas of flexibility, user-centered development and risk management into the organization.  We built the system using several techniques borrowed from agile software development, we built the system using

Rapid iteration: one-week cycles

Focus on user goals: stories and prioritization; everything explicitly addressed user goals; prioritized stories every week

Continuous Communication: IM and velocity

Photo by: elroySF, Flickr



## Conclusions

- Did it work?  We don't know yet, but are hopeful

- Engineers don't mind designing, if they're given good tools
- Practices are better than processes
- Document ownership and maintenance needs to be built in
- Tools are not ends in themselves.  They must support existing work practice before they can change it

Developers were excited and eager

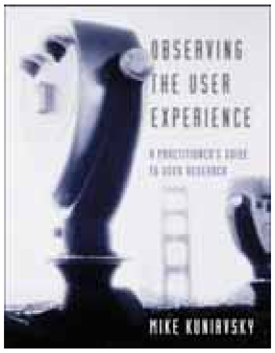
The process of thinking about what guidelines were for, how they were supposed to be used, why they weren't used proved valuable.  Doing that to other practices could prove equally valuable.

Introducing practices is more effective than introducing processes.  Developers were much more eager to use this system than they were to do "more work." Everyone hates change, even if they hate the status quo.  Having people excited about changes already gave the project momentum it wouldn't have had otherwise.

## Thanks!

**Mike Kuniavsky**
mikek@orangecone.com

**Nadav Savio**
nadav@giantant.com


OBSERVING THE USER EXPERIENCE
A PRACTITIONER'S GUIDE TO USER RESEARCH
MIKE KUNIAVSKY

Morgan Kaufmann, 2003

ISBN: 1558609237

**Full case study**
http://www.orangecone.com/kuniavsky_guidelines_CASE.pdf