# High-Fidelity Prototyping

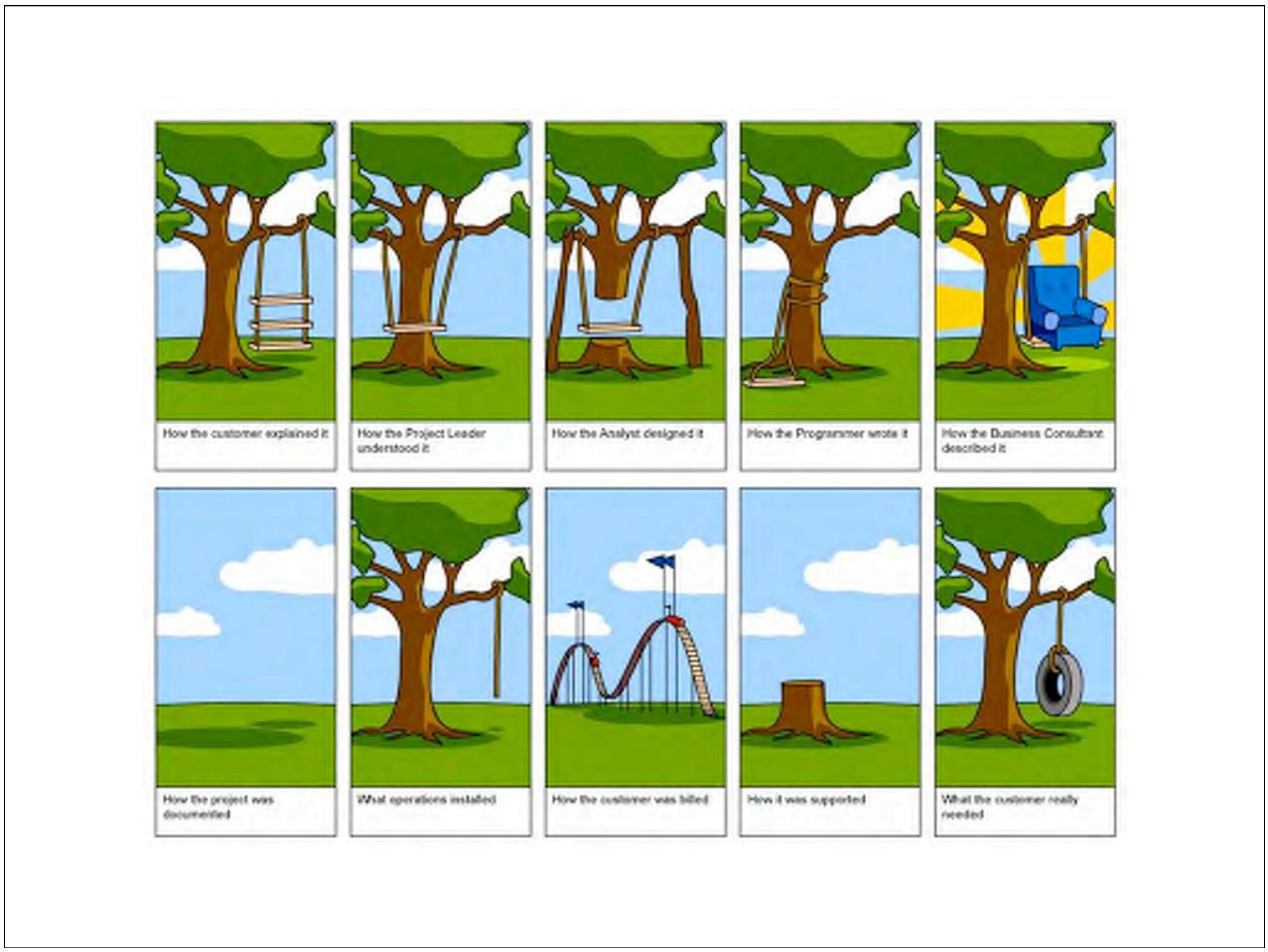
a case study and discussion


presented
for Refresh Dallas
on September 14, 2006
by Mark Kraemer



A discussion on High Fidelity Prototyping
presented
for Refresh Dallas
on September 14, 2006
by Mark Kraemer

for more information visit http://markup.thekraemers.com






By the nature of their roles and past experiences, each stakeholder will have their own perspective on what a project should be.

# Why prototype?

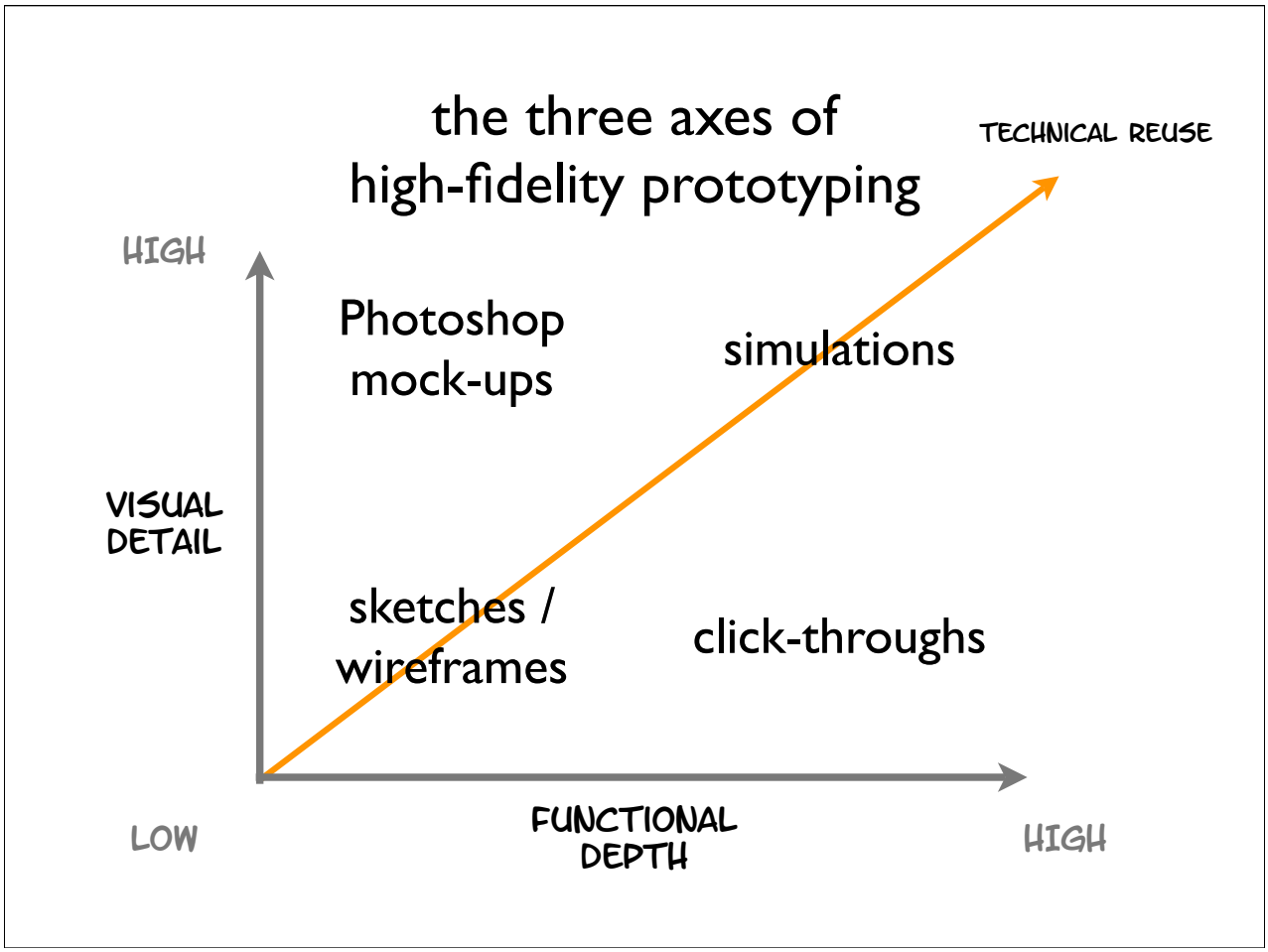
Designer: ensures that screens are designed

Client: UAT happens sooner than later

Back-end developer: illustration > words



Lots of reasons why we prototype, but each stakeholder has particular benefits from the prototyping process.






Lots of people are talking about “hi-fi prototyping”. Just like introduction joke illustrated, “hi-fi prototype” means different things to different people.

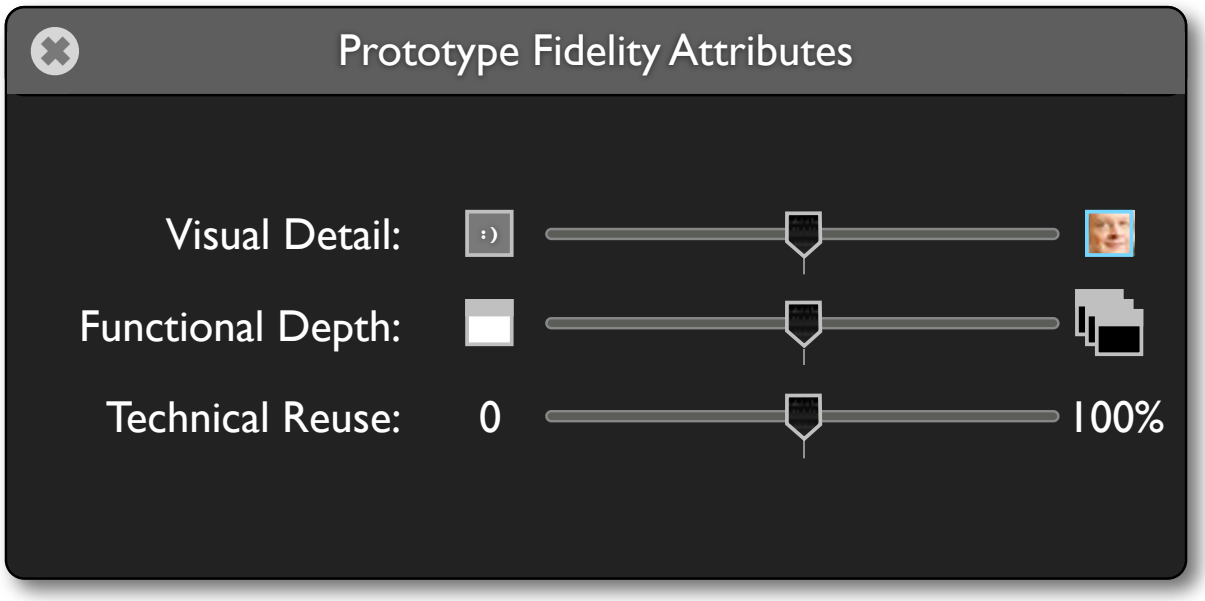
Graphic Detail: spans from Page Description Diagrams ➜ Sketches ➜ Photoshop mock-ups

Functional depth: spans from Page Description Diagrams / Sketches ➜ Paper Prototypes / HTML click-through wireframes

Combine both: rich HTML click-throughs models ➜ simulations ( iRise, Axure, Serena Composer)

Generally, the more fidelity in any of the three major attributes requires more time, cost, or talent to produce.

# which style is best?






Greater fidelity for any of the primary attributes will require more time and talent. If you do all 3 full-tilt, then you're not prototyping, you're developing. (Fidelity means a copy, not the real thing!)

We're user-centric, right? The prototype is an interim deliverable itself. It has users.

So, the situation begs the question "who is this for, and how will they use it?" Let's match the fidelity to meet their needs.

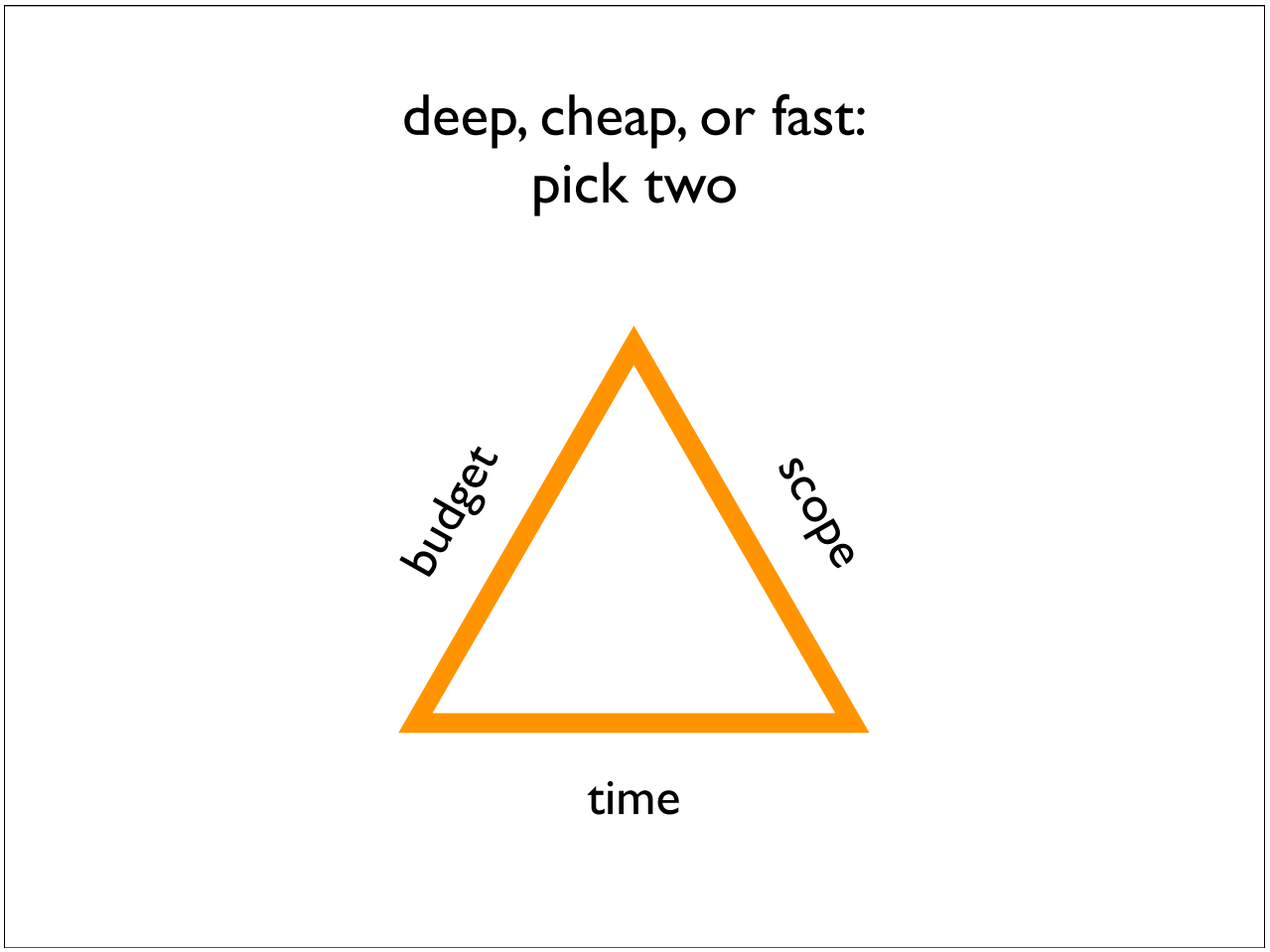
## who are the prototype's users?

| | visual detail | functional depth | technical reuse |
|---|---|---|---|
| investors | ✔ | ? | ✖ |
| project sponsors | ✔ | ? | ? |
| end users | ? | ✔ | ✖ |
| business analysts | ✖ | ✔ | ✖ |
| graphic artists | ✖ | ? | ✖ |
| front-end developers | ✔ | ? | ? |
| back-end developers | ✔ | ? | ✔ |



This is a short list of possible users with possible needs—different users on different projects will have different needs.

The big question is "who needs to approve this before we move forward in the project, and how will the person I am handing it to use it?"



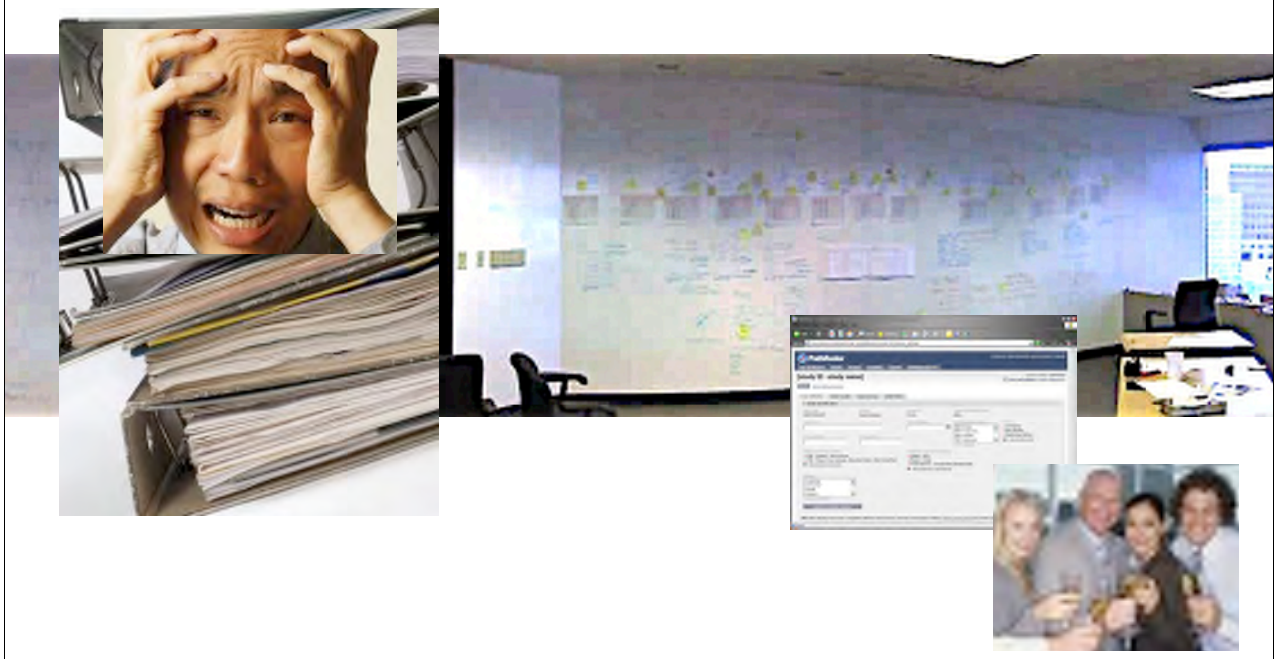



Physical constraints of what any team can provide dictate that a project's budget, scope, and schedule must be proportionate. But most stake-holders will want as much fidelity as you can muster, so trade-offs will likely need to be made.

Expose the stake-holders value system early. Ask them "What's most important? Time to deliver, budget, or scope?" Knowing the client's value system up front will help later on if trade-offs need to be made.

case study:

# The Pathfinder Project



background:

# the premise

1. collect

2. share



The client was a consumer goods company with international presence. They had many brand and regional managers all around the world who each had their own process and system for collecting market research. The client realized they could all benefit by sharing their research.

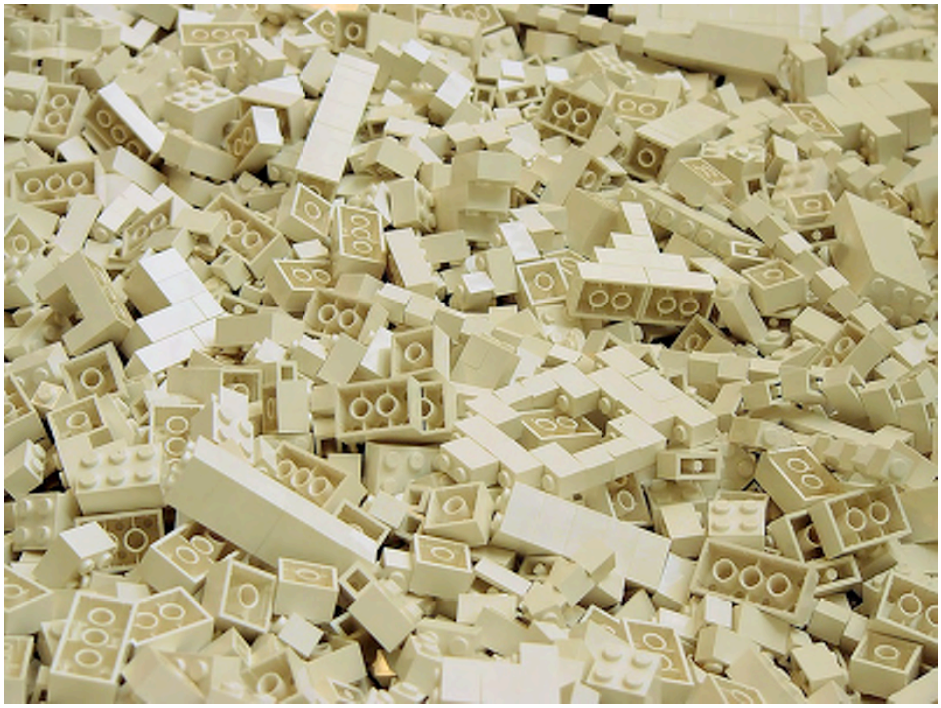
The project seemed deceivingly simple because it was a specific task for a specific function.

However, it was really complex because:
- Crossed all products in all divisions (division managers, brand managers)
- Crossed all countries (all of those above x lots more)

Seems simple, but every business unit in every country had their own method and they weren't eager to change.

## background:

# the technology

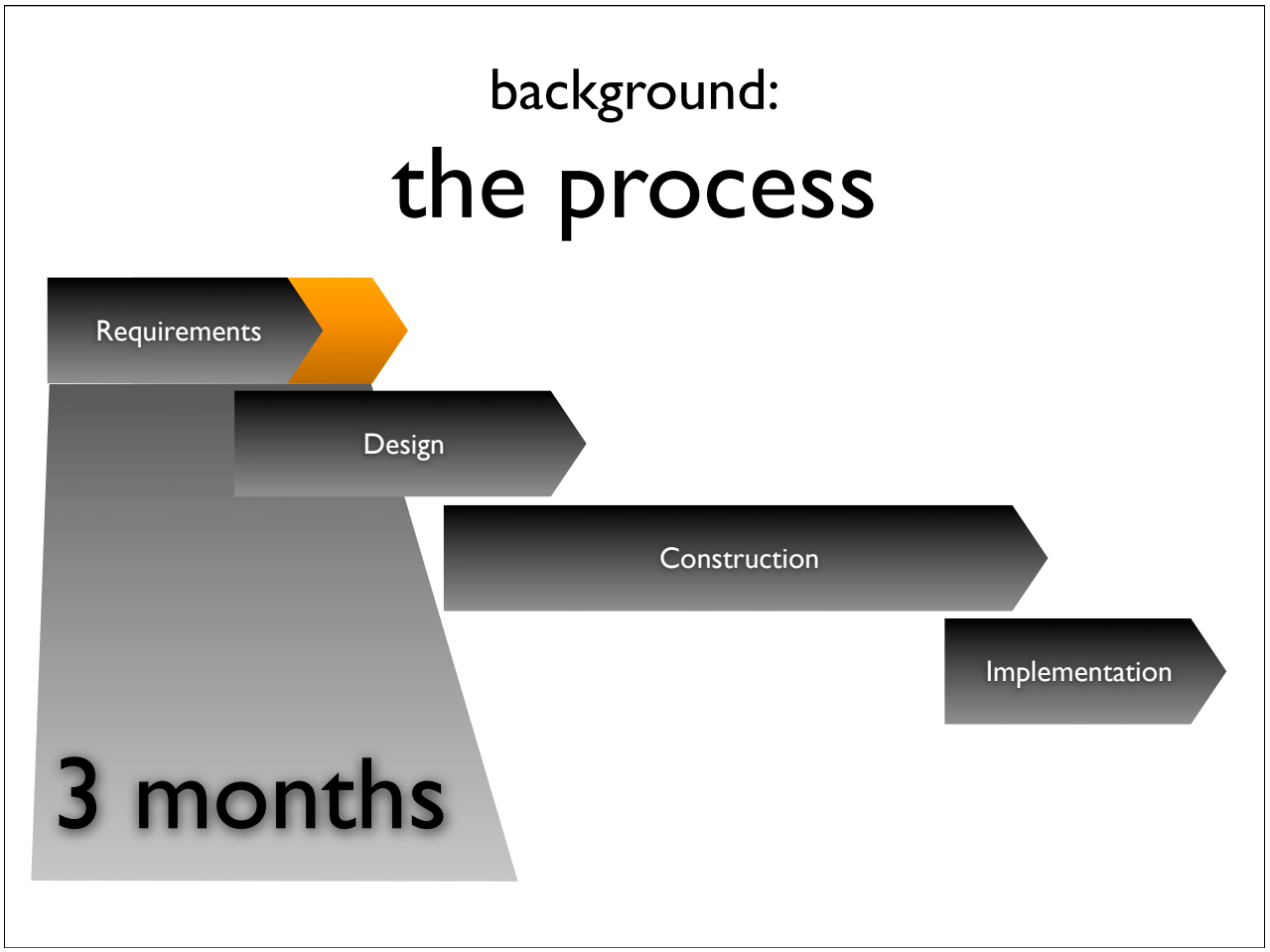




The client was using a package our expert developers implemented many times before.

We were expecting custom coding on the back-end, but our previous projects' front-ends used standard screens with only minor modifications. In the past, they rarely customized the UI much if at all.

I was brought in early on as a UI architect for 2 days of requirements / design for some concepts / mock-ups.





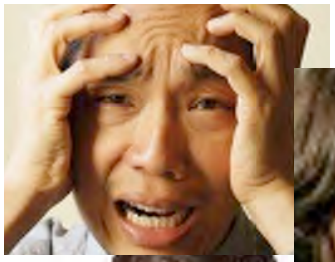

Budget dictated we use the client's "large project" methodology. It was a traditional waterfall / stage-gate approach, ISO9000-like rigor.

Requirements → Design Construction → Testing → Implementation

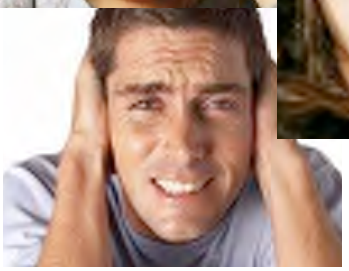
It took 3 months to get a first draft of the requirements using their requirements for requirements.

# requirements results

sponsors



PM / BA



developers

12

When printed, a single copy of each document filled a 2” binder.
The pages were:

30% Word documents and
70% Excel tables listing requirements
with a few charts sprinkled in

Lots of deliverables to show (everyone worked hard), but little value to demonstrate:

The business units were not ready to sign-off as they weren’t comfortable that their specific needs were being addressed.

Our team didn’t have confidence to bid on the construction because we still didn’t have a clear picture of what the requirements really required.

This very large project was stalled.

What everyone did learn: the project was much more complex than a configuration for a product “out-of-the-box.” None of the standard workflow or library screens were going to fit the complex requirements.

# what was missing?

| | visual detail | functional depth | technical reuse |
|---|---|---|---|
| project sponsors | ! | ? | ✖ |
| regional / brand players | ! | ! | ✖ |
| client PM / BA | ✖ | ! | ✖ |
| our PM / BA | ! | ! | ! |
|  UI architect | ! | ! | ! |
| our back-end developers | ! | ! | ! |

# what do we try next?



We pow-wowed for a week to figure out the next steps.

How can we figure out how to estimate the construction for this thing when we don't understand the big (medium, or little for that matter) picture?

We already worked hard for 12 weeks on the requirements, what other deliverables do we need to get a better understanding?

We even brought the key client business analyst to think with us for a day.

We needed less words and more pictures!

We needed a graphic representation of the workflow!

We decided we needed a prototype!

## functional depth:
## no problem — we know workflow


15

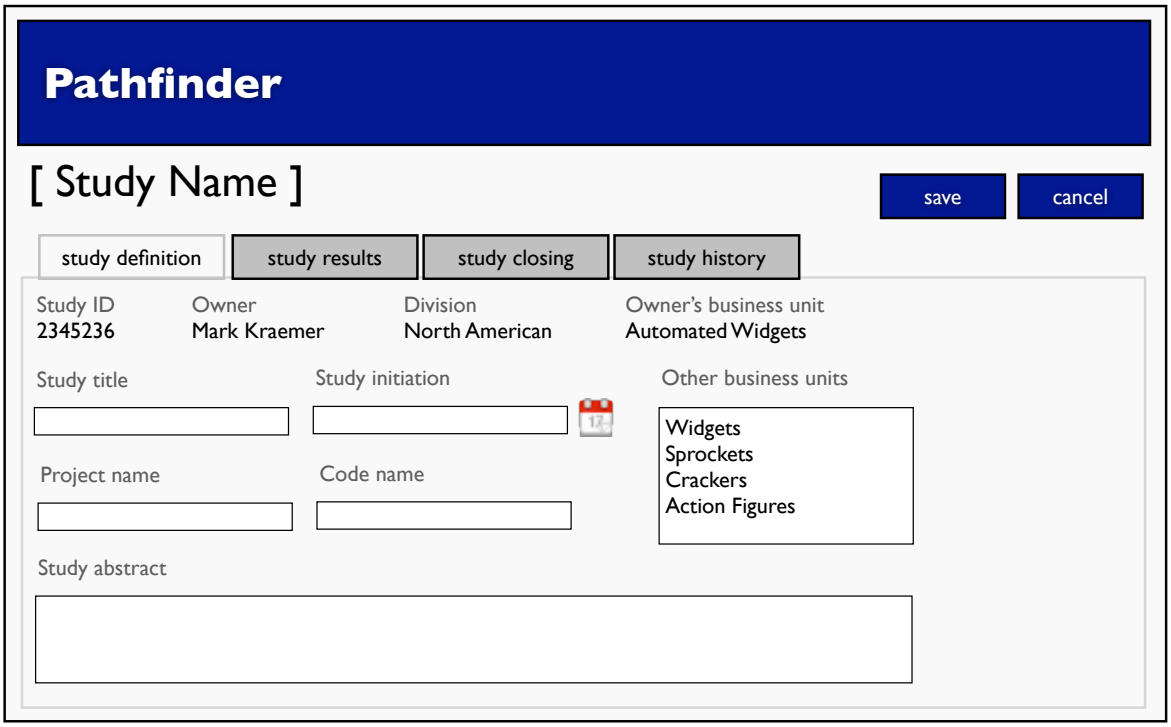
The documented requirements were lots of lists. There wasn't any context for the steps of the most complicated workflow processes.

We needed a graphic representation of the workflow!

No problem. We had done lots of workflow projects, and lots of workflow diagrams. We needed to do them for this project as well. The client methodology didn't include them, but we needed them to understand the workflow.

But the workflow in a nice diagram won't be enough. Workflow alone wouldn't demonstrate the various perspectives required for each step in the process.

# visual detail:
# I Know It When I See It

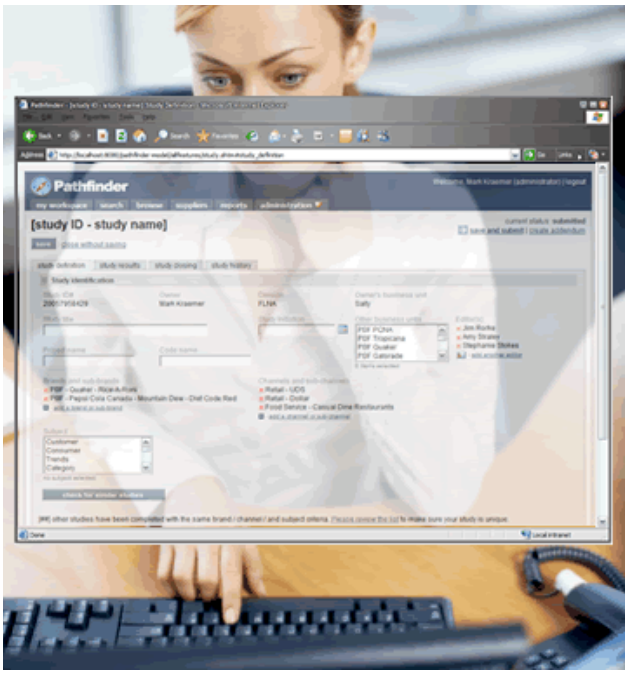





Back in the requirements process, we drafted a few screens in PowerPoint. The client wasn't visually demanding (branding, emotional appeal, etc.)
so, maybe we could just extend the PowerPoint screens?

Well, not really. We already exhausted the "data to pixel" density on the preliminary screens. We needed a lot more detail than a non-scrolling, feels less than 72dpi PowerPoint page could provide.

# I Know It When I Use It




visual style =
understanding +
enthusiasm

17

Visio or other wireframes could provide more functional detail, but they lacked the final look and feel we needed to emotionally re-engage the clients.

Have you ever gotten excited about looking at a large collection of wireframes? - IAs are not allowed to answer!

We needed lots of visual detail to show exactly how it would look. We needed to eliminate the clients' confusion of interpreting the requirements.
Something like Fireworks or Photoshop would work. If the different stake-holders could see what we were proposing, we'd eliminate at least half of the communication required to get consensus.

We could even string them together as images in a browser and use image maps to let them click through the basic use cases.

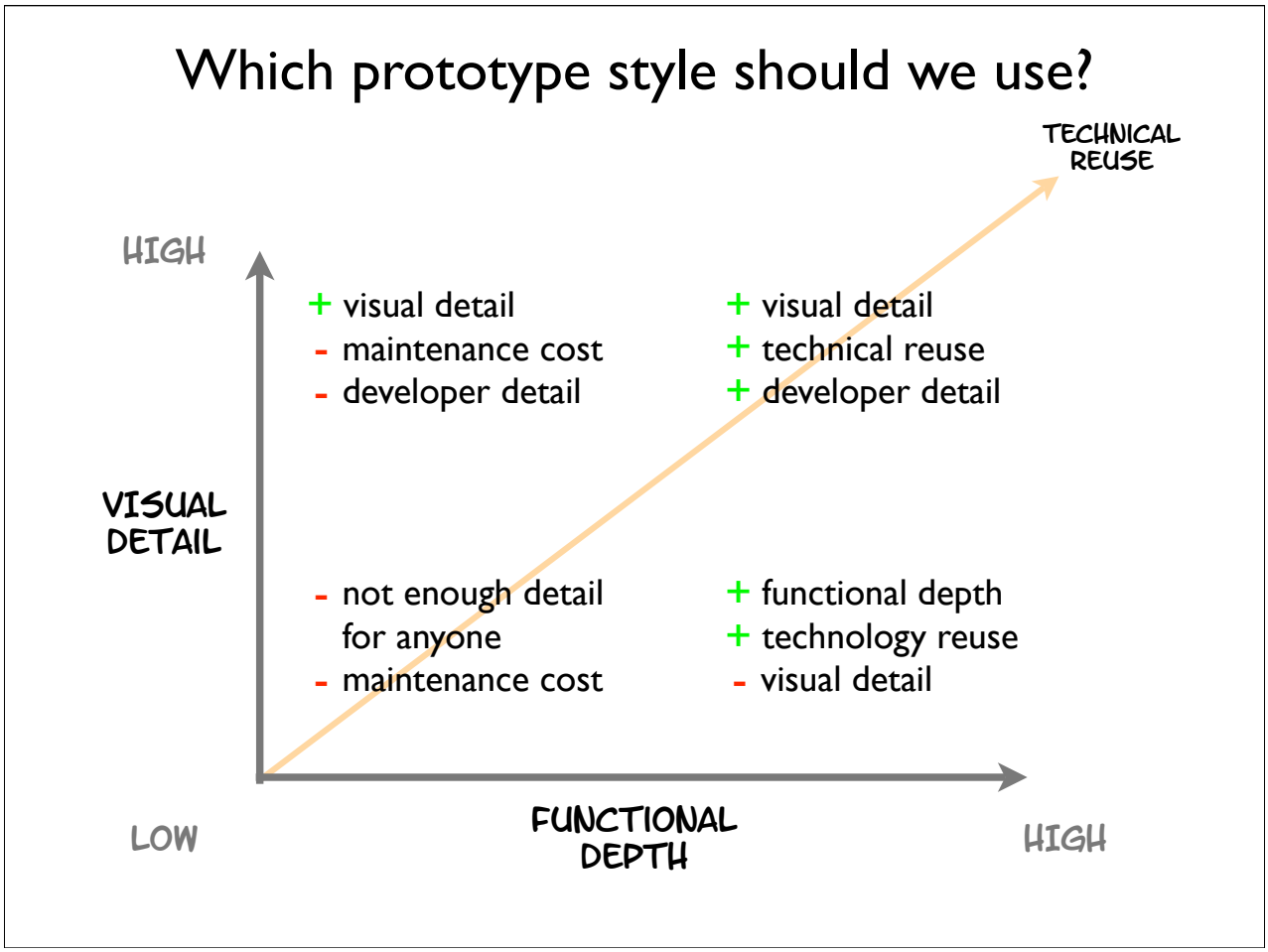
That sounds great, but the developers were in the room too.




How The Heck Am I Going To Code That?

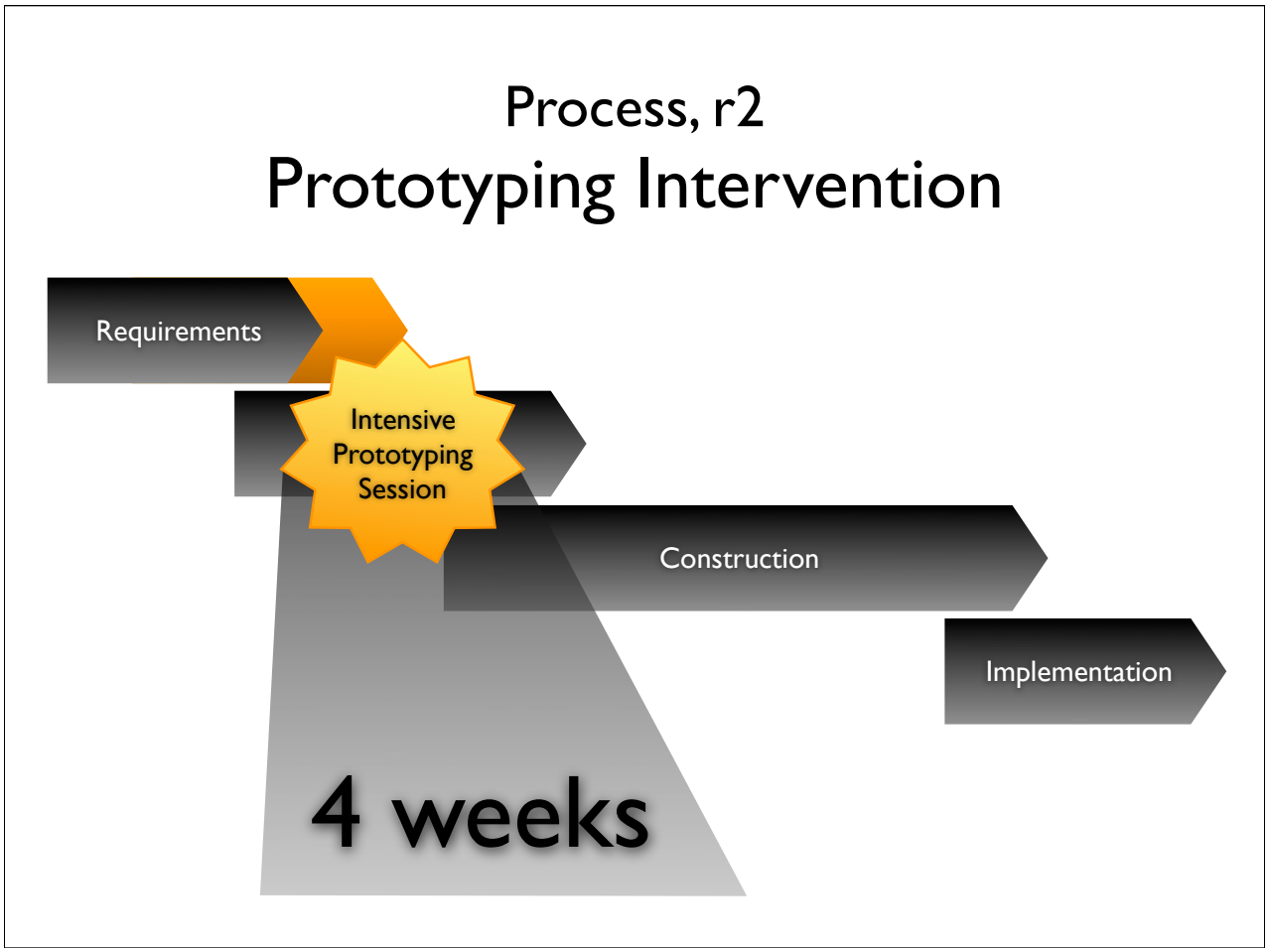
The developers realized that this project was going to be more complex than configuring a product-package.

The development team understood basic HTML, but they back-end specialists. They weren't front-end developers. They needed some development help for the HTML, CSS, and javascript behaviors.






Given that every stake holder needed more detail in each attribute before anyone could commit to move forward on the project, we needed a realistic looking prototype that demonstrated depth and breadth of functionality all through real presentation-layer code.



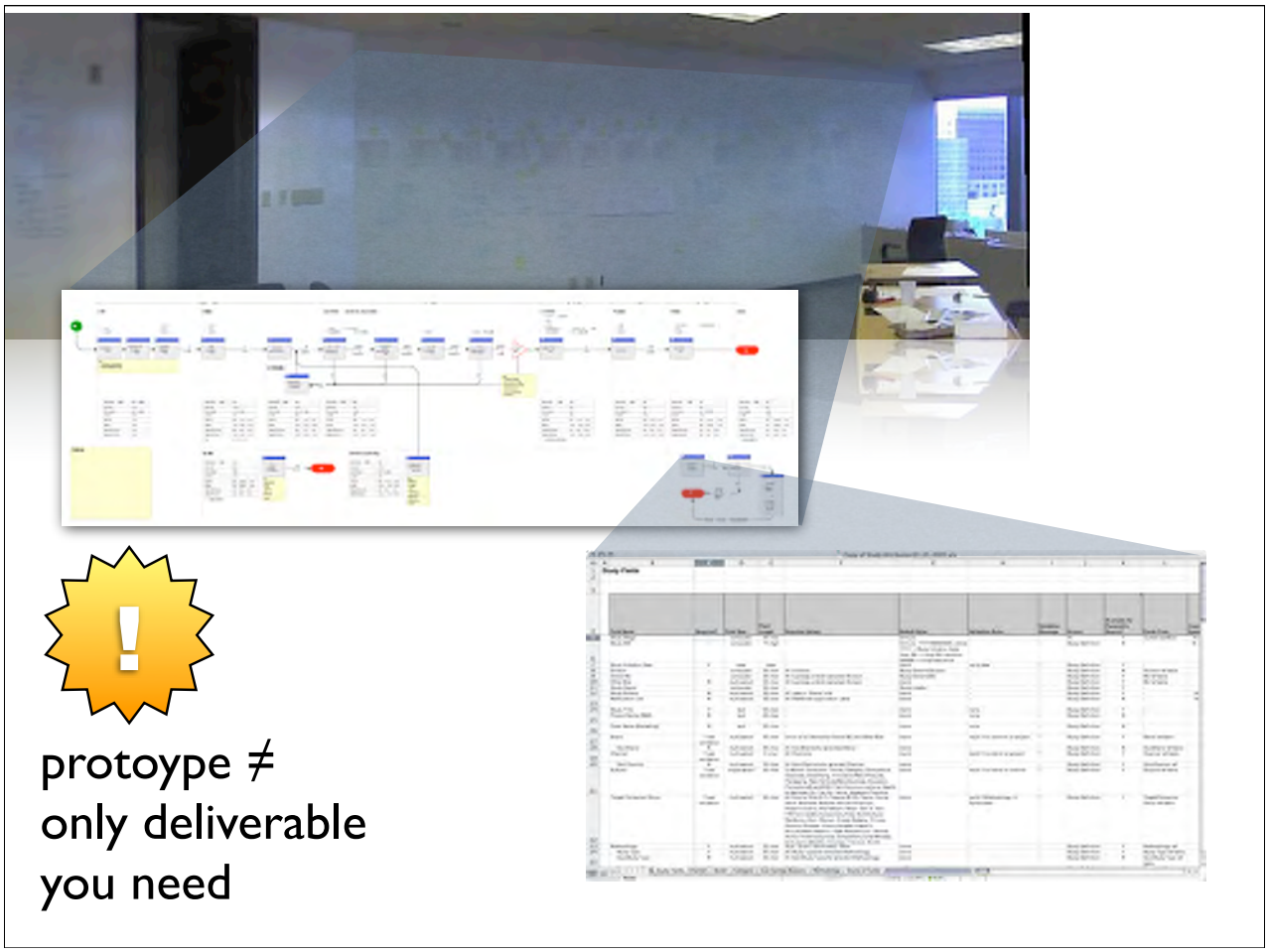



We decided to ask for 4 weeks to develop a hi-fidelity prototype that would:

- Demonstrate in detail how each of the primary use cases would look on real screens
- Use real HTML, CSS, images, and javascript to render the screens

To ease concern about more slippage, we partnered to agree this would take the place of the design phase and would be back on schedule upon completion.

We're starting UI construction (the UI framework) while we're still in design!




21

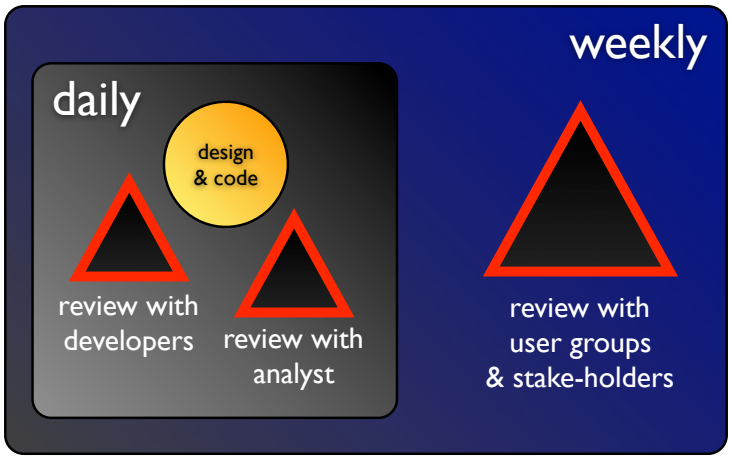
We flew the business analyst in to spend the week with us.

We talked through each use case in excruciating detail.
- outlined the steps
- identified the audiences for each step
- identified the fields in each step (both editable and display-only)
- identified the security for each field in each step for each user (read/change/hide)

We used:
- dry erase markers on a floor to ceiling whiteboard
- printed excel spreadsheet (field inventory with type, validation rules, security matrix)
- sticky notes (exceptions and other notes)
- hand-drawn sketches to rough out each screen (full page, low detail key concepts)

We turned all that into a giant Visio diagram, each box with the blue top-border indicated a required screen. Each screen in turn had its own detailed spreadsheet.

# Weeks 2-4



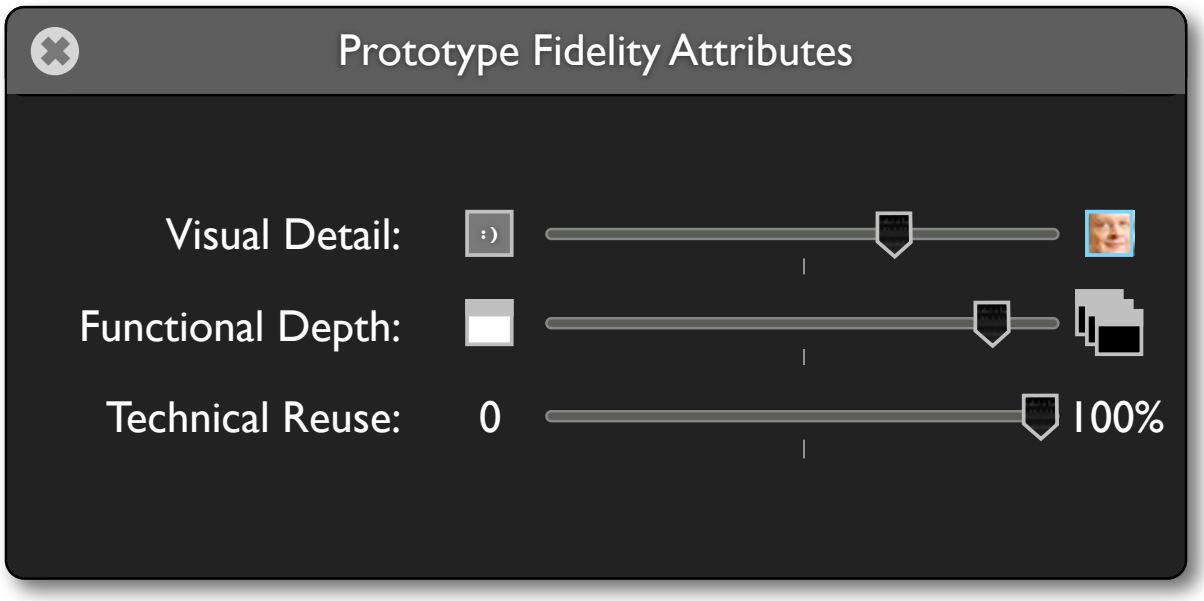

prototyping thrives on feedback



Daily: code, late-day review with developers and with the business analyst
Weekly: review with a major stake-holder group.

Prototyping thrives on feedback

# building the prototype






We were shooting for:

Visual Detail: higher than average

Functional Depth: we needed to demonstrate at least 90% of the total functionality to have confidence to bid on the construction phase

Technical Reuse: we were constructing the presentation layer mark-up and style guide as we went. The project's design was was the UI construction phase.

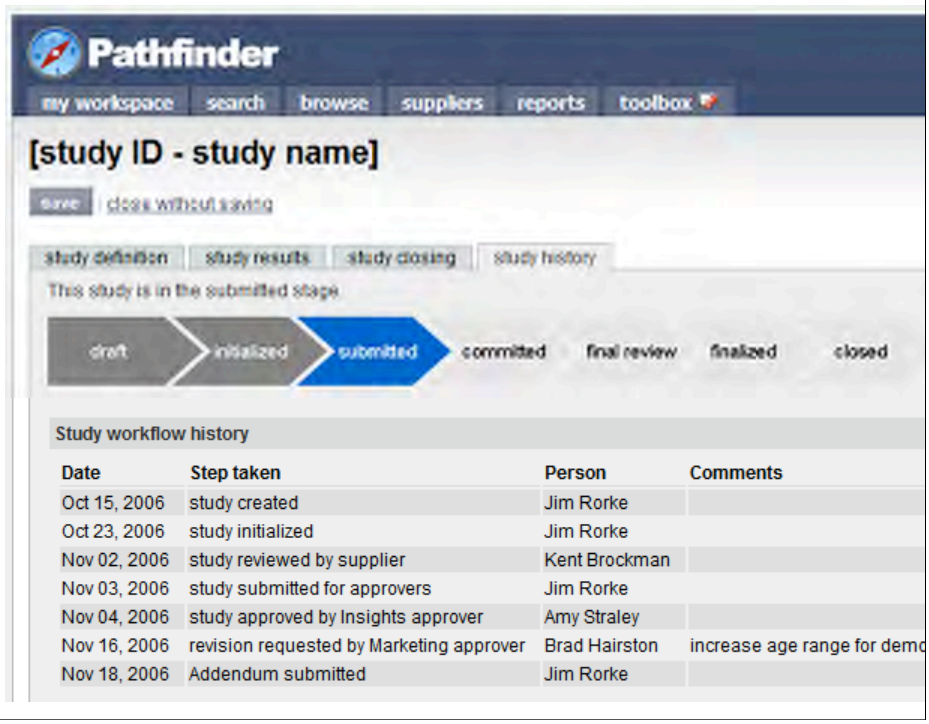
# visual style

GENERIC LOGO

JUST A HINT OF COLOR (MOSTLY GRAY)

ALL IN CSS FOR EASIER CHANGES LATER

we built the visual foundation






Being an internal application, the stake-holders had limited demands for the application look and feel.

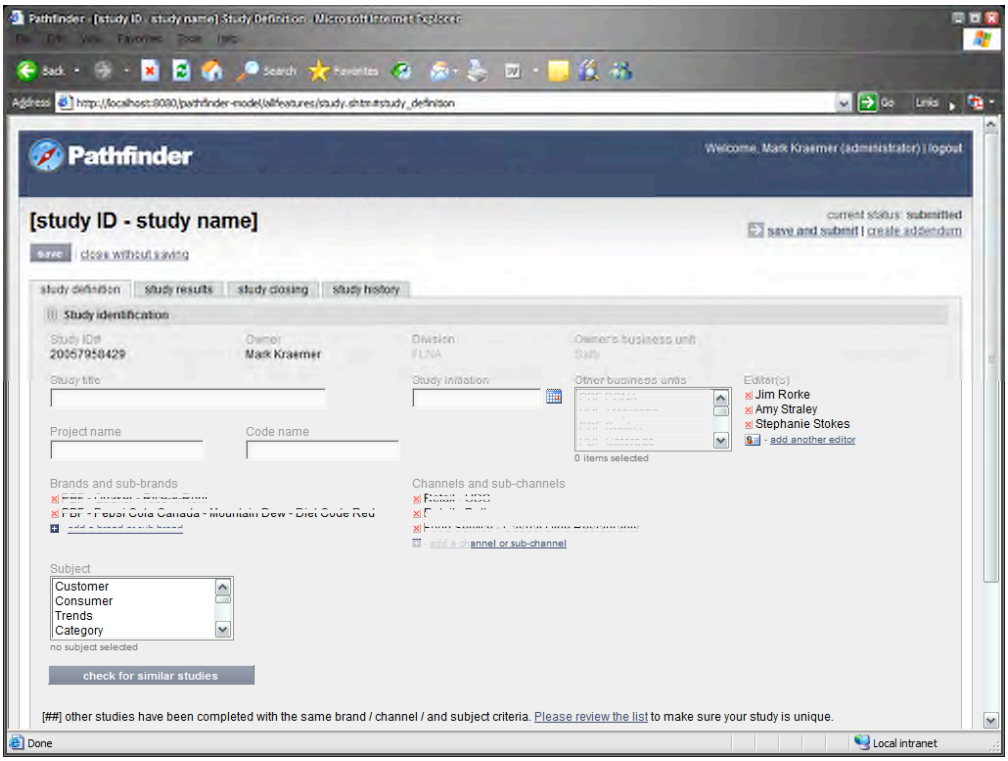
They wanted it to have a “commercial-grade” visual style, but did not have specific requests for appearance.

We provided a modicum of style to make the prototype feel more like a real application than just HTML wireframes.

Proper separation of style from layout will allow visual style tweaking later in the process.

(The only thing we ended up changing in implementation was the application name and logo!)

# content / functionality






Start with the base-case: the most common, least complicated pass through the workflow.

This is the soul or the essence of the application. What's the big idea? Let's address it first.

We made it scenario-based, created a little story for consistent context from screen to screen.

Worried about the task itself (the "little IA") rather than the global navigation.

# scenarios provide context

scenario : use case
::
persona : target audience

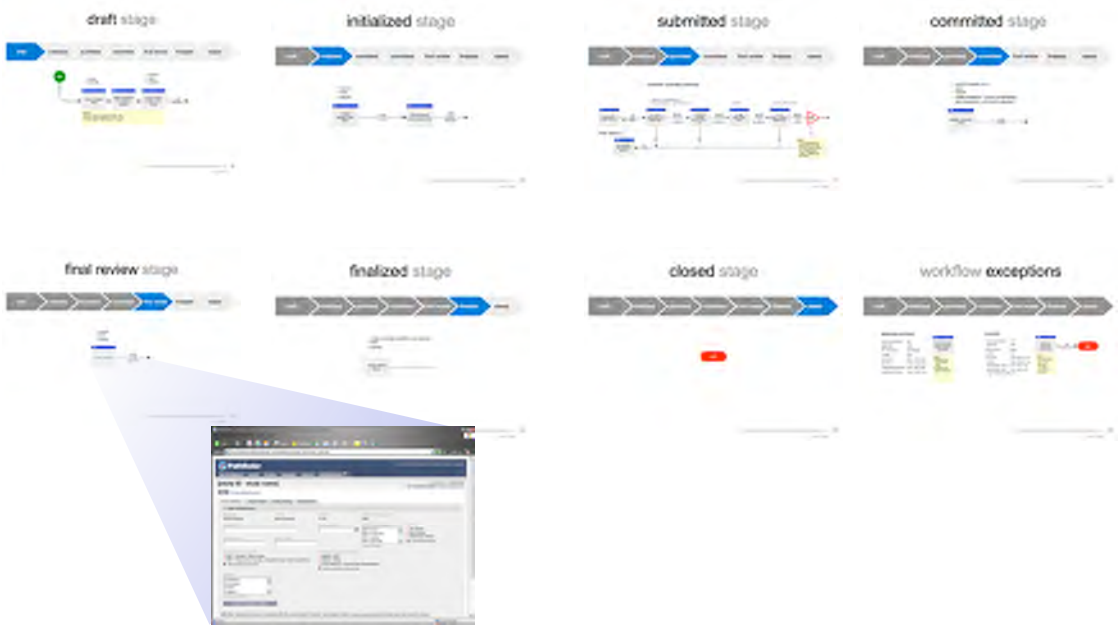
a real story provides needed context



Use-Cases are often sterile, written in a purposely generic style. When fleshing out a use case for prototyping, use a story with real names and real entities that would be common to the process. The stake-holders familiarity with these entities will help guide them through the new screen flows and process.

See Dan Brown's "Representing Data in Wireframes" IA Summit Poster available online at http://www.greenonions.com/archives/2005/03/08/ia-summit-posters/ for a great explanation of why non-realistic data doesn't do justice to a prototype's need for functional fidelity.

# went deep
# before going broad

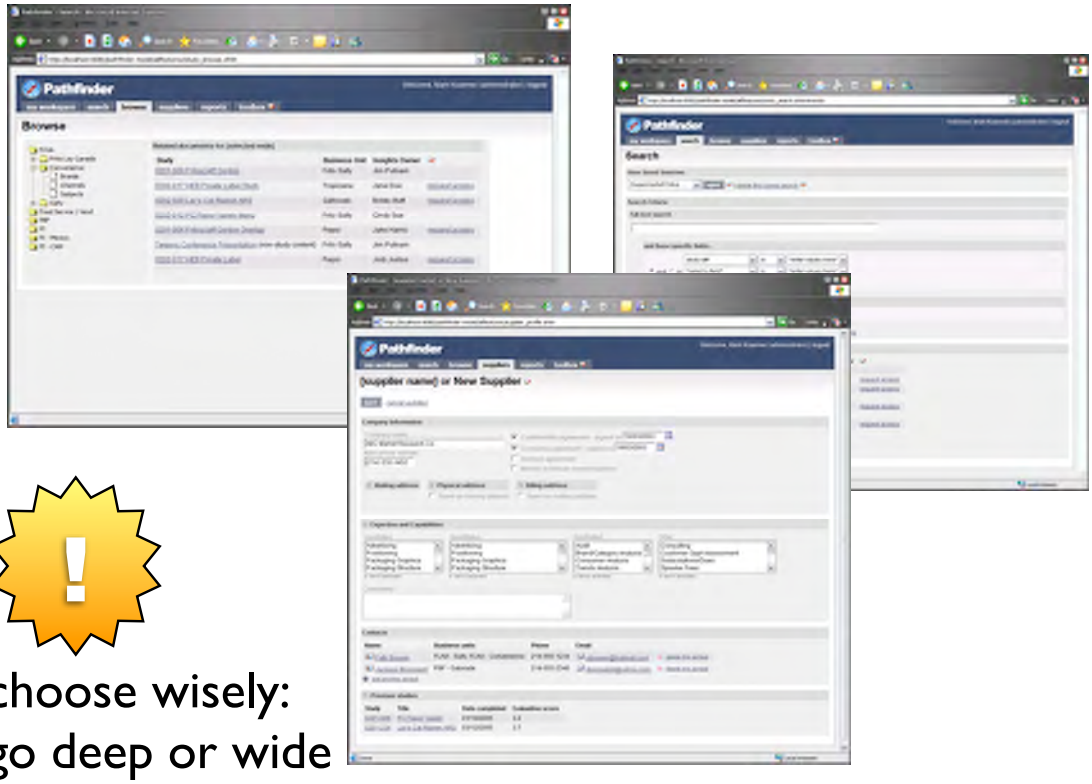





We added the complexities to the first workflow before worrying about the other functions.

The eight stages illustrated above (including those grouped together under “exceptions”) each had 1 to 6 screens. We developed a hi-fi mock-up of each screen using a consistent scenario for context to guide users through the entire process.

# attacking other functions






choose wisely:
go deep or wide



After we were satisfied with the complete first function, we started concentrating on the other functions.
Each new function introduced the need for more elements in the global navigation “big IA”.

## interaction depth: how low can you go?

- at a minimum try to provide **order of screens in a workflow**
- then show at least one example of **validation messages**
- as time allows, show as many **interactive behaviors** as you can

show as much as you can, at least enough to explain the big idea



At least show screens in the proper order of the most prominent workflow.

Provide an example once, no need to repeat or branch if the pattern remains the same.

If there are interactive behaviors that are unique to the project, try to demonstrate them with as much fidelity as time and your media allow.

## technical reuse: how to stage?

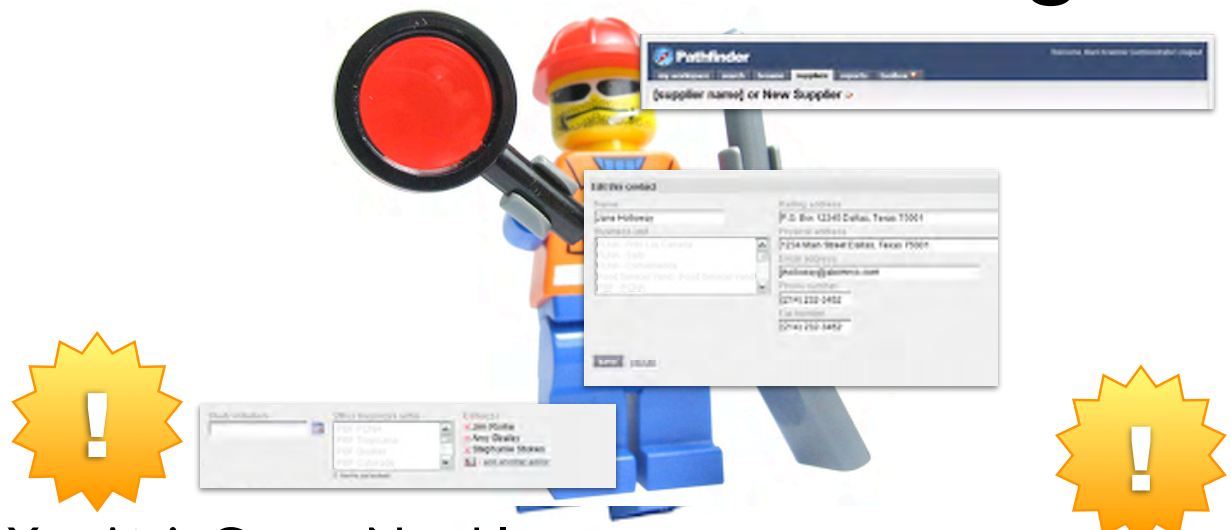
- completely static - no server?
  + portable (run on any machine without a connection)
  - JavaScript for includes / reusable code
- use a simple server
  + reuse via SSI, PHP, JSP, ASP, Ruby, etc.
  - need server or connection to the server



Sometimes when working at a large client with restrictive Internet / Intranet security policies, you might not be able to get to a development server easily or quickly. A completely static solution might be your only choice until an internal development server can be procured and configured.

technical reuse:

# YAGNI vs. refactoring




You Ain't Gonna Need It,
just build it!

when you do re-use,
then re-factor

31

**You Ain't Gonna Need It.** Don't over-think how something is used. If you need a new style for a specific situation, code it inline or in the head.

Architecting too much up front will take valuable time that could be spent iterating details with your users.

Once you find yourself needing that style or page component again (a global navigation component, groups of fields that appear in multiple pages) that's the time to refactor, pull it out of the single page, and put it in a place where it can be called from multiple pages.

Typical items that are reused in a prototype:
- Global Navigation
- Sub-navigation metaphors (tab sets, side-bars, etc)
- Common field controls or sets (dialog boxes, pop-ups, address fields, etc.)

# prototyping final result

- All stake-holders signed off on design
- We got back on schedule for construction
- We gained twice the understanding
  in half the time

# wish list

- more usability testing
- greater attention to accessibility
- more patterns for reuse from the start

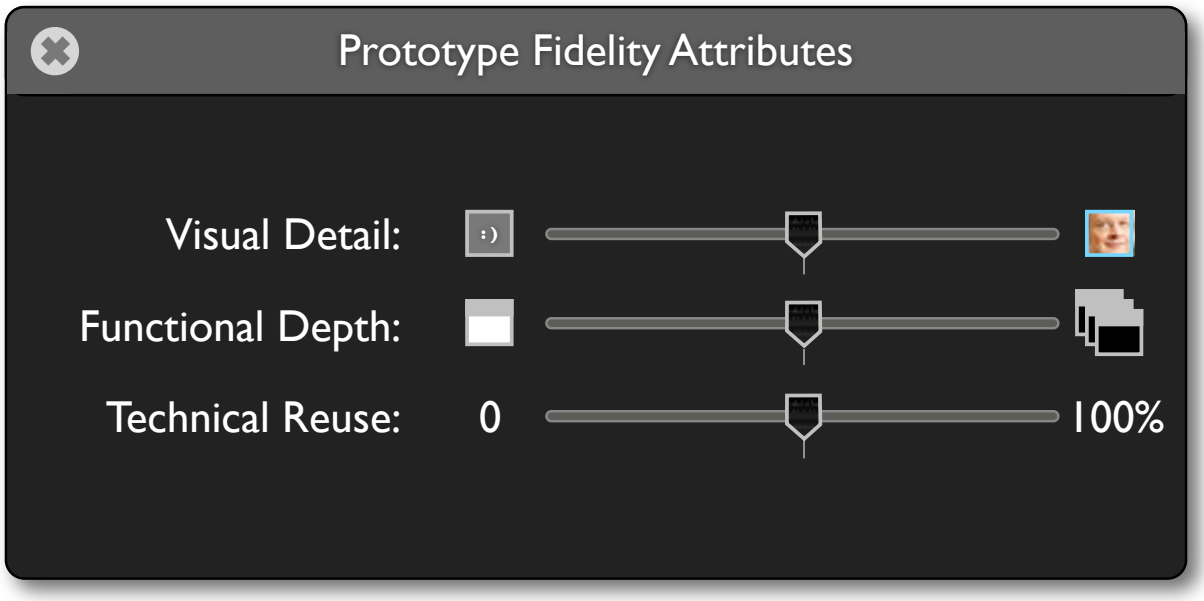


We didn't include any accessibility planning or testing.

I wish we had a "small" patterns library to pull from (basic CSS, global nav, commons forms, etc.).

Only on-page tab behavior was working ... all other client-side behavior was faked or implied.

# summary






Your prototype is a deliverable to an interim set of users.

Be user-centric.

Use a high-fidelity style that suits your interim users' needs.


# useful links

Visit http://markup.thekraemers.com now for links to:

- Henrik Olson's "Balancing fidelity in prototyping"
- Dan Brown's "Representing Data in Wireframes"
- Garrett Dimon's "Just Build It: HTML Prototyping and Agile Development"
- This deck (available by Sept 17, 2006)
- My del.icio.us feed on prototyping

Visit Mark-Up to learn more:

http://markup.thekraemers.com

Drop me a line with questions or comments:

mark@thekraemers.com