Ongoing Software Development without Classical Requirements

Thomas A. Alspaugh and Walt Scacchi
University of California, Irvine

Without requirements, one would expect …

- Projects whose products fail to meet stakeholder needs
- Or fail to be reliable, evolvable, or exhibit other desired software qualities
- Schedule slips, budget overruns
- In extreme cases, failure to produce any usable product at all

Yet open-source software (OSS) development works

- GNU/Linux
- Apache HTTP server
- PostgreSQL database system
- Mozilla Firefox browser
- Eclipse development platform
- If these are the kinds of system you get from *not* doing requirements ...

But not always
(though it's hard to compare)

- Those systems are the exception; most OSS projects do not flourish ...
  but note that starting an OSS project is trivial
- Difficult to compare, but OSS projects may fail at nearly twice the rate of CSS (closed-source software) projects ...
  but note that the criteria differ for CSS and OSS
- We don't really have good data ...
  but it's the CSS data that is thin

A closed-source software (CSS) development context

- System produced by a development group
- For a client and users outside that group
- Developers may not have domain expertise
- Development against a budget and schedule
- Requirements as client-developer contract

An open-source software (OSS) development context

- System produced for developers' own use
- They are stakeholders and domain experts
- You want it, you code it up and argue for it
- Strong emphasis on extensibility
- Much discussion, little central control


Without overt
requirements artifacts
or processes

"Classical Requirements" for purposes of our study

- Requirements document or central requirements repository
- Requirements preferentially described in terms of the problem space rather than the solution space
- Requirements processes for completeness, internal consistency, and external consistency with stakeholder needs and domain

Research questions

- RQ1: To what extent do OSS projects in fact use Classical Requirements?
- RQ2: Where OSS projects do not use Classical Requirements, what artifacts and processes are used instead, if any?

Data

- Our previous work
- All results reported by other researchers (five reports), with some data re-examined in its original context
- Newly-collected data
- All are ongoing projects
- Twelve projects examined closely, others less so

What did we find?

- Immediately recognizable: Artifacts and processes for *feature requests* and for *bug reports*
- Discussions on email lists, electronic bulletin boards
- Discussion focus: architecture, implementation
- Requirements considered at best indirectly, and stated implicitly

One (1) self-described OSS requirements document

- Firefox2 [2006], first reported by Noll [2008]
- High-level, non-specific *provisionments* (next slide):

Bon Echo will update its appearance to look and feel like a modern native application on all platforms. Incremental polish and refinement to the user interface will focus on improving the usability and accessibility of primary product features.

- Subdivided, allocated to milestones, given priorities; project management document
- We found *no other OSS requirements document*

A provisionment

- Points to an implementation, and sketches the kind of behavior that is referred to
- “Go play with this implementation and see what it does”
- Provisionments *very* widely used,
  either as the statement itself
  or as the starting point for a stated variation

A *way of expressing*, not a *kind of thing to express*

- A feature request, a bug report, a software quality: kinds of things to express
- A provisionment:
  a way of expressing something
- The "something" can be a feature, a bug report, a software quality ... or a requirement

Provisionments in use

"Have you tried tabbed browsing [in the Opera web browser]? Now that I've tried it, I won't go back to windows everywhere. The idea is that pages have their own tabbed windows. Instead of juggling windows, you just click their tabs. The beauty part is new pages open in the background, just as you requested. The tab tells you when the page is done loading. Then you just click over. Shweet!"

(in a Mozilla newsgroup 1999, quoted by Noll 2007)

More provisionments

- "You could add a link to the existing superbill page ..." *(feature request in terms of a stated difference from a specific version of the system)*
- "I think that existing Firebug users would complain if the Profiler is removed or providing [sic] different kind of results." *(could be proposing a requirement, stated negatively in terms of specific version of the system)*

Are provisionments ...

- *advantageous?* They can be, as a compact representation of projected future complex behaviors described in terms of current ones
- *limited to OSS?* No, we have seen them in CSS development and heard reports from others
- *good requirements?* Not by classical standards; they are solution-space not problem-space, and they define in terms of an implementation

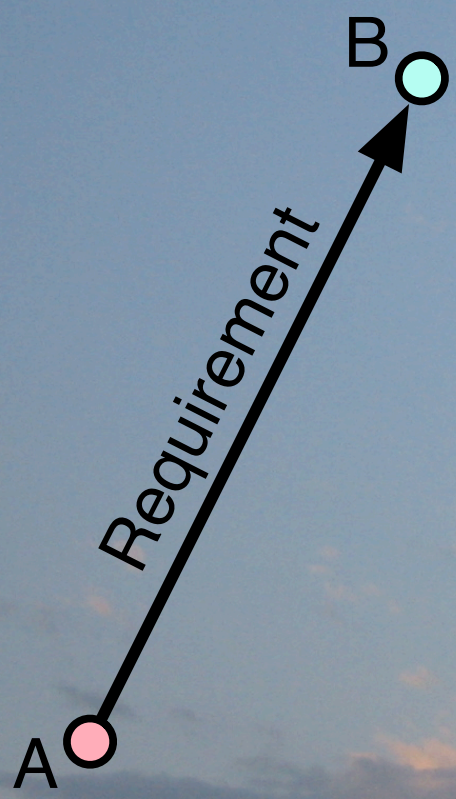
Study results

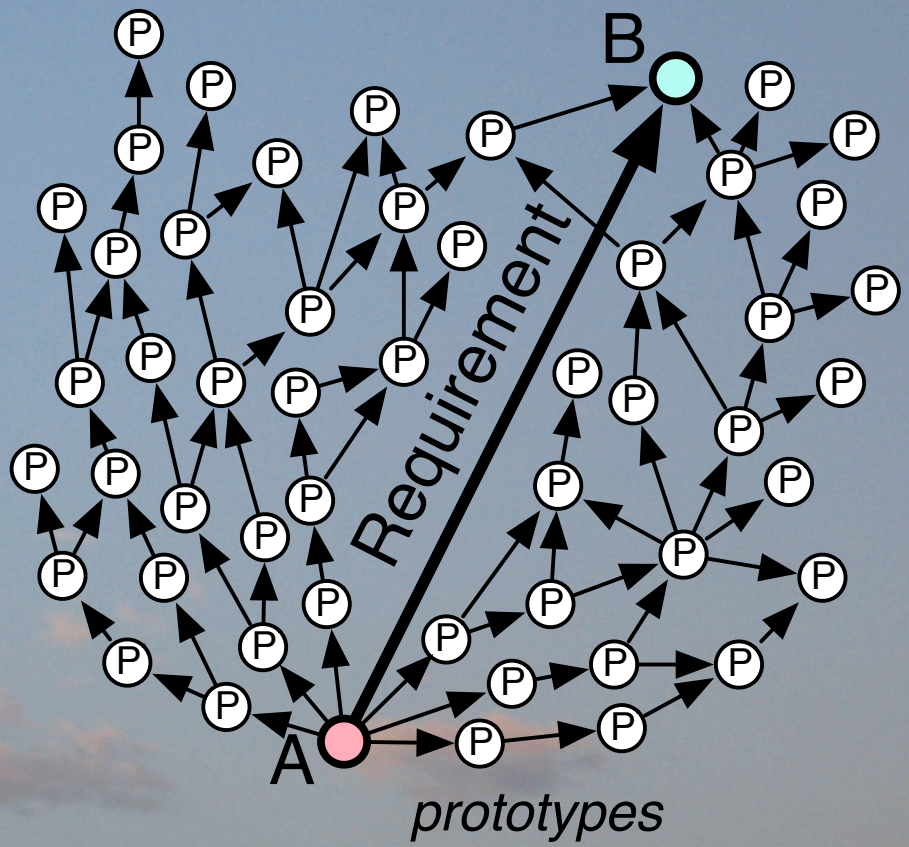
- (RQ1) Classical Requirements almost completely absent from OSS projects
- (RQ2) Instead of Classical Requirements artifacts, we saw provisionments
- (RQ2) Instead of Classical Requirements processes, we saw (most commonly) solution-space discussions of provisionments

Discussion

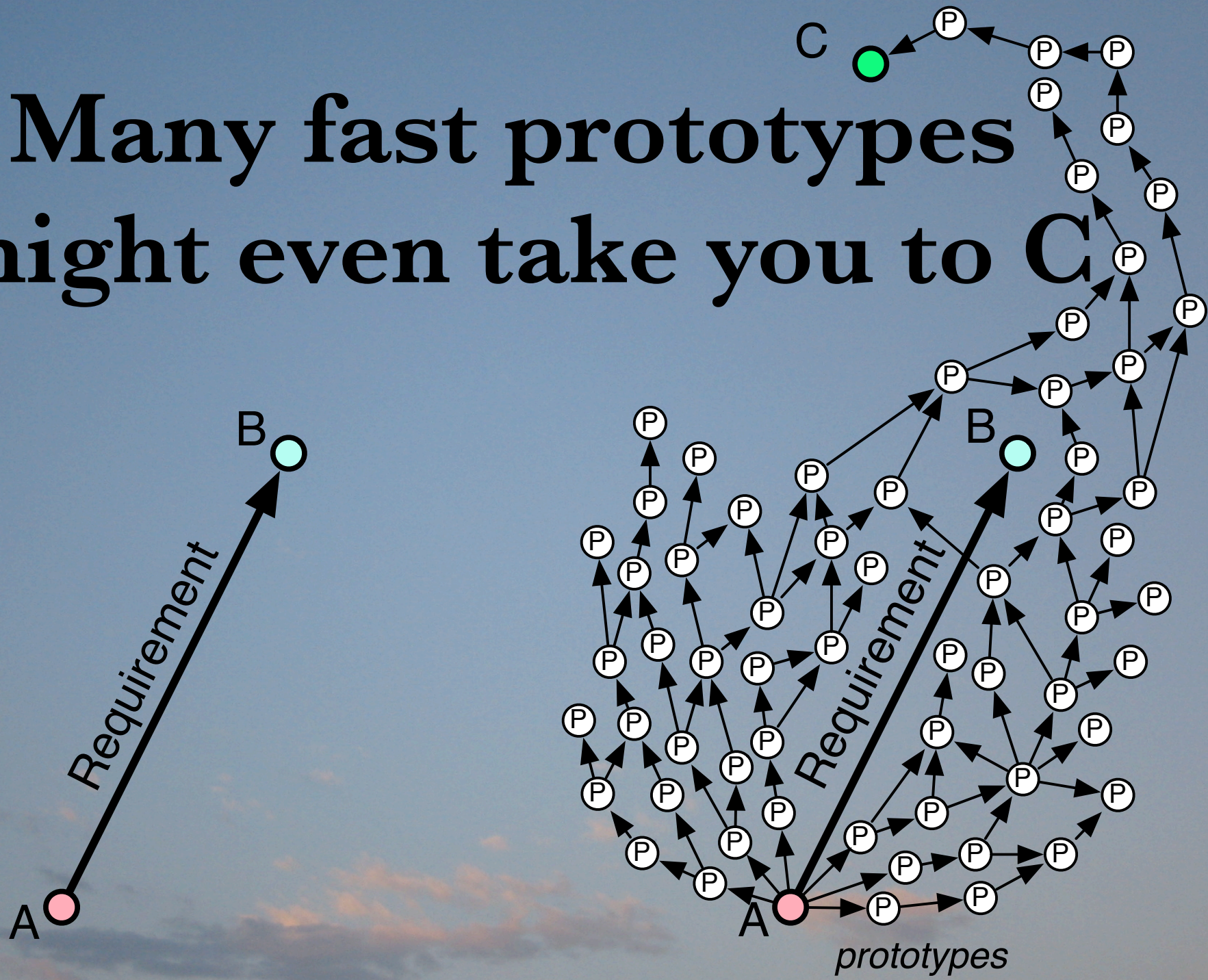
- Different means to achieve the same goals
- No definitive statement in one place of what the system is or is not to do
- Much architecture/implementation discussion
- Requirements foresight, insight, creativity perhaps replaced by lots of prototypes produced quickly

Two ways from A to B





Many fast prototypes might even take you to C



Ongoing Software Development without Classical Requirements

Thomas A. Alspaugh and Walt Scacchi
University of California, Irvine