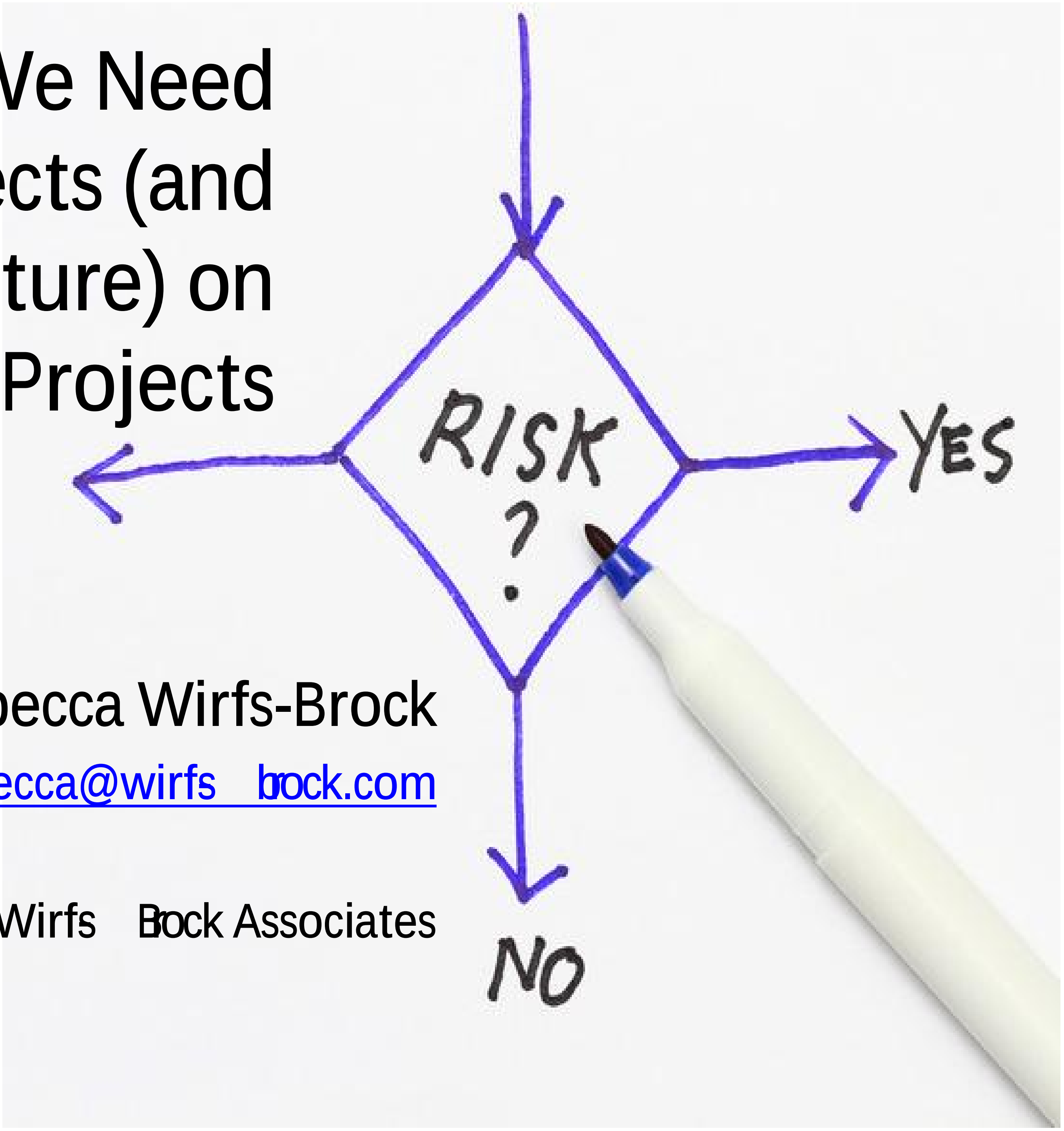
Why We Need Architects (and Architecture) on Agile Projects



Rebecca Wirfs-Brock

rebecca@wirfs-brock.com

Three Questions…

- If you are agile, how much architecting do you need and when?
- How can you manage architecture risk on large, complex agile projects?
- What is the role of an agile architect?

Agile Values

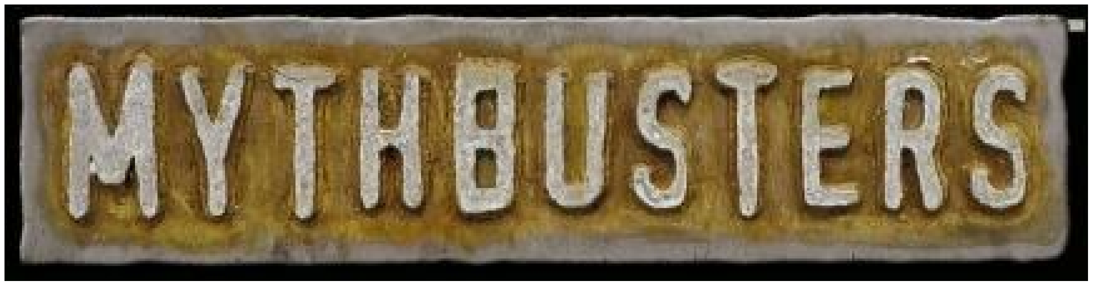
- Core values:
  - Design Simplicity
  - Communication
  - Teamwork
  - Trust
  - Satisfying stakeholder needs
- Constant learning

Qualities of Good Agile Architecture

- Designed for test.
- Modular.
- No unintentional data redundancy or overlapping functionality.
- Pragmatic. Does what it needs to without extras.
- Supports performance, reliability, modifiability, usability,….goals.

Agile Misconception: Simple Design is Always Best

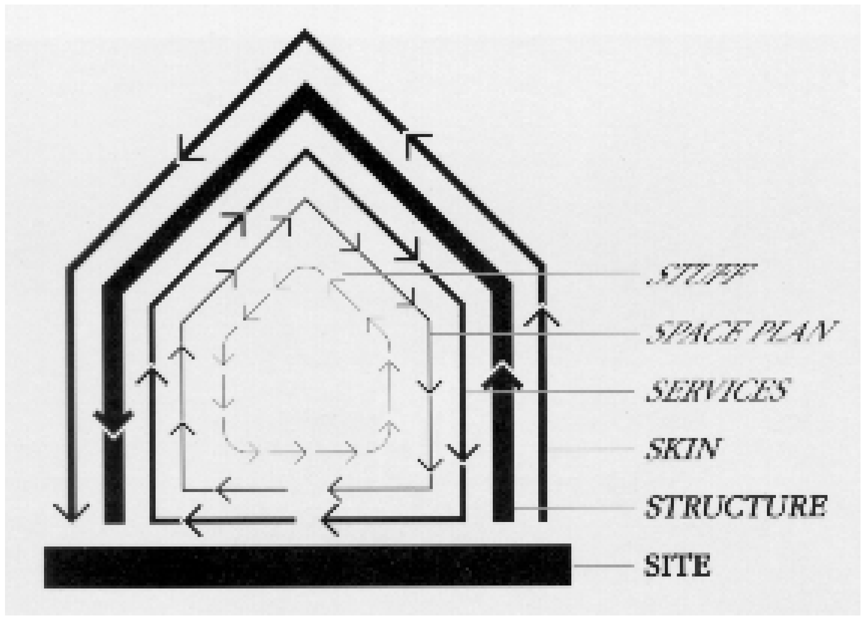
- Does that mean you should never…
  - create a framework?
  - write code that needs comments?
  - never implement a complex solution?
  - anticipate future features?



Stuart Brand's Shearing Layers

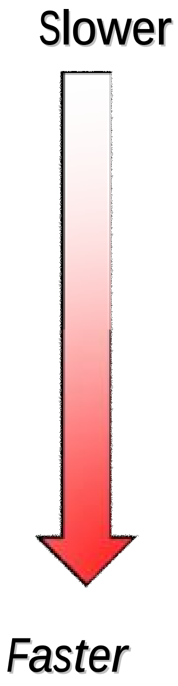
- Buildings are made of components that evolve at different timescales.
- Layers: site, structure, skin, services, space plan, stuff. Each layer has its own value, and speed of change (pace).
- Buildings adapt because faster layers (services) are not obstructed by slower ones (structure).

—Stuart Brand, *How Buildings Learn*



Yoder and Foote's Software Shearing Layers

"Factor your system so that artifacts that change at similar rates are together."—Foote & Yoder, Ball of Mud Pattern



Layers

- The platform
- Infrastructure
- Data schema
- Standard frameworks and components
- Abstract classes and interfaces
- Classes
- Code
- Data

Agile Design Values

- Respect your system’s shearing layers.
  - Understand the rates of what changes.
- Make what is too difficult, time consuming, or tedious easier.
  - Create tools, leverage design patterns, build or use frameworks, use data to drive behavior…
- Don’t overdesign!!!
- Don’t under architect.

The Boundary Between Architecture and Design

Architecture

- "Architecturally significant" design issues
- Balances big picture and details
- Considers many factors

Design

- Designs and implements solutions
- Makes detailed decisions
- Primarily focused on technical concerns

How Much Architecting Do You Need?

Project Criticality

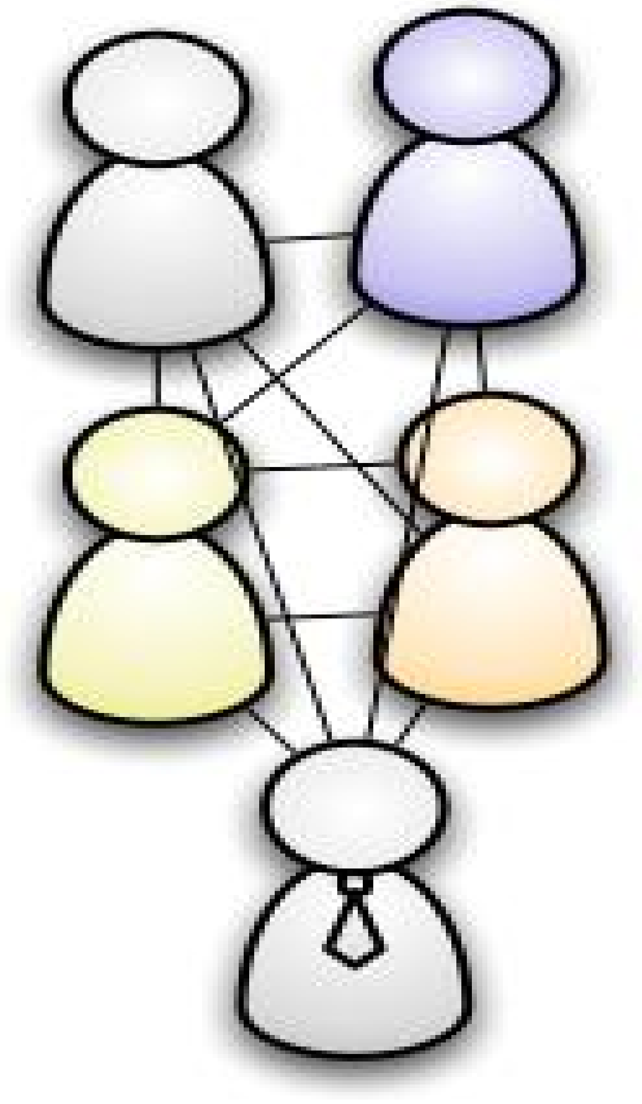
Life	L6	L20	L40	L100	L200	L1000
Essential money	E6	E20	E40	E100	E200	E1000
Discretionary Money	D6	D20	D40	D100	D200	D1000
Comfort	C6	C20	C40	C100	C200	C1000
	1-6	- 20	-40	-100	-200	-1000

Project Size

Alistair Cockburn’s project characteristics grid

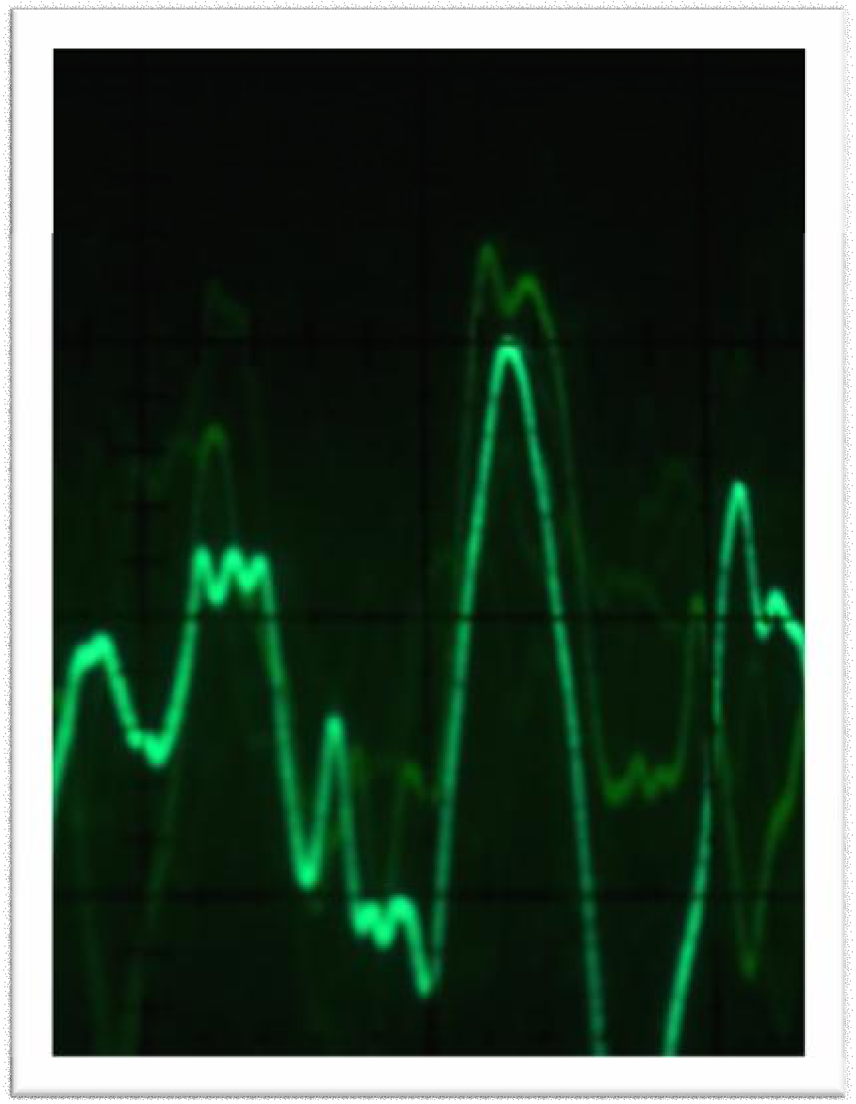
What’s a Small Project?

- A team of 6-8
- Working on non-life critical projects
- Architecture typically evolves along with implementation without much risk
- May or may not need extra architecture attention



Small Project Architecture Practices

- Design “Spikes”
  - Goal: Figure out a design approach.
  - Time: Few hours to a few days.
  - Tools: CRC Cards, exploratory coding, whiteboard sketching.



Small Project Architecture Practices

- Experiment on Branches
  - Goal: Experiment away from main code branch.
  - Time: Few hours to a few days.
  - When done: Merge or throwaway branch code.

Small Project Architecture Practices

- Incrementally refine abstractions
  - Goal: Refactor to eliminate redundant code.
  - Time: Few minutes.
  - When done: Whenever you spot duplication.

Small Project Architecture Practices

- Monitor technical debt.
  - Term invented by Ward Cunningham.
  - Piles up when you continually implement without going back to reflect new understanding.
  - Can have long term costs and consequences.

All Tasks Aren’t Alike

- The Core—fundamental to your software’s success
- The rest—requires far less creativity or inspiration
- The Revealing—lead to new, deeper understanding
  - Always a surprise
  - Require invention and innovation
  - Hard to predict when they will be done

Keeping Architecture in Mind

- Sort tasks into “problem buckets”: core and the rest
- Make sure each iteration gets enough core work accomplished
- Get team involved on core issues
- Use post-iteration reflections to ask why things were harder than they first appeared
- Break out of planned iteration cycles to tackle revealing problems (they need more than a quick design spike)

The Bigger the Project….

THE MORE THERE IS TO COORDINATE

Agile Misconception: Upfront Thinking, Planning, Investigating, Architecting is Wasteful

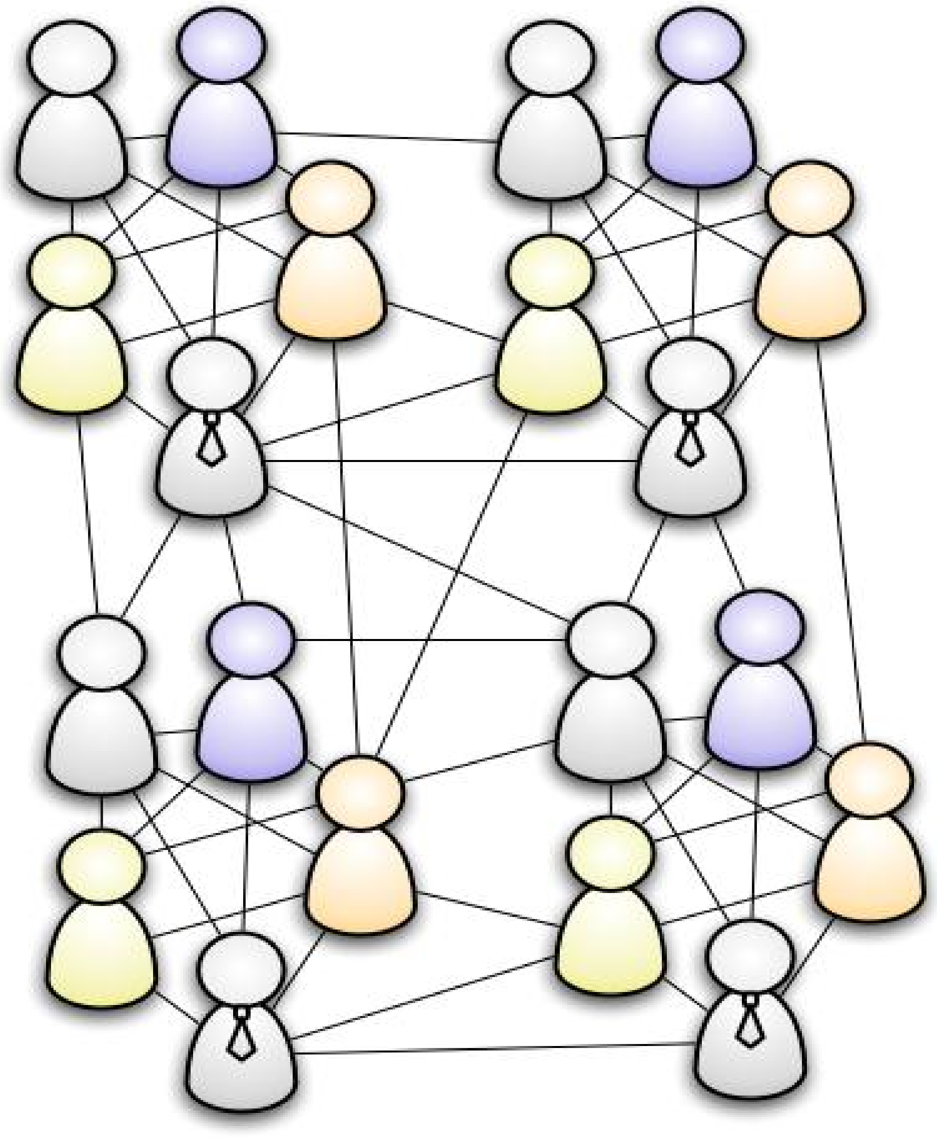
- A reaction to “too much” thinking without “doing”.
- Reality:
  - You need to strike a balance: Find the right time and effort for your project
    - Some upfront planning is critical for large, complex projects
    - Ongoing thinking, prototyping, and architecture experiments are important too.

A Better Way to Act:

CHOOSE THE MOST RESPONSIBLE MOMENT

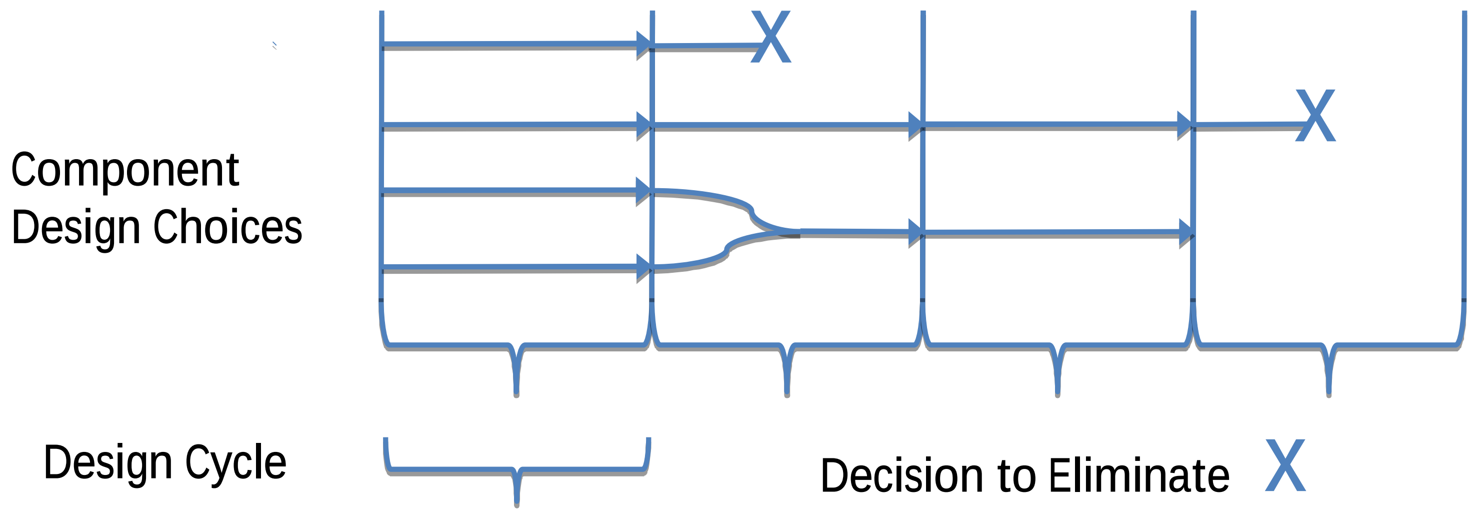
Team Size Matters

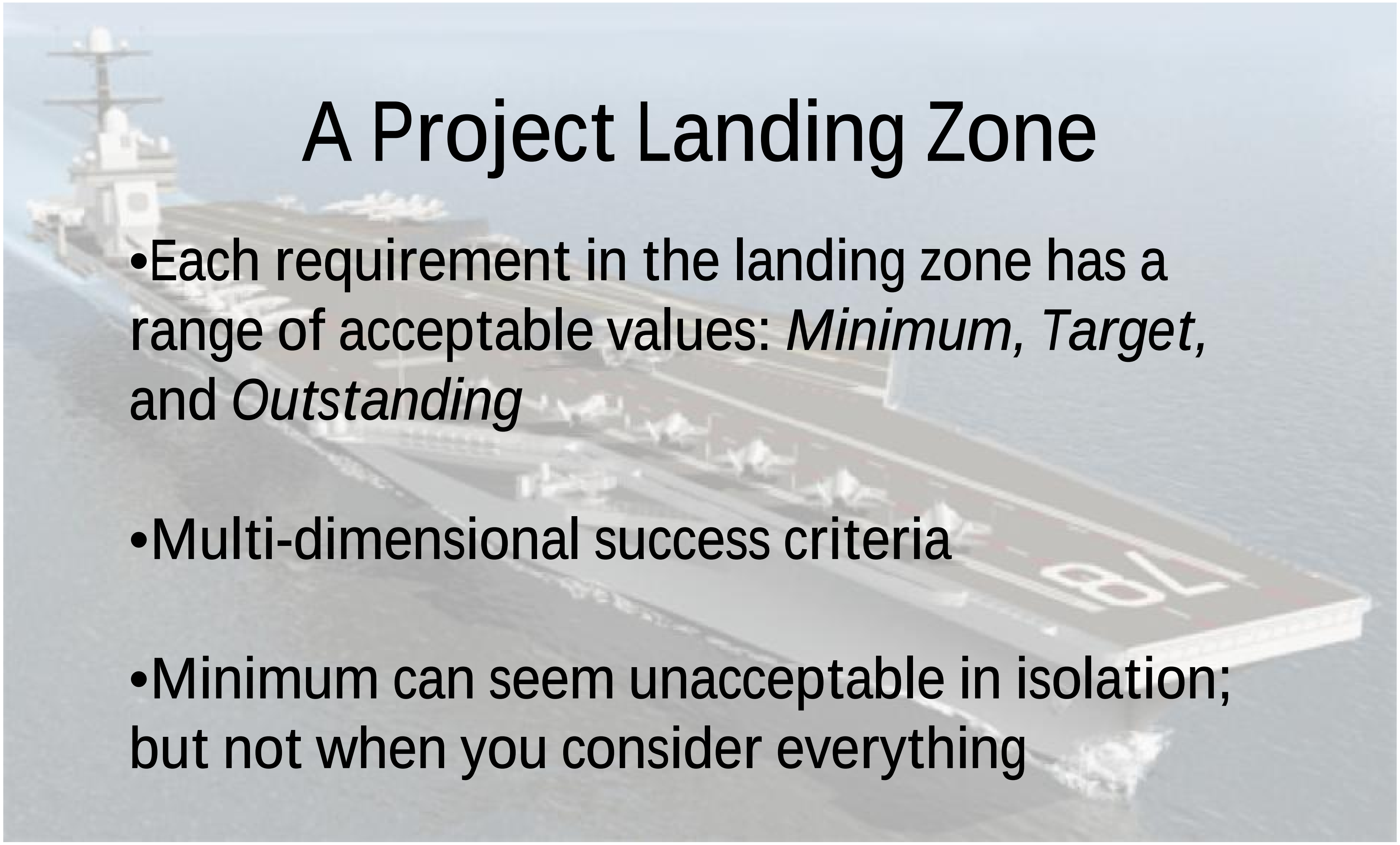
- >9 and any group splits into teams
- No one knows everything or everybody
  - Expertise uneven
  - Skills varied
  - Specialists
- Work needs coordination
- Architecture allowed to "naturally" emerge often reflects the organizational structure



Architecture Risk Reduction Tools

- Project/product road maps and timelines
- Landing zones
- Design innovation spikes
- Architecture spikes
- Risk reduction backlogs
- Set-based design





A Project Landing Zone

- Each requirement in the landing zone has a range of acceptable values: *Minimum, Target,* and *Outstanding*

- Multi-dimensional success criteria

- Minimum can seem unacceptable in isolation; but not when you consider everything

A range of measurable attributes that must be achieved to declare project or product success

Landing Zone Precision & Granularity

Attribute	Minimum	Target	Outstanding
Data Quality: Accuracy (percent in error) for critical data attributes	<2.5%	1.5%	0.5%
Performance: loan payment transactions per hour	60,000	75,000	100,000
Usability: Learning loan management system tasks by a new quality analyst	< 16 hrs	8 hrs	4 hrs

Landing Zones & Architecture

Photo by e.r.w.i.n. Used with attribution
http://www.flickr.com/photos/eherrera/5104896694/

- Identify and manage
  - Potential risks
  - Innovations required
  - Skills to be acquired
  - ...

Landing Zones on Agile Projects

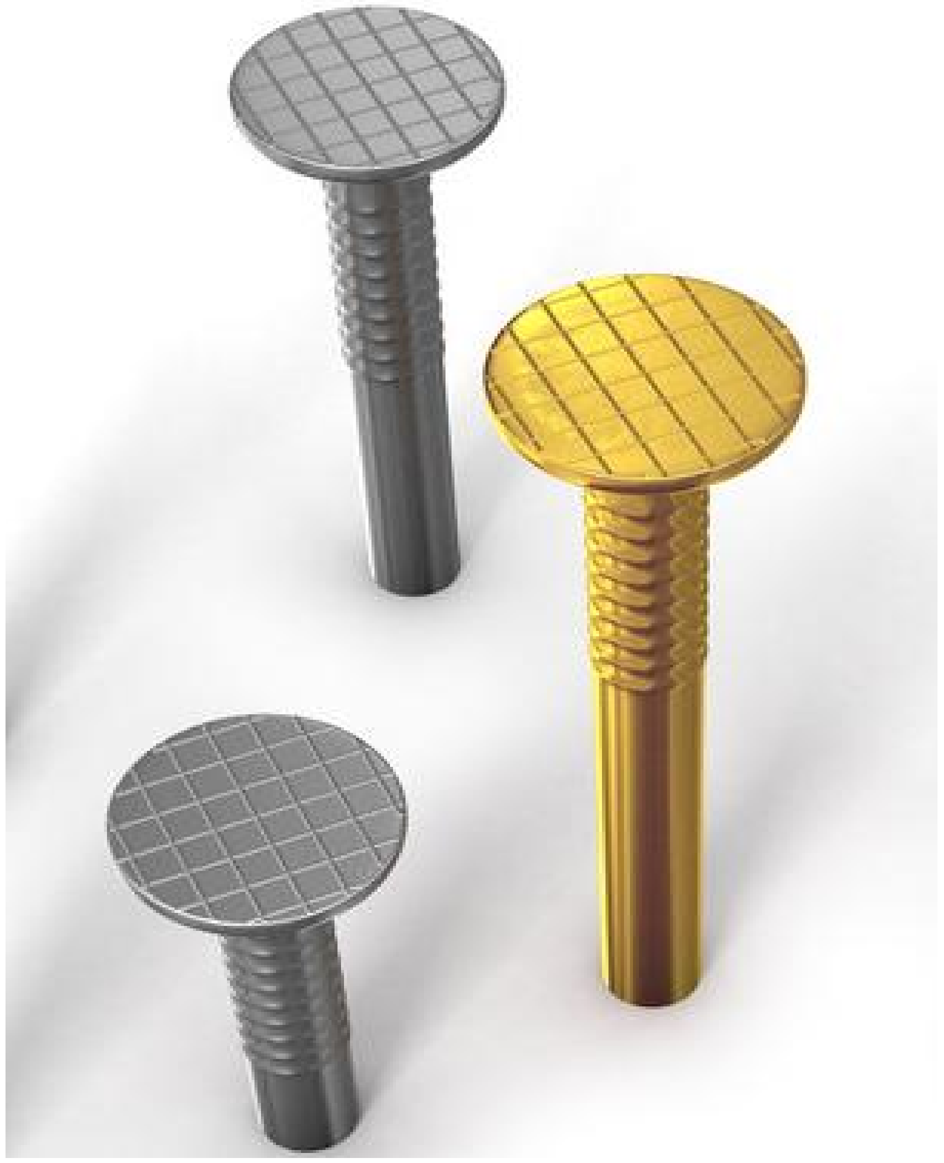
- Helps make sense of the bigger picture:
  - What happens when one attribute edges below minimum?
  - When will targets be achieved?
  - What do we need to do architecturally to achieve targets?

Design Innovation Spike

- Answers deep questions about potential solutions for achieving landing zone targets
- Not as tactical or incidental as an XP Design Spike

What You Do In an Innovation Spike

– prototyping
– design noodling
– looking outside
– experimenting
– modeling
– vet ideas



Example Innovation Spikes

- Business transaction redesign
- Document parsing
- Fact representation & rule simplification
- Automated location of external resources
- ...
- Scale up, scale out, re-distribute, re-think...
  - Try out radical changes in how things are done

Design Innovation Spike Best Practices

- Small, smart, goal-oriented teams
  - avoid us vs. them mentality
- Evidence-based answers
  - working prototypes
  - existing similar things
- Failure is an option
  - permit answers that shift goals

Criteria For an Architecture Spike: *Answer Bounded Questions*

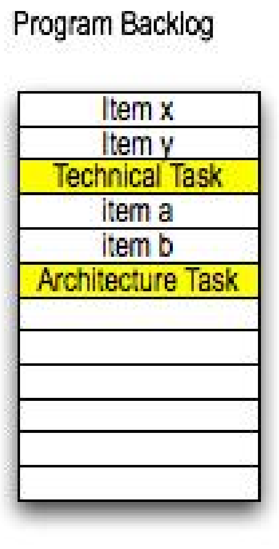
- Buys information
  - Feasibility
  - Rework effort
  - Reasonable design approach
  - Alternatives
- Better estimates
- Actionable

Architecture Debt

- Compromises in the system that have significant impacts.
- Not isolated.
- Difficult to reverse.
- Examples:
  - reliance on a poorly designed framework
  - inconsistent service interfaces

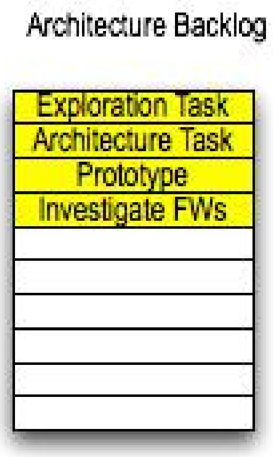
Ways To Manage Architectural Tasks

1. Add to Program Backlog



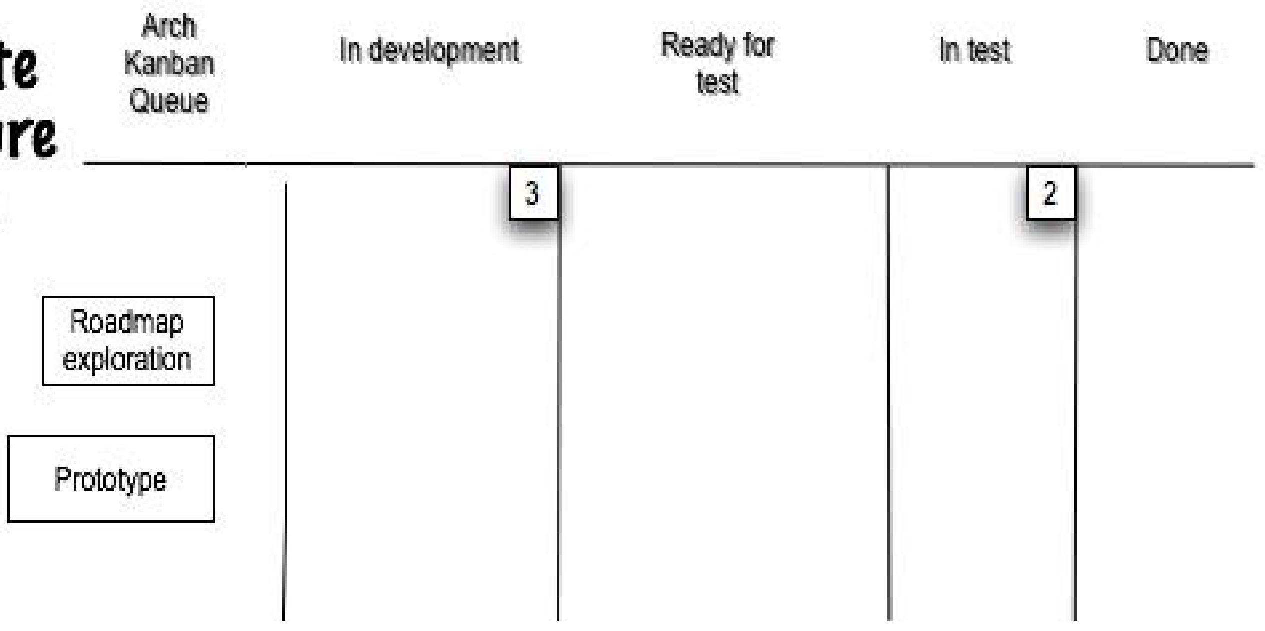
Program Backlog
Item x
Item y
Technical Task
item a
item b
Architecture Task

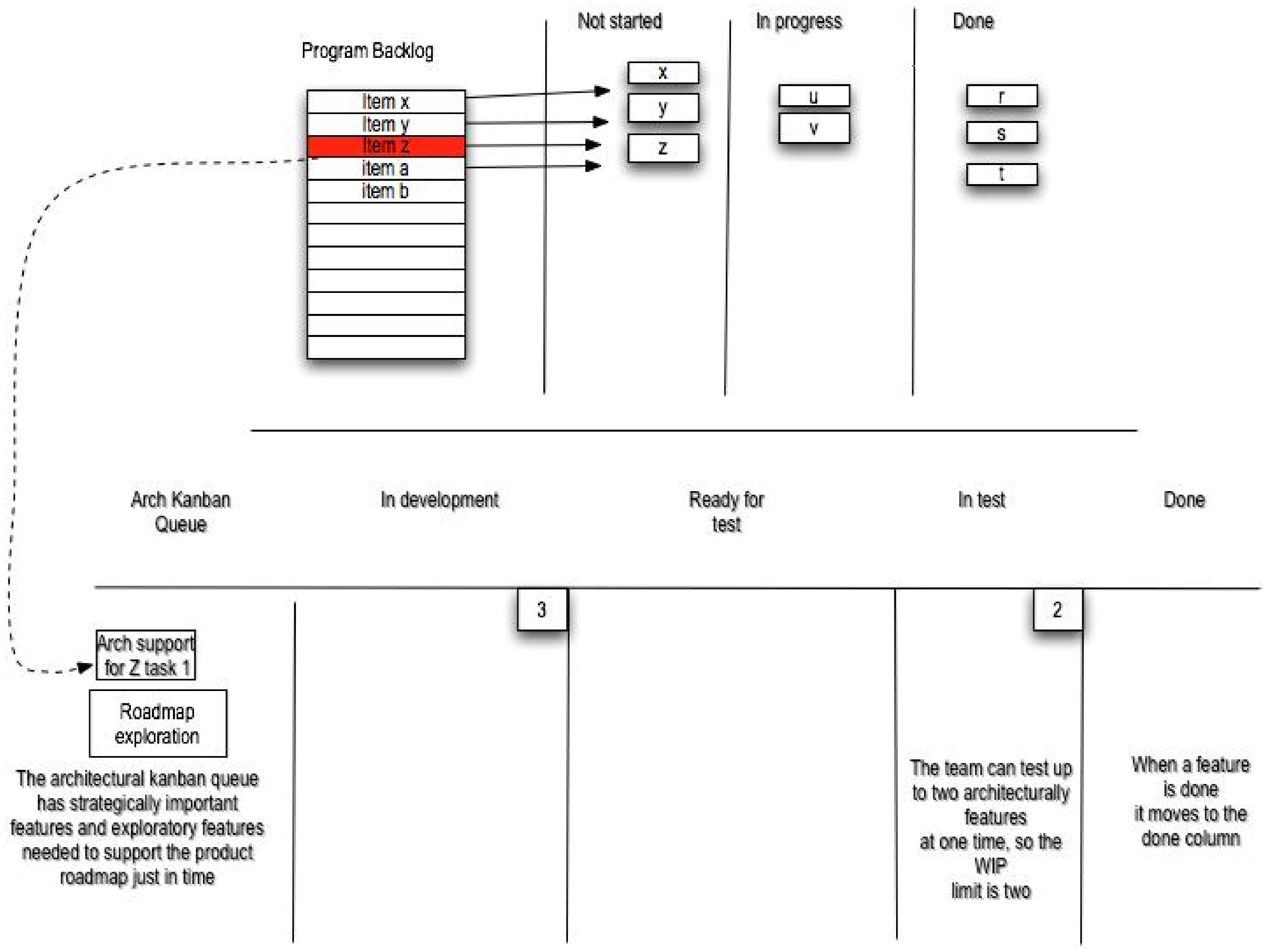
2. Separate Architecture Backlog

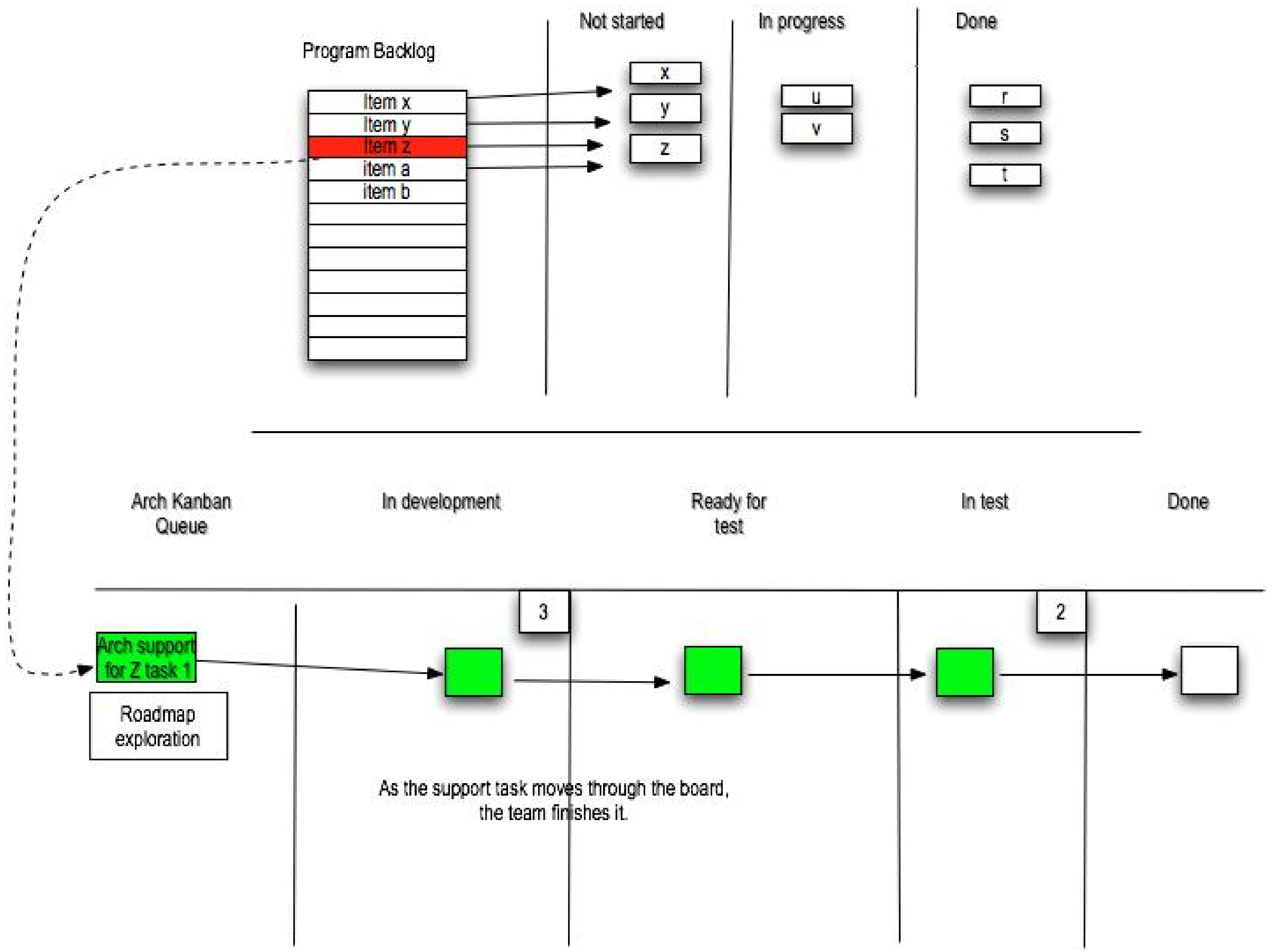


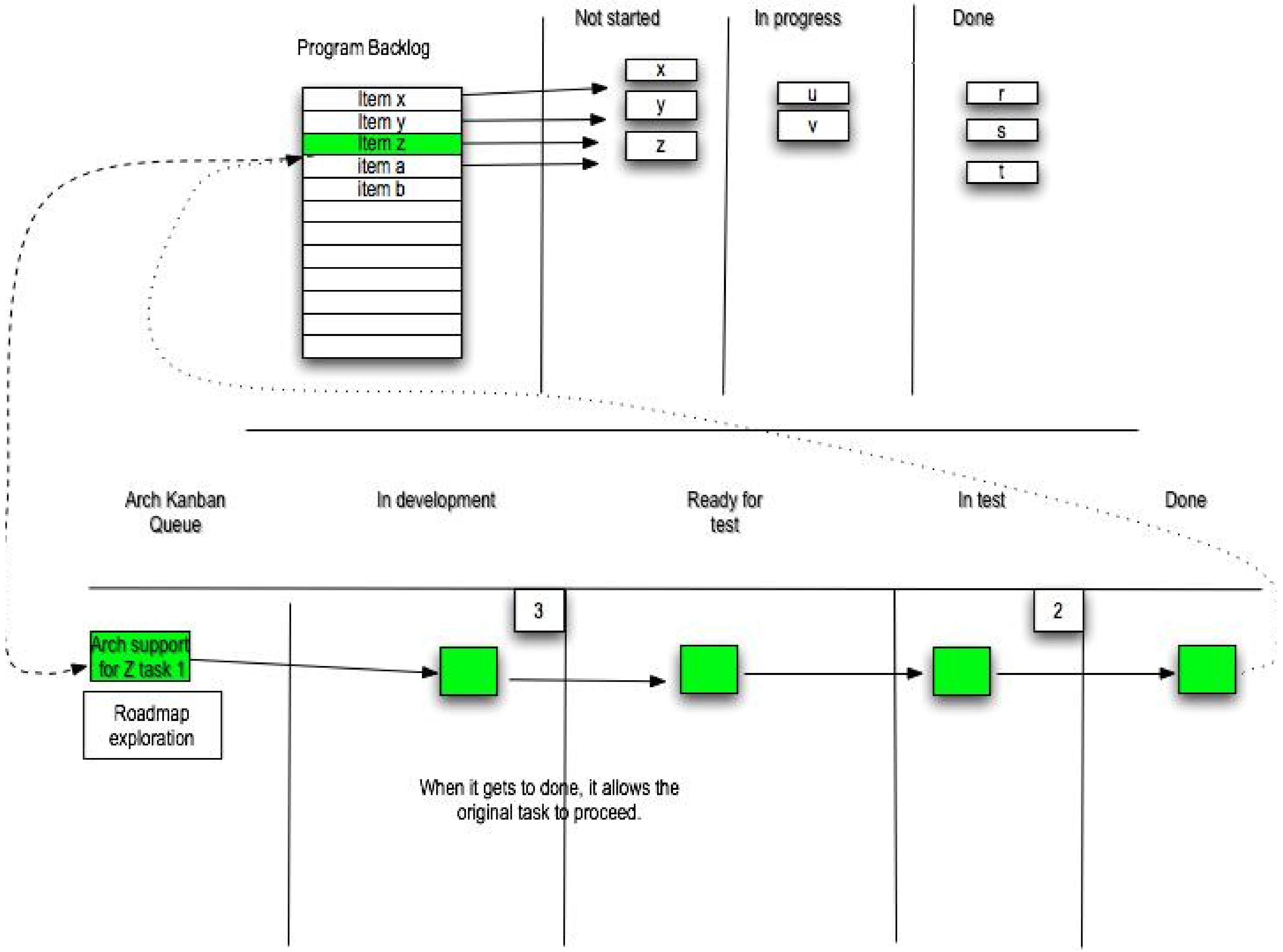
Architecture Backlog
Exploration Task
Architecture Task
Prototype
Investigate FWs

3. Separate Architecture Kanban




Program Backlog
Not started
In progress
Done
Item x
Item y
Item z
item a
item b
x
y
z
u
v
r
s
t
Arch Kanban Queue
In development
Ready for test
In test
Done
3
2
Arch support for Z task 1
Roadmap exploration
The architectural kanban queue has strategically important features and exploratory features needed to support the product roadmap just in time
The team can test up to two architecturally features at one time, so the WIP limit is two
When a feature is done it moves to the done column


Program Backlog
Not started
In progress
Done
Item x
Item y
Item z
item a
item b
x
y
z
u
v
r
s
t
Arch Kanban Queue
In development
Ready for test
In test
Done
3
2
Arch support for Z task 1
Roadmap exploration
As the support task moves through the board, the team finishes it.


Program Backlog
Not started
In progress
Done
Item x
Item y
Item z
item a
item b
x
y
z
u
v
r
s
t
Arch Kanban Queue
In development
Ready for test
In test
Done
3
2
Arch support for Z task 1
Roadmap exploration
When it gets to done, it allows the original task to proceed.

What Can Go On An Architecture Backlog?

Architecturally meaty feature

Design spike related task

Architecture investigation

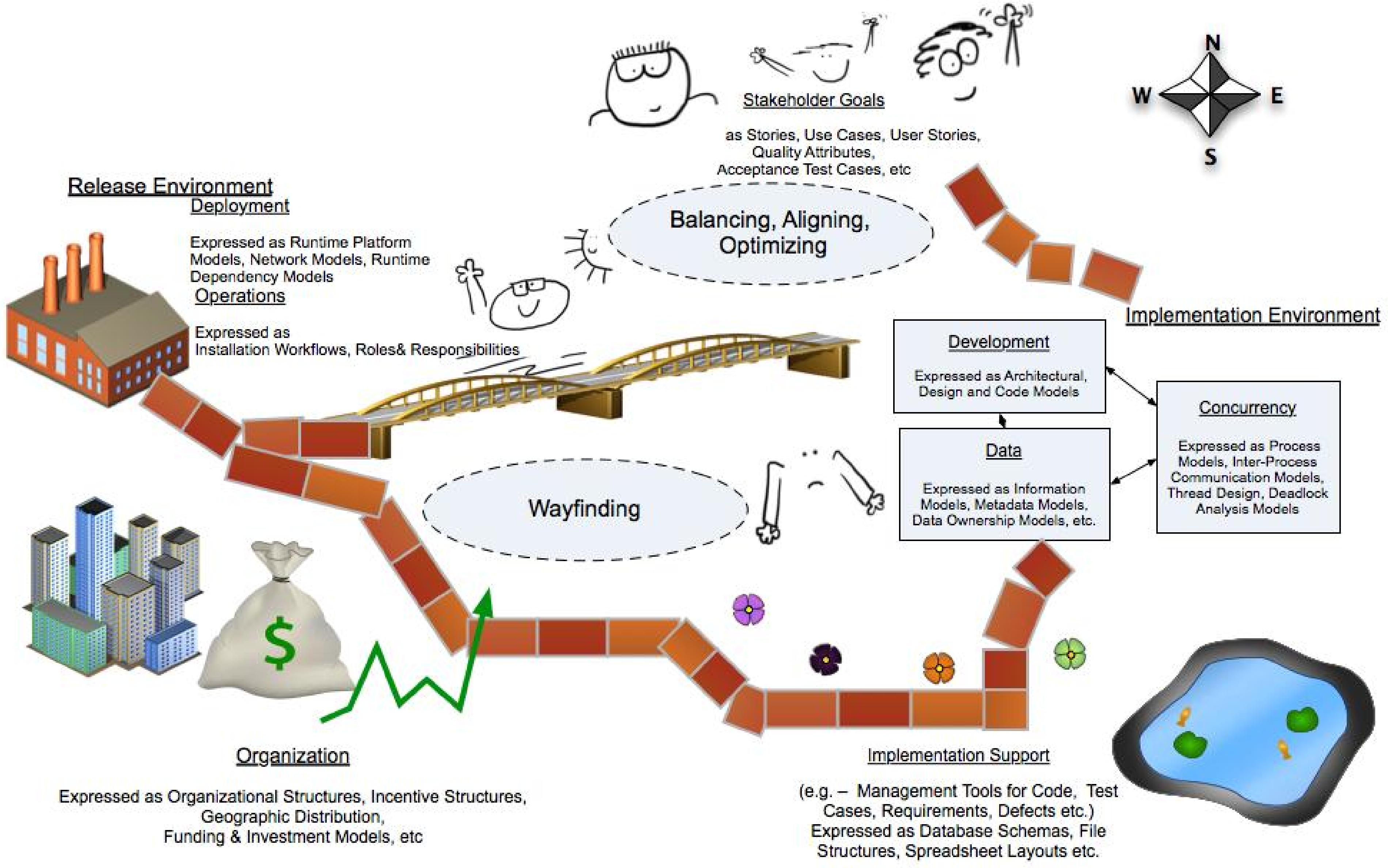
Prototype

Framework development

Roadmap exploration

WHAT DO AGILE ARCHITECTS DO?

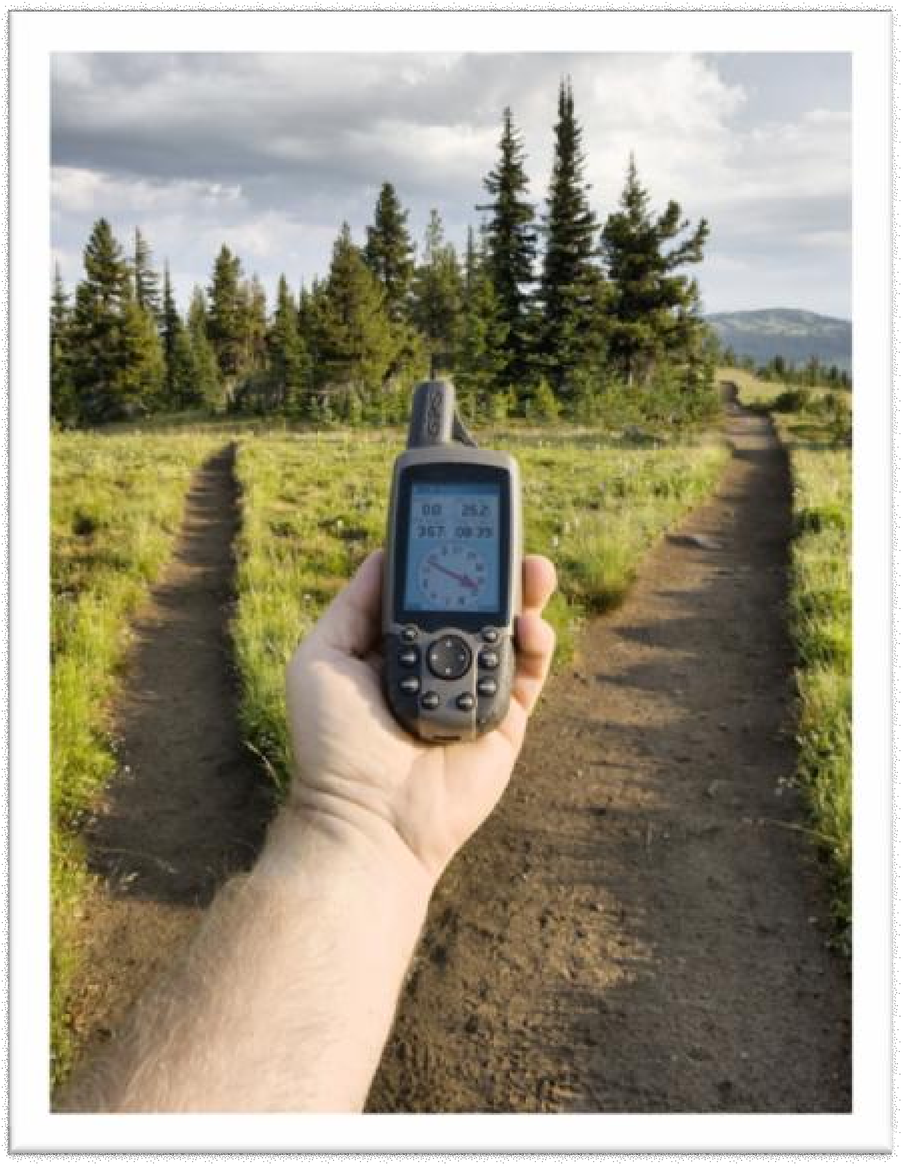
The Agile Architecture Landscape



Stakeholder Goals
as Stories, Use Cases, User Stories, Quality Attributes, Acceptance Test Cases, etc
N
W
E
S
Release Environment
Deployment
Expressed as Runtime Platform Models, Network Models, Runtime Dependency Models
Operations
Expressed as Installation Workflows, Roles& Responsibilities
Balancing, Aligning, Optimizing
Implementation Environment
Development
Expressed as Architectural, Design and Code Models
Concurrency
Expressed as Process Models, Inter-Process Communication Models, Thread Design, Deadlock Analysis Models
Data
Expressed as Information Models, Metadata Models, Data Ownership Models, etc.
Wayfinding
$
Organization
Expressed as Organizational Structures, Incentive Structures, Geographic Distribution, Funding & Investment Models, etc
Implementation Support
(e.g. – Management Tools for Code, Test Cases, Requirements, Defects etc.) Expressed as Database Schemas, File Structures, Spreadsheet Layouts etc.

Agile Architecture Wayfinding

- Scouting—looking enough ahead
- Exploring potential paths
  - Short experiments
  - Extrapolations
  - Conclusions based on experience, intelligence gathered & intuition
- Explaining and selling architectural ideas



Differences Between Agile and Traditional Architecture

Traditional

- Big picture thinking
- Produces Models and blue prints
- Not so hands-on
- Focused on compliance

Agile

- Balances big picture & details
- Produce what's needed to make informed decisions
- Hands-on
- Focused on sustainability

Models

“Big M” vs. “little m”

Lots of time invested	Not a lot of time invested
Intended to last	Intended to communicate
“Definitive”	Often discarded
Usually formal	Can be formal or informal
May not be widely used or understood	Made to be viewed

Agile architects create models as needed

CRC Cards: A “little m” model

The First CRC Cards

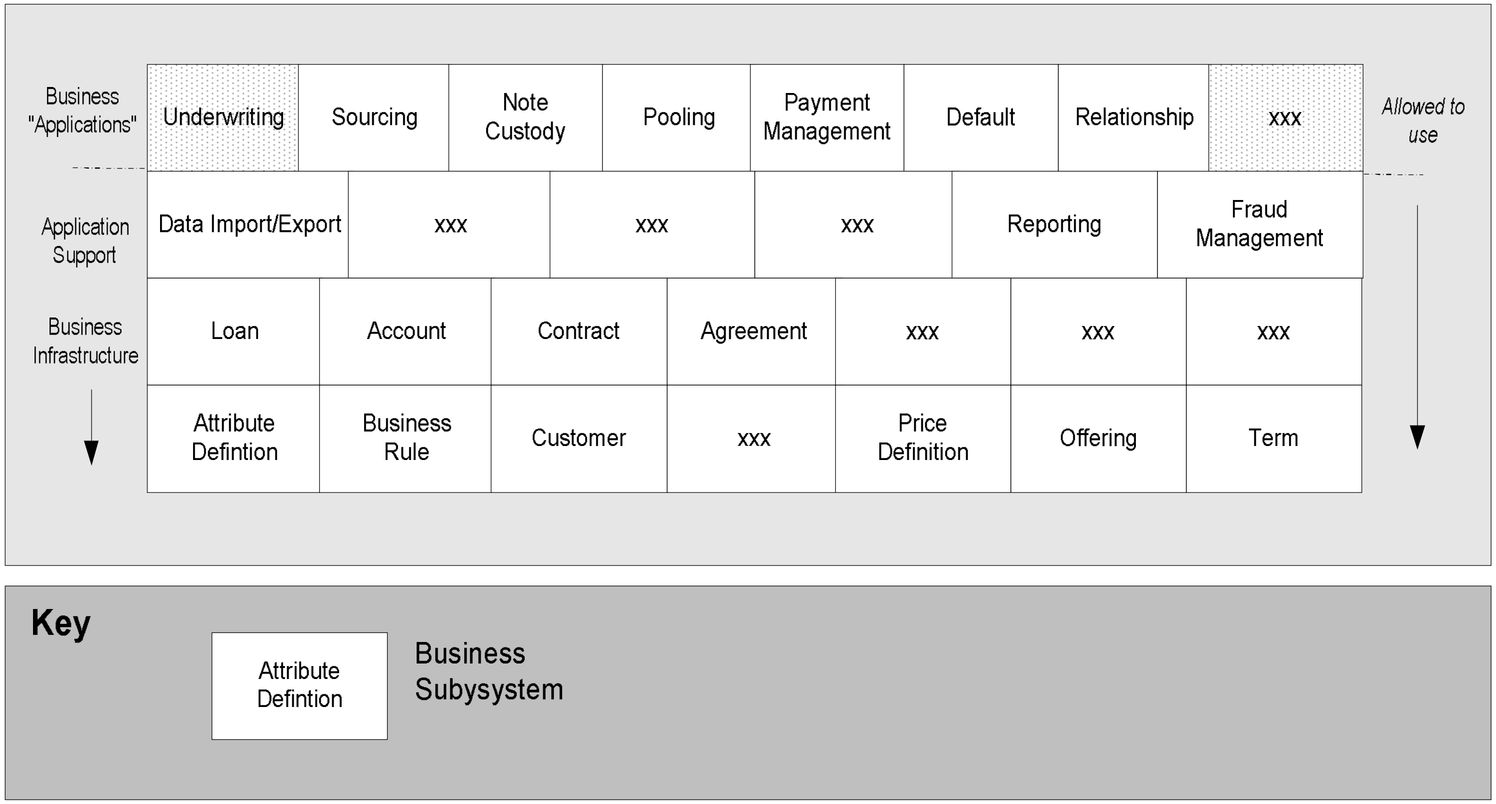
“A Laboratory For Teaching Object-Oriented Thinking,”
Kent Beck, Apple Computer, Inc., Ward Cunningham, Wyatt Software Services, Inc.
OOPSLA 89

Model	
Maintain problem related info	
Broadcast change notification	

View	
Render the model	Model
Transform coordinates	Controller

Controller	
Interpret user input	Model
Distribute control	View

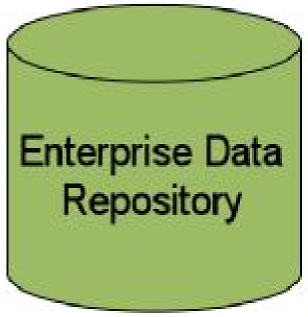
Example: Component Responsibility Descriptions



"The Customer component is responsible for knowing the organizations and individuals. It includes authentication and role-based authorization for detailed tasks and contact information for organizations."

Supports interactive web and self service applications
Provides storage for:
- Transactions that will affect systems of record
- Staging information closer to the user to support high performance access
- Data required by end users that comes from systems of record that do not have 24 x 7 availability

Supports the event driven and service integration architecture .
Provides storage for:
- transformation and enrichment services
- long running transactions.
- audit and performance metrics
- messages that need replayed in case of an unexpected failure
- error handling



Repository for those business entities that are shared across systems of record
- Customer is an example on such an entity
- Is responsible for managing the synchronizing those entities across systems
- Fundamentally a store for business identity management

Repository for business data and transactions
- Based on business processes
- Considered the single source of the truth as it relates to a given entity
- A given entity should have one and only one system of record

Example: Database "Responsibilities"

Supports capturing and storing data to support reporting and business analytics
Provides Storage for
- Time variant/non volatile data sourced from systems of record
- Historical record of transactional data
- Archival data for those systems of records not capable to support historical tracking of data

Indicators You've Paid Enough Attention to Architecture

- Developers can easily add new functionality.
- New functionality doesn't "break" existing architecture.
- Stable interfaces.
- Consistency.
- Few areas that developers avoid because they are too difficult to work in.
- Defects are localized.
- Able to incrementally integrate new functionality.

Values Important to Agile Architects

- Balance
- Testable architectural qualities
- Being hands-on
  - programming, designing, reading code, building things…
- Sustainable development

Sustainable Architecture

- Stewardship
  - Follow through
  - Ongoing attention
  - Not ignoring the little things that can undermine our ability to grow, change and adapt our systems

Thank you

-Rebecca
rebecca@wirfs-brock.com

The Responsible Designer Blog:
www.wirfs-brock.com/blog

www.wirfs-brock.com