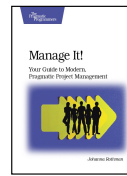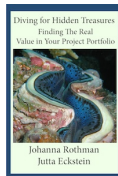# Scaling Agile Projects to Programs: Networks of Autonomy, Collaboration and Exploration
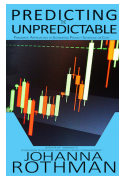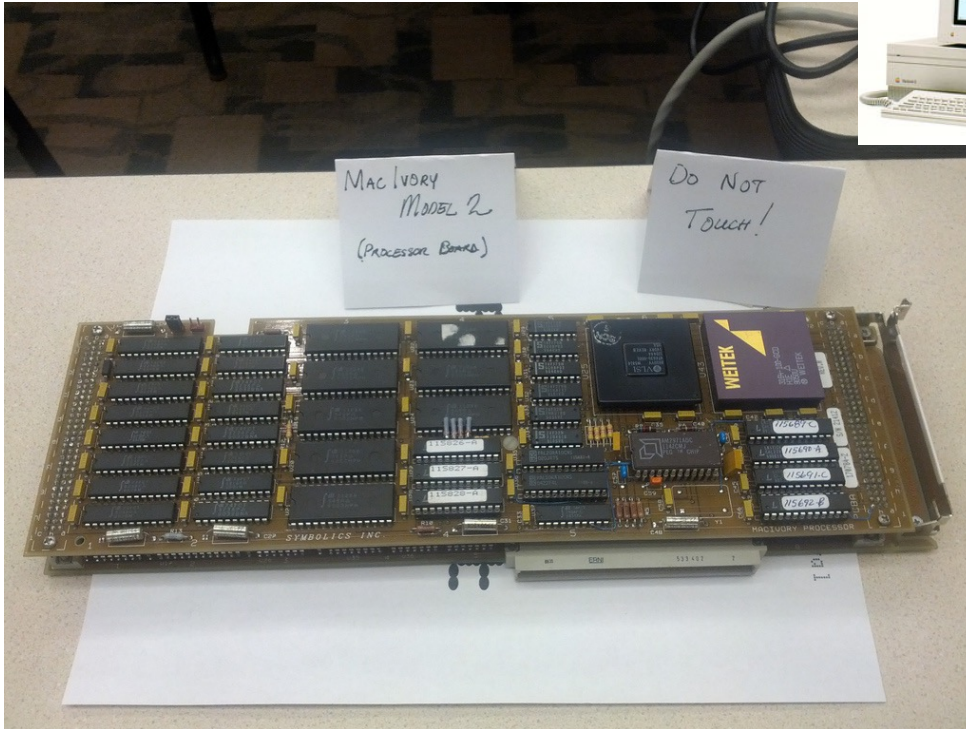
Johanna Rothman

*Agile and Lean Program Management: Scaling Collaboration Across the Organization*

@johannarothman

www.jrothman.com

jr@jrothman.com

781-641-4046
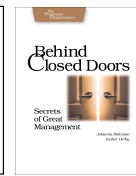
---

# Agenda

- **Program Managers and what programs look like**

- Agile roadmap

- Program teams and kanban

- Small-world networks

- Estimation

- Culture of delivery

# What Worked?

- Clear program vision

- Clear deliverables

- Feature teams who delivered at least every day

- Unit tests and system tests as support for collaboration

- Respect and trust across the organization

| Delivery |
|----------|
| Trust |
| Respect |

# Program Management

Program of Concurrent Projects



- Organizing and coordinating several projects' results into one deliverable

- That overall deliverable has the value to the organization
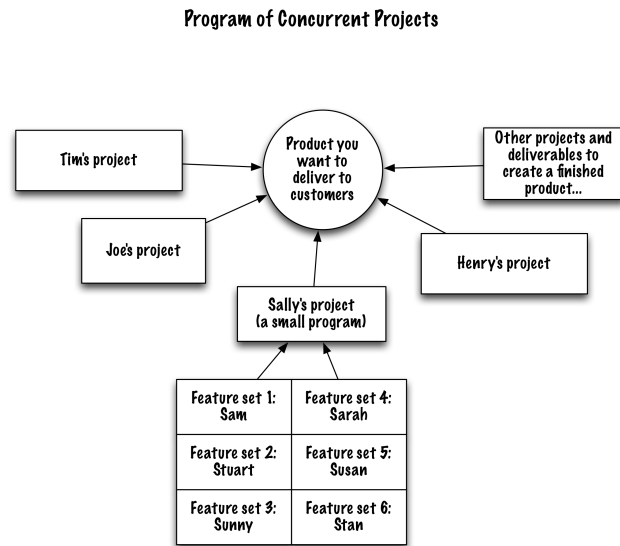
---

# Programs Are Riskier Than Projects

- Projects don't scale linearly

- The larger and the longer the program, the more risky it is

- The more pieces the program has, the more risky it is

# Integrated System Programs

- Possible examples:
  - Smart phone
  - Embedded system

**Integrated System Program**

| GUI (Common to all products) |
| :---: |
| API (Common API for the GUI) |

| Product 1 | Product 2 | Product 3 | Product 4 | Product 5 | Product N... |

**Platform of Common Services**
**(All products use this platform)**
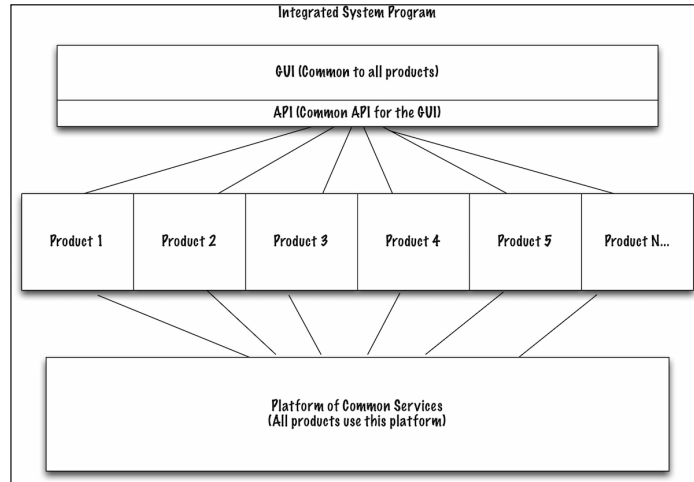
# Inter-Related Programs

- Possible examples:
  - Operating system and related products
  - Base product and layered products

**Inter-Related Product Program**

| GUI | GUI | GUI | GUI | GUI | GUI |
| :---: | :---: | :---: | :---: | :---: | :---: |
| API | API | API | API | API | API |
| Product 1 | Product 2 | Product 3 | Product 4 | Product 5 | Product N... |

**Platform of Common Services**
**(All products use this platform)**
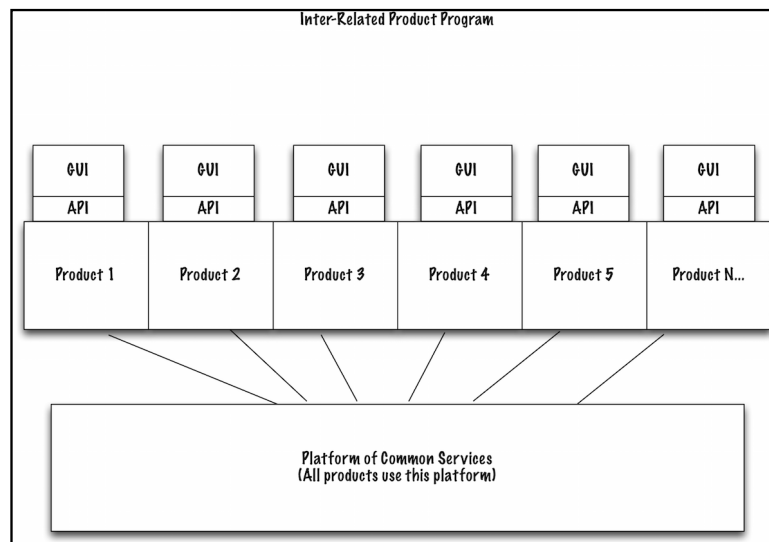
# Release Frequency, Costs & Decisions

**Potential for Release Frequency**

| Software as a Service | Boxed Software | Product with Firmware | Software with Hardware or Mechanical components |
|---|---|---|---|
| **Continuous** | | | **Infrequently** |
| Continuous Deployment: As often as several times a day | Often: But the cost of release is still high | Less Often: The cost of release is high | Infrequently: Every release might be a major release |

◄──────────────────────────────────────────►

Architectural decisions and releasing costs less

Architectural decisions and releasing costs more

Last responsible moment is later

Last responsible moment is earlier

---

# Program Managers Are Servant Leaders

- Your product might change how you program-manage

- Servant leadership works

  - Ask for the results you want

  - Ask for estimates, don't mandate

  - Ask for bad news

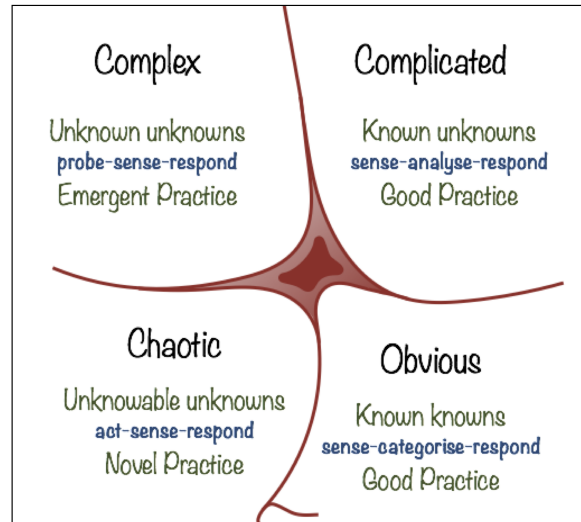  - Facilitate problem solving where the problems are

# Programs: Complex or Complicated

- Do you have known unknowns?

  - Ask experts for help

- Do you have unknown unknowns?

  - Try experiments

- Do you not have any idea at all?

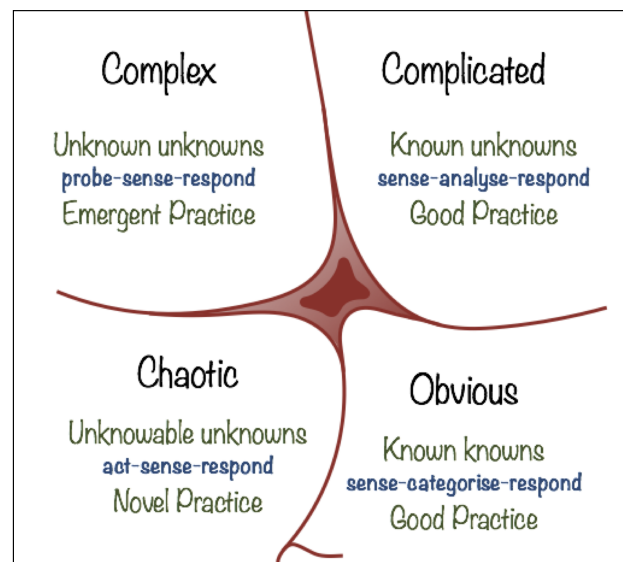  - Do something to move to Complex or Complicated

**Complex**

Unknown unknowns
probe-sense-respond
Emergent Practice

**Complicated**

Known unknowns
sense-analyse-respond
Good Practice

**Chaotic**

Unknowable unknowns
act-sense-respond
Novel Practice

**Obvious**

Known knowns
sense-categorise-respond
Good Practice

---

# Activity: Where is Your Program?

- Where is your program in the Cynefin framework?

- Do you need to obtain some data?

**Complex**

Unknown unknowns
probe-sense-respond
Emergent Practice

**Complicated**

Known unknowns
sense-analyse-respond
Good Practice

**Chaotic**

Unknowable unknowns
act-sense-respond
Novel Practice

**Obvious**

Known knowns
sense-categorise-respond
Good Practice
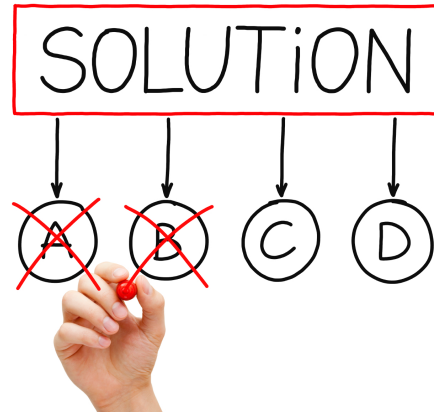
# Resiliency Over Defined

- Many of you have governance challenges from your waterfall days

- Agile and Lean programs demand resiliency—ability to flex and change—over defined deliverables

SOLUTiON

A B C D

---

"Scaling" Agile requires:
Autonomy
Collaboration
Exploration

# Agenda

- Program Managers and what programs look like

- **Agile roadmap**

- Program teams and kanban

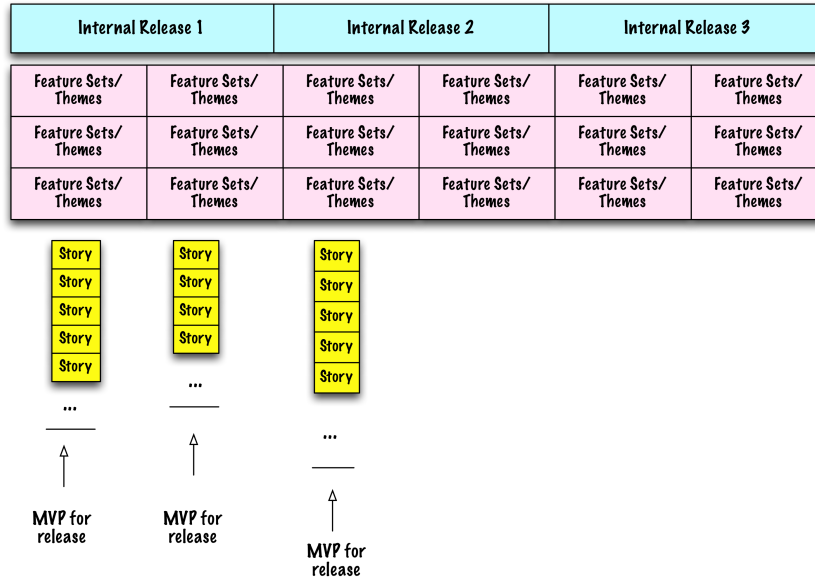- Small-world networks

- Estimation

- Culture of delivery

---

# Agile Roadmap

### Agile Roadmap for a Product: Several Quarters Out

| Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|----|----|----|----|----|----|

| External Release Tulip | External Release Daisy | External Release Rose | External Release Carnation |
|---|---|---|---|

| Int. Release 1 | Int. Release 2 | Int. Release 3 |
|---|---|---|

| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |
|---|---|---|---|---|---|---|---|---|---|
| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |
| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |

# One Quarter Perspective

**Agile Roadmap for a Product: This Quarter**

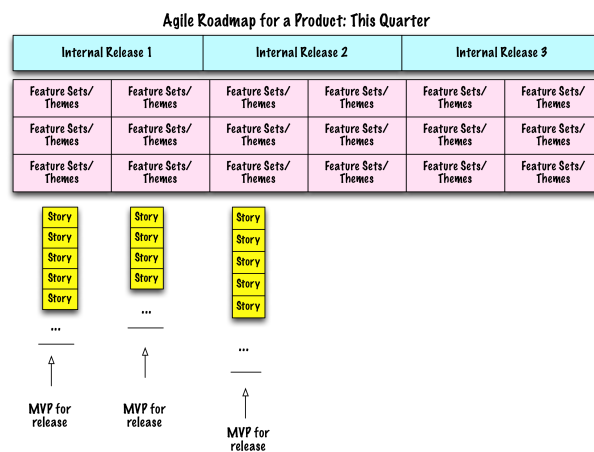| Internal Release 1 | | Internal Release 2 | | Internal Release 3 | |
|---|---|---|---|---|---|
| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |
| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |
| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |

Story
Story
Story
Story
Story
...
↑
MVP for release

Story
Story
Story
Story
...
↑
MVP for release

Story
Story
Story
Story
Story
...
↑
MVP for release

---

# Discussion: How Far Out do You Plan Deliverables?

- Do you have a 6Q roadmap?

  - Do you need a 6M roadmap instead?

- How many iterations/features/ MVPs do you plan for at one time?

- How often can you release:

  - Internally?

  - Externally?

**Agile Roadmap for a Product: This Quarter**

| Internal Release 1 | | Internal Release 2 | | Internal Release 3 | |
|---|---|---|---|---|---|
| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |
| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |
| Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes | Feature Sets/ Themes |

Story
Story
Story
Story
Story
...
↑
MVP for release

Story
Story
Story
...
↑
MVP for release

Story
Story
Story
Story
Story
...
↑
MVP for release

# Roadmaps & Backlogs Provide Team Autonomy

- Roadmap is the wish list

- Backlog is what teams will do

- Rank everything by value

  - Sometimes learning about risk is valuable

---

# Change is Key

- The more often you release, the more successful the product will be

  - The more often you can change the roadmap

- Fast feedback all over the program

- Inspect and adapt
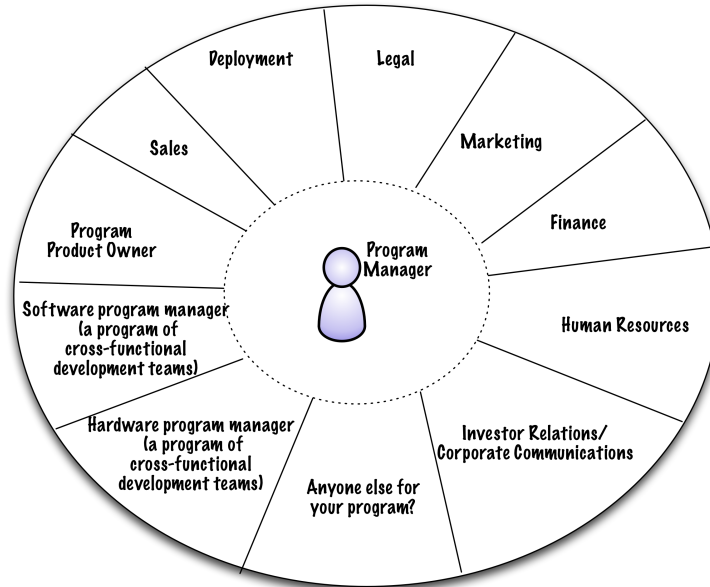
- Value-based approach to product development

# Agenda

- Program Managers and what programs look like

- Agile roadmap

- **Program teams and kanban**

- Small-world networks

- Estimation

- Culture of delivery

---

# Program Teams Solve Problems Across the Organization

- Core team shepherds the business value of the product

  - Facilitates problem-solving across the organization

- Software program team shepherds the business value of the software

  - Facilitates problem solving across the feature teams

# Core Program Team Manages Risks Across the Organization

---



Possible Kanban Board for Core Team

# Software Program Team Manages Obstacles for Feature Teams



Software Program Team

- Joe's Feature Team
- Tim's Feature Team
- Henry's Feature Team
- Deployment
- Any other project teams needed

Program Architect

Program Product Owner

Communities of practice:

Architecture, product owner, test, security, etc.

Software Program Manager

Sally's Project

| Feature set 1: Sam | Feature set 4: Sarah |
|---|---|
| Feature set 2: Stuart | Feature set 5: Susan |
| Feature set 3: Sunny | Feature set 6: Stan |

© 2015 Johanna Rothman

---



Possible Kanban Board for Software Program Team

| Ranked Backlog | In Progress | | Risk Management or Mitigation | Decision Needed Post-Action | Waiting: Stuck Items | Done |
|---|---|---|---|---|---|---|
| | Action item analysis | Action Item resolution | | | | |
| (notes) | Item and date started. Who is working the item. | | | | | (notes) |
| Transfers | (note) | | | | (note) | |
| Performance Monitoring | | | (note) | | (note) | |
| Admin | (note) | | | | | |
| Engine | | (note) | | | | |
| Hardware | | | (note) | | | |

26

© 2015 Johanna Rothman

# Agile Changes Program Management

- Teams (not managers) manage:

  - Commitment

  - How they build features

  - Evolve architecture

- Program management

  - Removes obstacles

  - Collects and explains program status



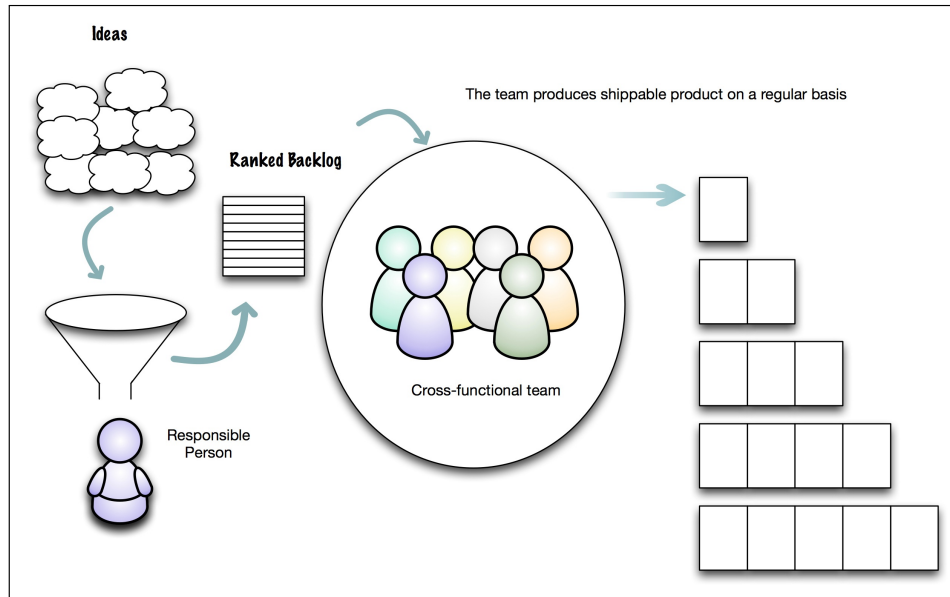Program of Concurrent Projects

Tim's project → Product you want to deliver to customers ← Other projects and deliverables to create a finished product...

Joe's project

Henry's project

Sally's project (a small program)
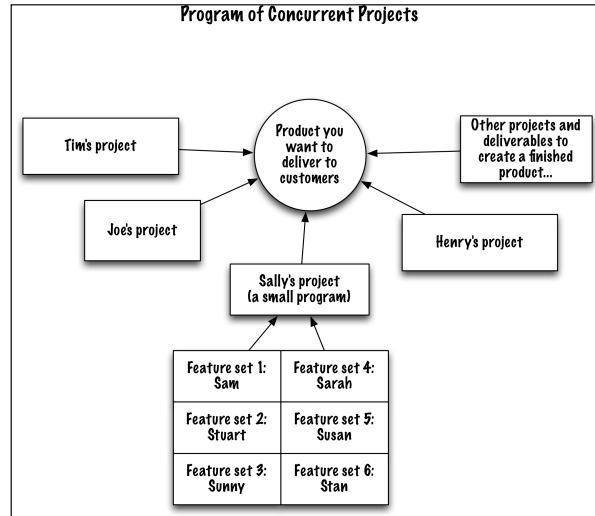
| Feature set 1: Sam | Feature set 4: Sarah |
| Feature set 2: Stuart | Feature set 5: Susan |
| Feature set 3: Sunny | Feature set 6: Stan |

# Agenda

- Program Managers and what programs look like

- Agile roadmap

- Program teams and kanban

- **Small-world networks**

- Estimation

- Culture of delivery

# Single Agile Team Collaborates Inside the Team



Ideas

Ranked Backlog

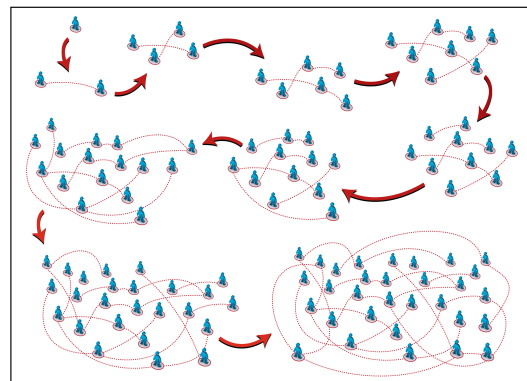The team produces shippable product on a regular basis

Responsible Person

Cross-functional team

---

# Scale from One Team to Many Teams?

- What do you need to scale collaboration from one team to multiple teams?

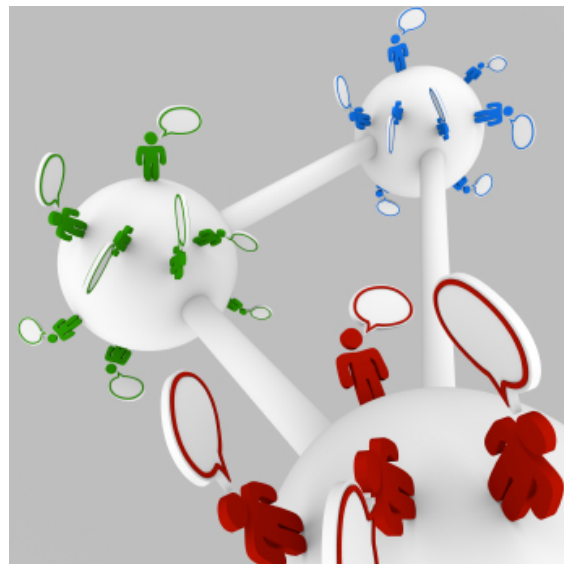- Hint: you do not need a framework

- Scale *out*, not up

# What's the Most Effective Way to Move Information In Your Organization?

---

# Rumor Mill
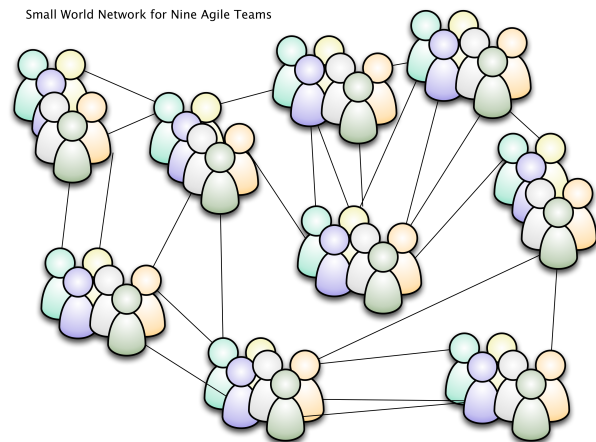
# Small World Networks

- Small world networks are more-and-less connected agile teams

- How connected are you to everyone else?

  - Some of you are highly connected

  - Some less so

- Programs take advantage of the network so they don't need hierarchies



Small World Network for Nine Agile Teams

---

# Organize the Teams

- Any form of agile or lean works for the project teams

  - Small batch size and continuous integration

- As long as the team delivers, you don't care how they organize themselves
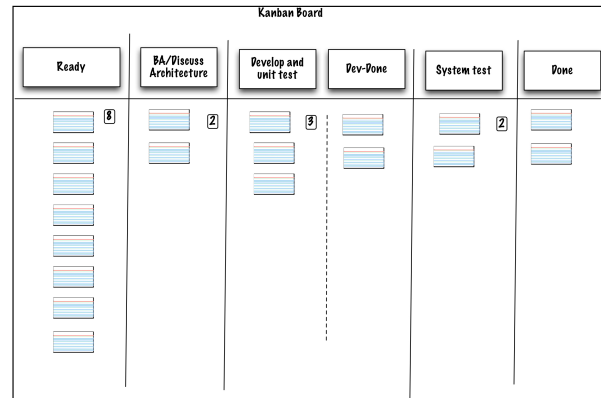
- Evolving architecture

- Plan to replan

# Lean Helps Create Transparency



- See the flow of work all over the organization

- Manage WIP

- Transparent approach to seeing all the work

---

# Use Small-World Networks

- Feature teams take responsibility

- Small-world networks provide autonomy and collaboration

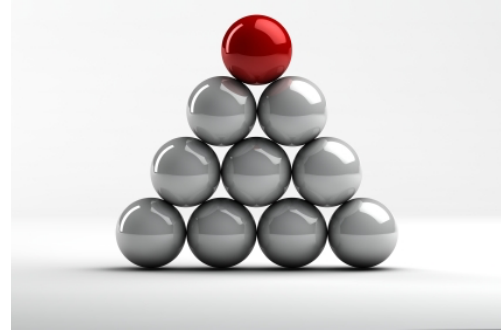- Communities of practice help with exploration

# Collaborate Across the Organization

- Transparent progress

- Transparent and pervasive communication

- Problem-solving autonomy

- Hierarchies slow everything down

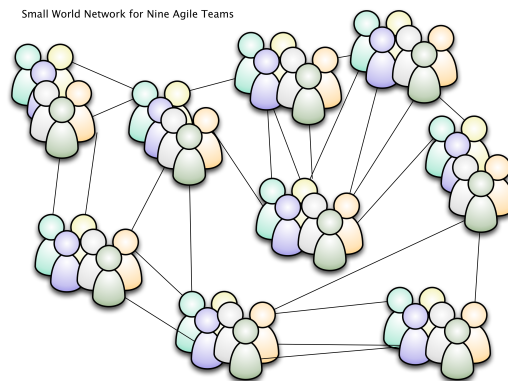- Information has to flow freely around the organization

---

# Discussion

- Where do you already have small-world networks in your organization?
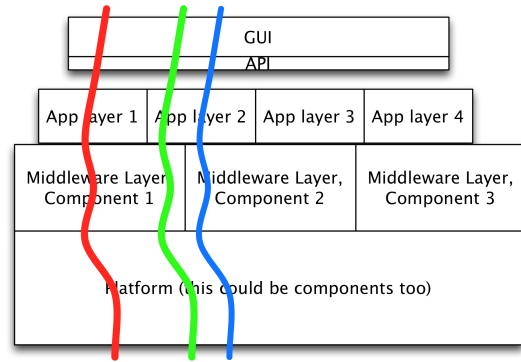
- How can you use them for better results?



Small World Network for Nine Agile Teams

# Teams Collaborate with Tests & Integration

- Create a program environment where people can collaborate

  - Tests and CI help teams collaborate

- Safe places to explore

- Autonomous teams

  - Deliver features often

| GUI | | | |
|-----|-----|-----|-----|
| API | | | |

| App layer 1 | App layer 2 | App layer 3 | App layer 4 |
|-----|-----|-----|-----|
| Middleware Layer, Component 1 | Middleware Layer, Component 2 | Middleware Layer, Component 3 | |

Platform (this could be components too)

---

# Agenda

- Program Managers and what programs look like

- Agile roadmap

- Program teams and kanban

- Small-world networks

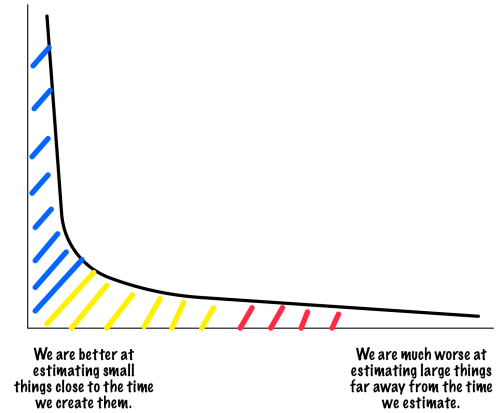- **Estimation**

- Culture of delivery

# Estimating a Program

Power Law Distribution: Example for Estimation



- You might need a gross estimate (high level estimate of "everything")

- The farther out or the more you need to estimate, the less you know

We are better at estimating small things close to the time we create them.

We are much worse at estimating large things far away from the time we estimate.

---

# What You Can Do for Estimation

- Work to a target date or cost

- Deliver small chunks, re-estimate at the small level

- Ask "How much do you want to invest before we stop?"

- Ask "How much value is this project or program worth to you?"

- Create a culture of delivery so you can change the roadmap and backlogs



PREDICTING THE UNPREDICTABLE

JOHANNA ROTHMAN
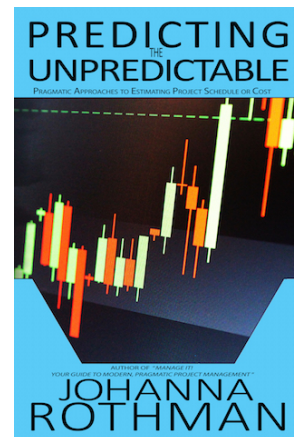
# Rank Everything by Value

- Updating the roadmap more frequently is often worth more than more estimation

  - Should we do this at all?

  - Business value points

  - Cost of delay

  - Waste

  - Who is waiting

  - Compare

  - Risk

- The more often you rank and update the roadmap, the more the teams work on what's most value

- My experience is that the teams do less overall

---

# Agenda

- Program Managers and what programs look like

- Agile roadmap

- Program teams and kanban

- Small-world networks

- Estimation

- **Culture of delivery**

# Delivery Solves Many Problems

- The more often teams deliver:

  - The more often you can update the program roadmap

  - The more often everyone gets feedback

  - The more often you can re-rank the next backlog

- A culture of delivering Running Tested Features solve many interdependency issues
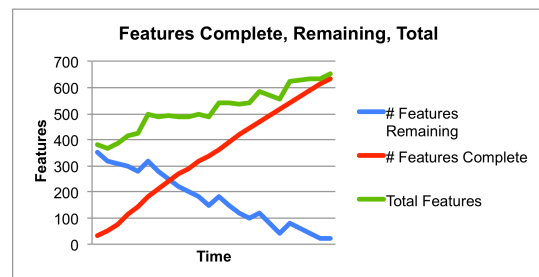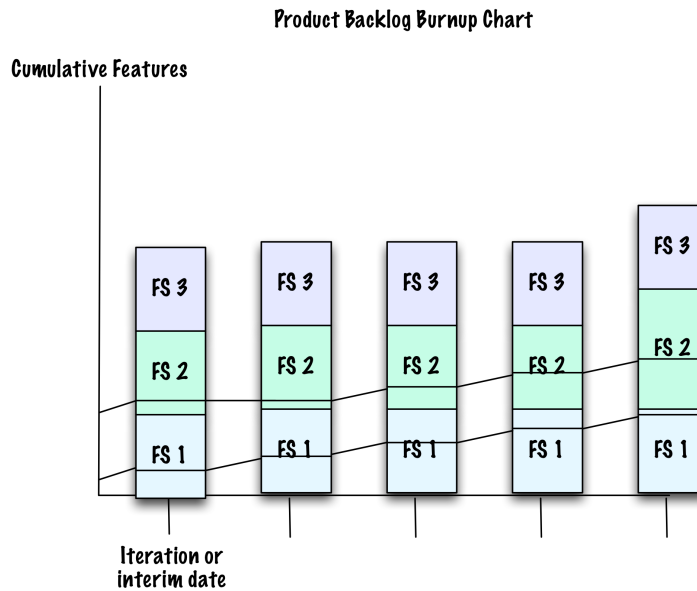
---

# Seeing Program Progress

- Program level measures

- Working product is the best measure

  - Do not even think about using team velocities and merging them

- Consider product backlog burnup



**Features Complete, Remaining, Total**

- # Features Remaining
- # Features Complete
- Total Features

# Product Backlog Burnup

**Product Backlog Burnup Chart**

Cumulative Features

| | FS 3 | | | | FS 3 |
| FS 3 | FS 3 | FS 3 | FS 3 | FS 2 |
| FS 2 | FS 2 | FS 2 | FS 2 | |
| FS 1 | FS 1 | FS 1 | FS 1 | FS 1 |

Iteration or
interim date

47 © 2015 Johanna Rothman

---

# Think Small to Go Large

- Trust teams to do their work (autonomy)

  - Small batch size

  - Technical practices essential

- Release often (to see progress and invite collaboration)

- Frequent releases encourage exploration

- It's agile and lean all the way across

48 © 2015 Johanna Rothman

# Stay in Touch?

- Many articles on jrothman.com and Pragmatic Manager email newsletter

- Look for *Agile and Lean Program Management: Scaling Collaboration Across the Organization* (in beta)

- Please link with me on LinkedIn