scruminc.

Agile Architecture

Fast, Cheap, Out of Control

Hosts: Jeff Sutherland
Presenter: Joe Justice

scruminc.

Scrum Inc. is the Agile leadership company of Dr. Jeff Sutherland, co-creator of Scrum. We are based in Cambridge, MA.

We maintain the Scrum framework by:

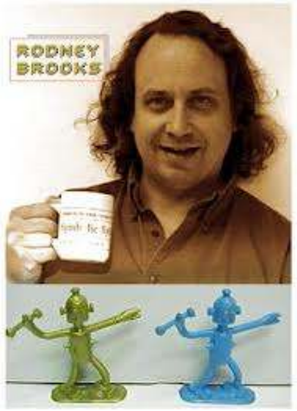
- Capturing and codifying evolving best practices,
- Conducting original research on organizational behavior
- Adapting the methodology to an ever-expanding set of industries, processes and business challenges

We also help companies achieve the full benefits of Scrum through our full suite of support services:

- Training (Scrum Master, Product Owner, Agile Leadership, online courses, etc.)
- Consulting (linking Scrum and business strategy, customizing Scrum)
- Coaching (hands-on support to Scrum teams)
- Publishing and new content development

We run our services company using Scrum as the primary management framework, making us a living laboratory on the cutting edge of “Enterprise Scrum”

Find out more at www.scruminc.com.



Rodney Brooks

CTO, Chairman, Founder, Rethink Robotics, 2008 – Present
iRobot Co-founder, CTO, Board member iRobot, 1990 - 2011
MIT Panasonic Professor of Robotics, 1984 - 2010

- **For 30 years we tried to build an intelligent system at the MIT AI Lab**
- **The best we could do was a smart chess program**
- **I decided on a totally different approach, no central processor, no database.**
- **It's called the subsumption architecture and it is**

FAST, CHEAP, and OUT OF CONTROL

Subsumption Architecture

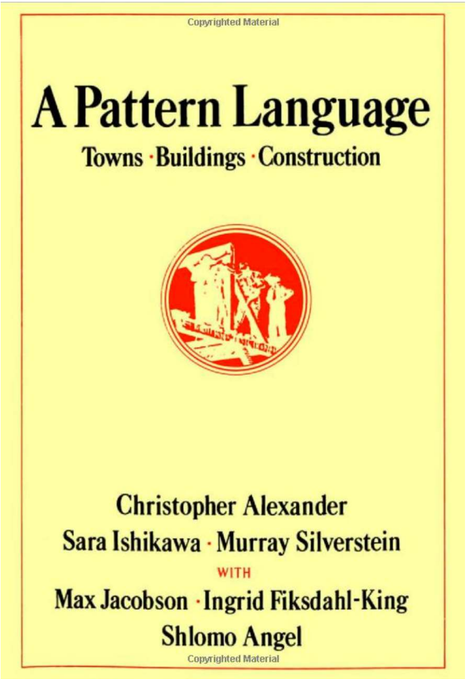
- No central processor
  - Every leg has a chip that can move the leg
  - Spine has chip that coordinates legs
  - Head has neural network chip that exhibits goal seeking behaviors along with gyroscope for balance
- All data is derived from sensors - the world is the database
- Emergent behavior is smarter than any individual component

So, Architecture is about Form more than Structure

- The way components are put together to make a system
- How they interact
- Architecture attempts to generate certain qualities
  - Usability, testability
  - Changeability, flexibility, maintainability
  - Extendability, refactorability, scalability
  - Portability, securability, deployability
  - Availability, reliability
- But the most important quality is the Quality Without a Name (QWAN)

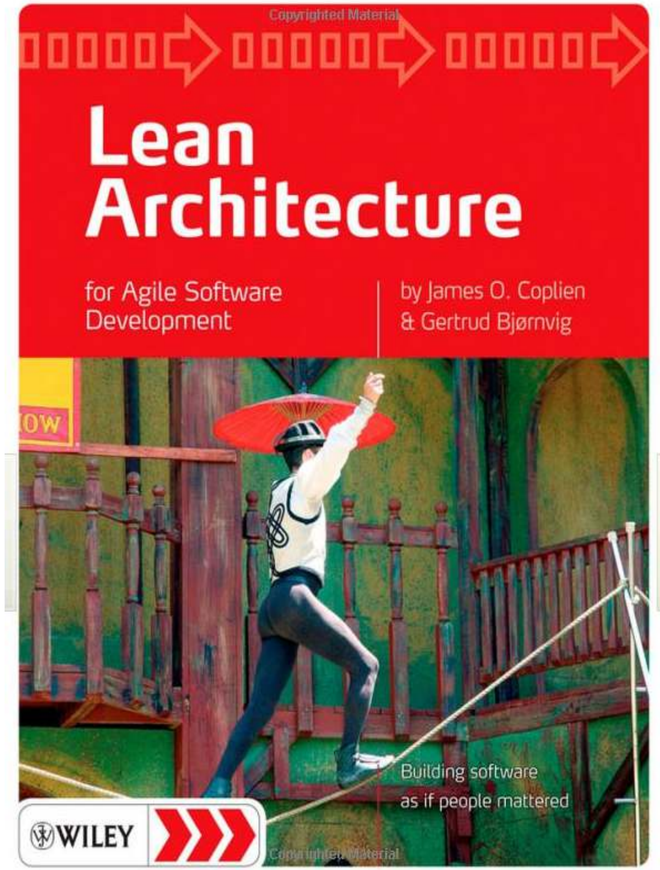
Alexander and the Patterns Movement

- The architecture of buildings created the software patterns movement
- Good architecture has QWAN
  - it feels like home
  - it is intuitively easy to navigate
  - it is comfortable and cozy for the user
  - it grows through accretion and becomes more beautiful with age
  - it is a feeling the user gets that they want to repeat again and again



What is Agile Architecture?

- Allows individuals and interactions on small teams to proceed autonomously with minimal impediments from the rest of the organization
- Works at the end of every sprint, even though it may change during every sprint
- Gets the customer involved so that the form produces user delight
- Embraces change, is constantly changing, and continually emerges
- **Agile architecture is Lean architecture**



scruminc.

Agile Requires Component Architectures

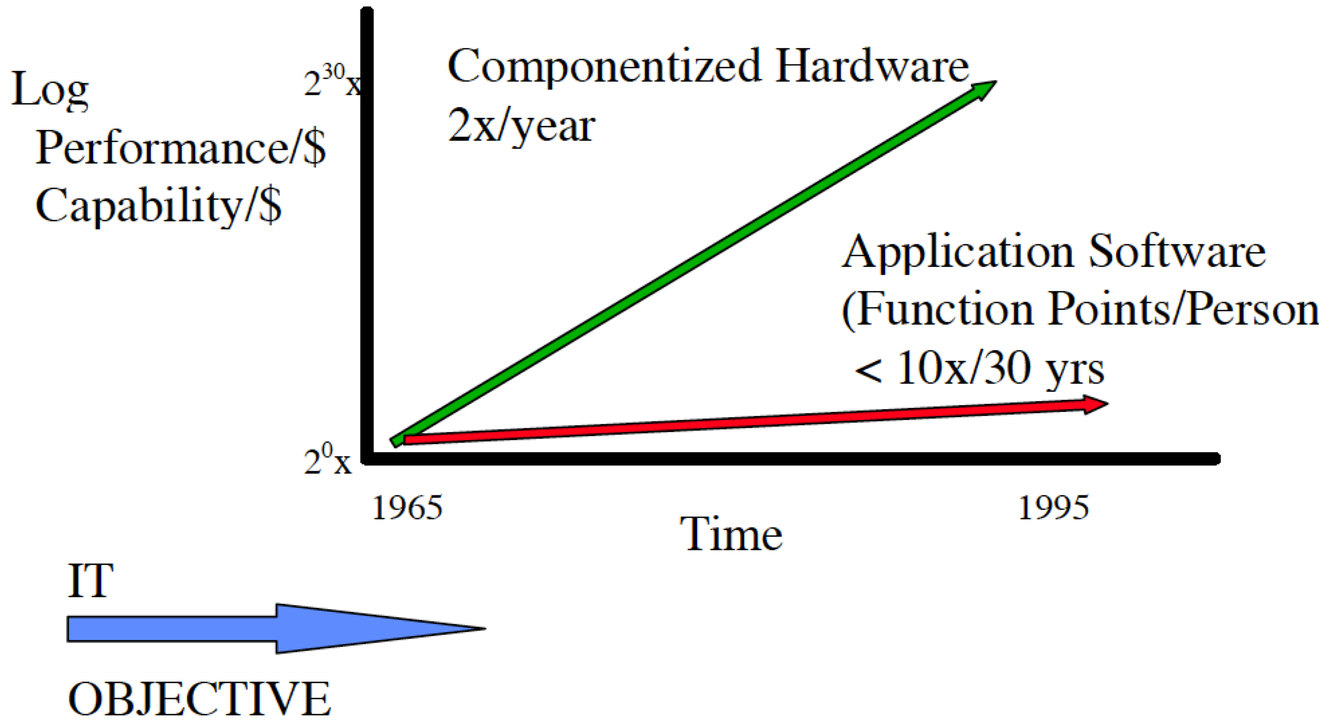


Figure 1: Hardware Price/Performance vs. Software Price Performance[7]

Good Architecture
- enables Moore's Law

Bad Architecture
- reinvents the wheel

J. Sutherland, "Why I Love the OMG: The Emergence of a Business Object Component Architecture," ACM StandardView, vol. 6, pp. 4-13, March 1998.

Roots of OMG Model Driven Architecture

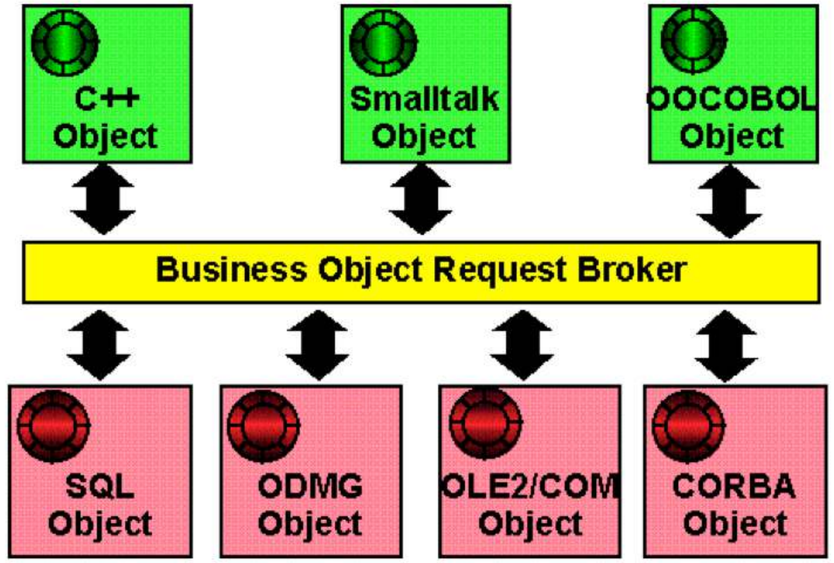


Figure 2. ***ANSI X3H7 Standardization Targets*****. 24 Sep 1994**[12]

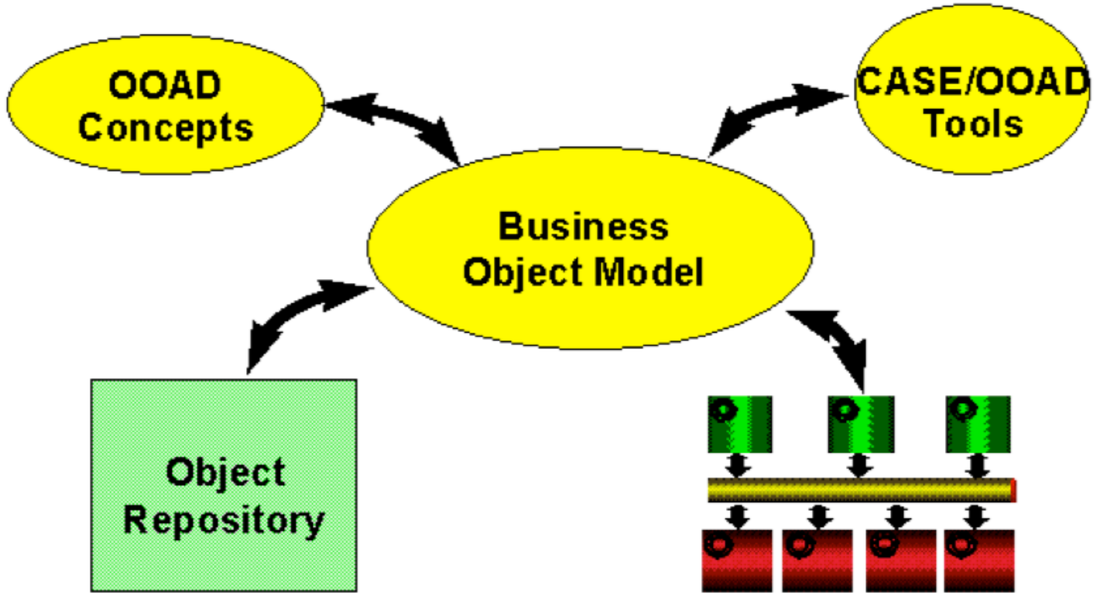


Figure 3. *ANSI X3H7 Standardization Targets*. 24 Sep 1994.

J. Sutherland, "Why I Love the OMG: The Emergence of a Business Object Component Architecture," ACM StandardView, vol. 6, pp. 4-13, March 1988.

Architecture in the First Scrum Team

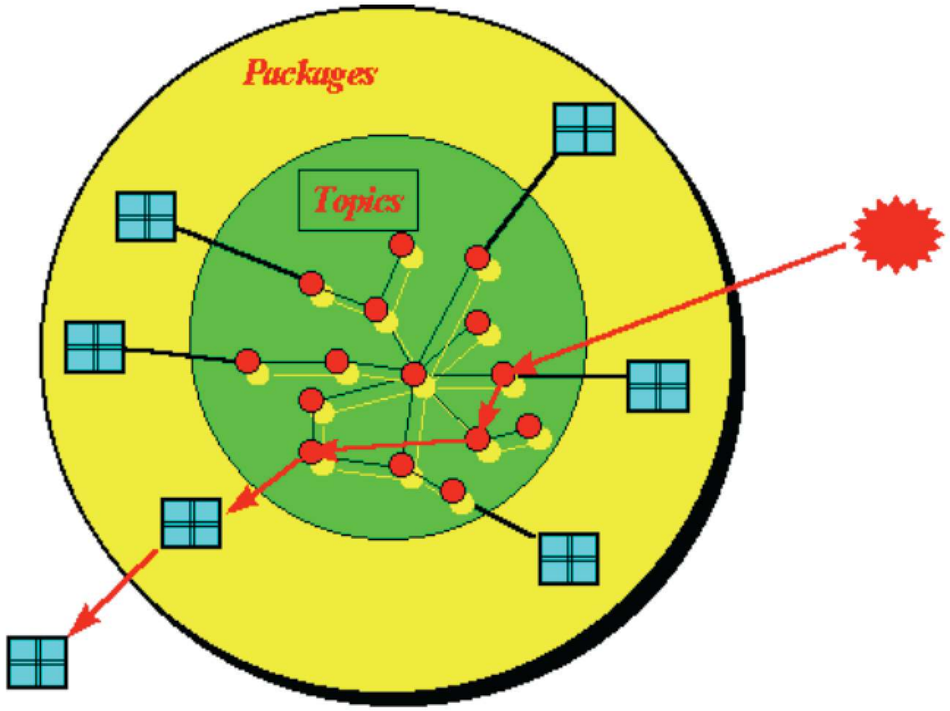


Figure 3: Firing a SyncStep causes a ripple through the system that triggers emission of a package of functionality visible to the user.

Sutherland, J., The Scrum Papers: Nuts, Bolts, and Origins of an Agile Framework. Version 1.1, Scrum Inc. 2012

scruminc.

Component Architectures

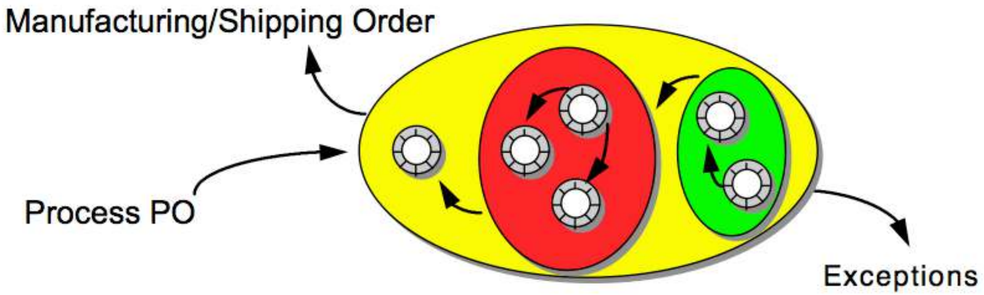


Figure 4: An Order Entry Business Object

- presentation
- business model
- data access

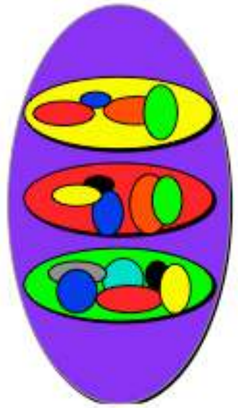


Figure 5: Client-Server Component

J. Sutherland, "Why I Love the OMG: The Emergence of a Business Object Component Architecture," ACM StandardView, vol. 6, pp. 4-13, March 1988.

Deployability of Architectures

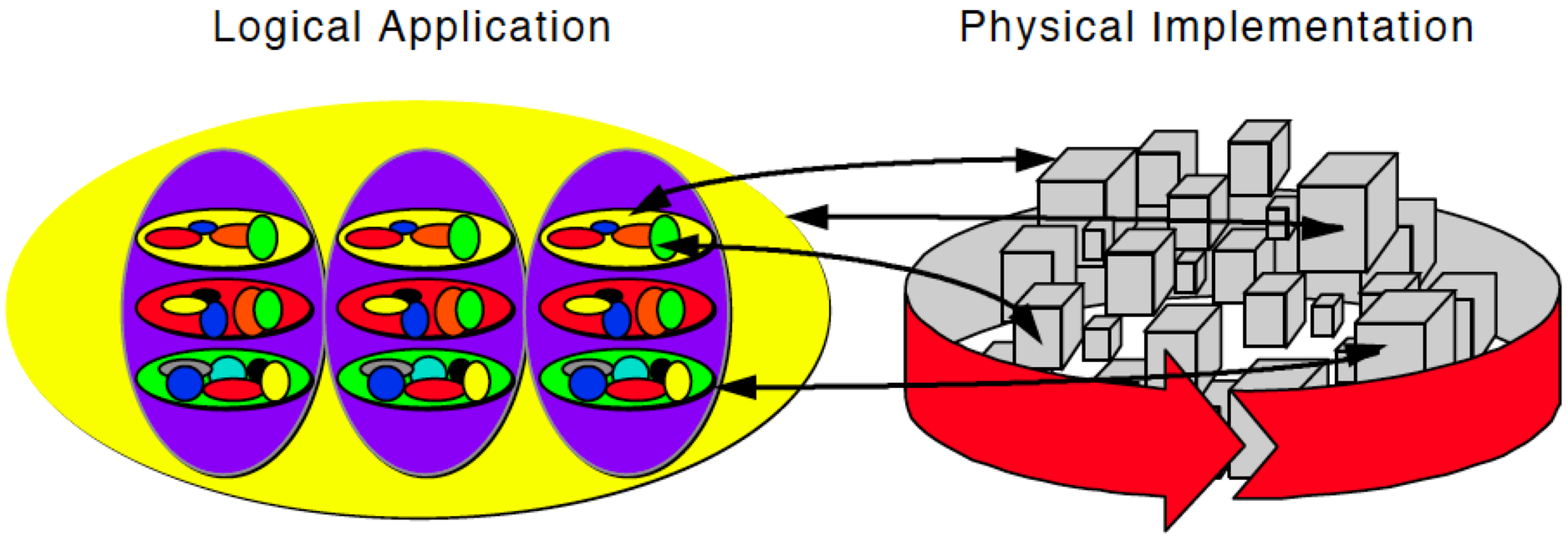
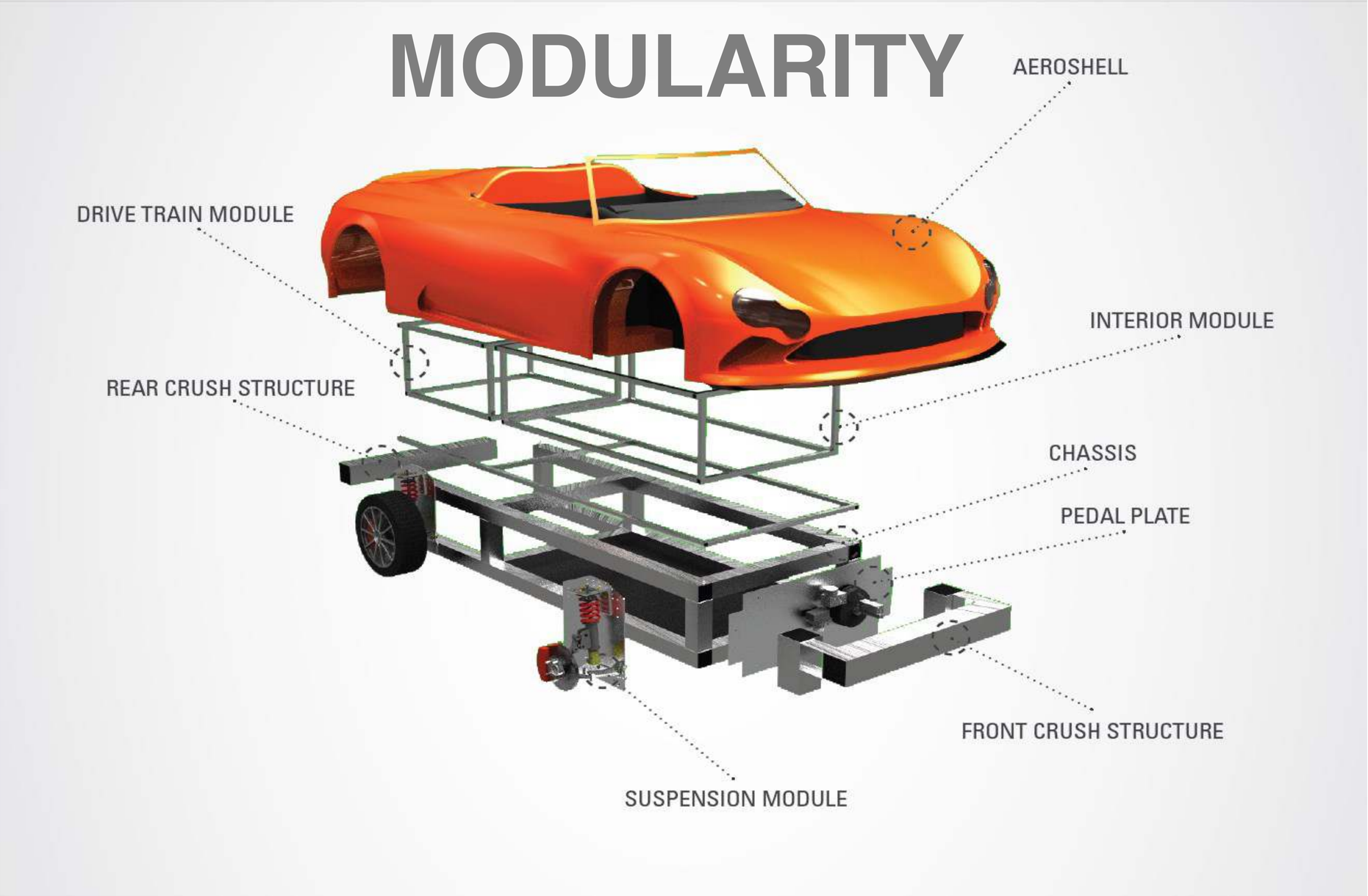


Figure 6: Application Business Object with Nested Client/Server Components

J. Sutherland, "Why I Love the OMG: The Emergence of a Business Object Component Architecture," ACM StandardView, vol. 6, pp. 4-13, March 1988.

So Joe, What is the Wikispeed Architecture?



WIKISPEED SGT01 prototype road vehicle, exploded view

scruminc.

8 Modules Allow Rapid Iteration

WIKISPEED modules cut-away view

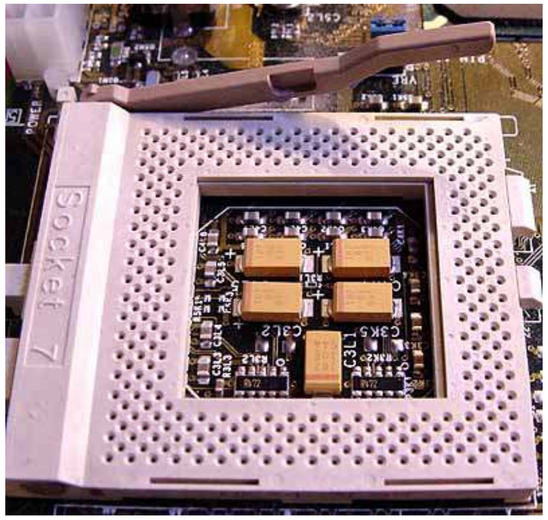
scruminc.

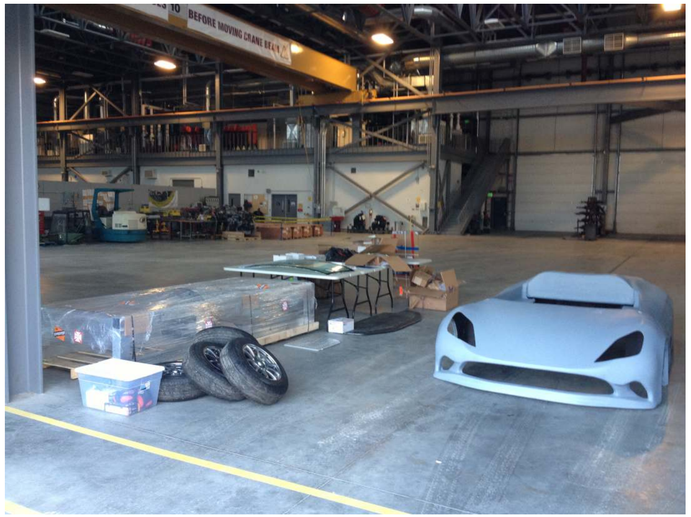
OOA Allows Distributed, Collaborative Teams

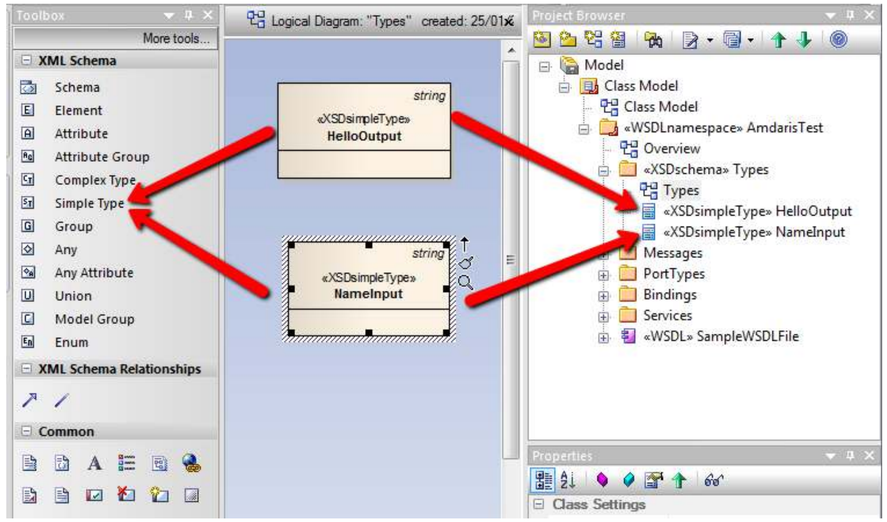
Scrum Inc. Build Party demo, Lockheed Martin Space Systems campus, Sunnyvale, CA, USA

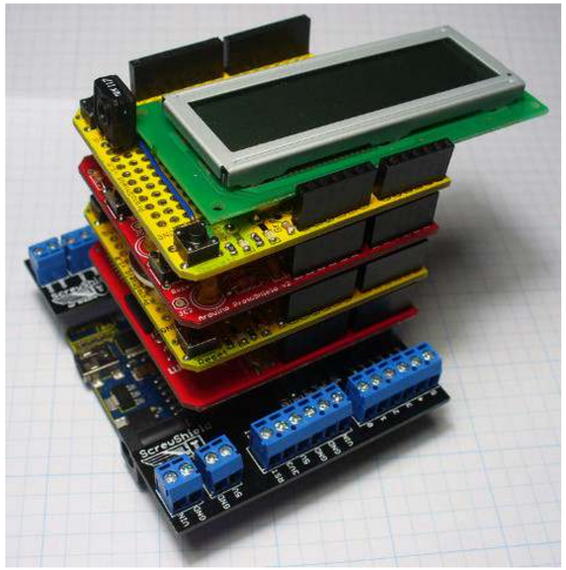
scruminc.

Modularity Requires a Known, Stable Interface









- 8 modules developed in parallel.
- INDEPENDENCE between modules.
- Record time to market

- Independently evolving clients and Services
- Ecosystems of Services
- MSOA
- Extend without rewrite

scruminc.

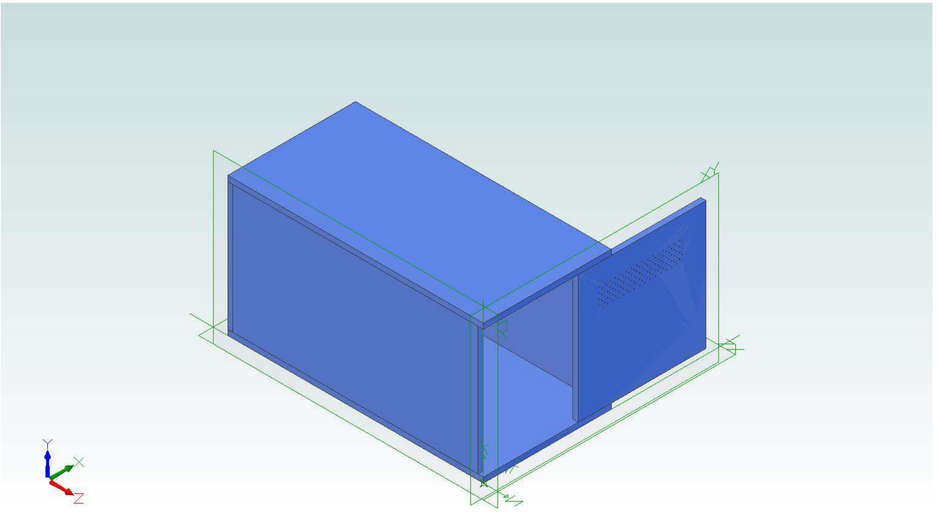
Costs of Modularity

- In WIKISPEED car: .5” wider, 12lbs heavier, across 78” wide 1404lbs vehicle.
- The extra 12lbs was offset the following sprint. (source, team WIKISPEED).

- In hand-held radio system: 3mm wider.
- The extra width was offset the following iteration. (source, David Slaten, Agile EE).

- In Construction: Overall more compact vs traditional construction (source, team WIKISPEED MicroHouse).

WIKISPEED suspension modules V25


WIKISPEED modular MicroHouse prototype

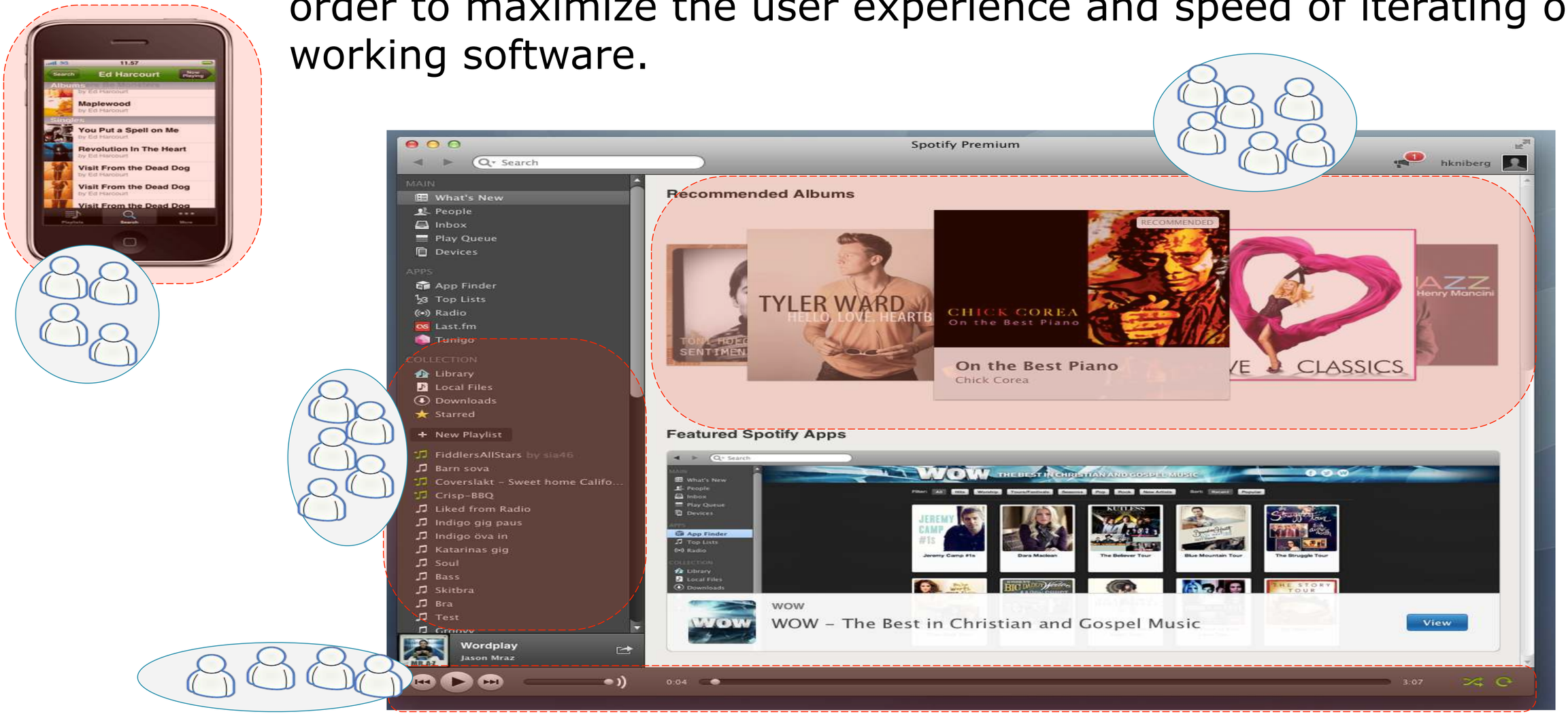
What is the Opposite of Modularity?

- Integration allows compaction, local optimization
- Integration inside modules, like re-factoring inside a wrapper with a known interface, creates elegance.
- Integration by collapsing modules together increases the cost to make change
  - like changing memory in your desktop versus most smartphones.

scruminc.

Squads Own Customer Visible Piece of Product

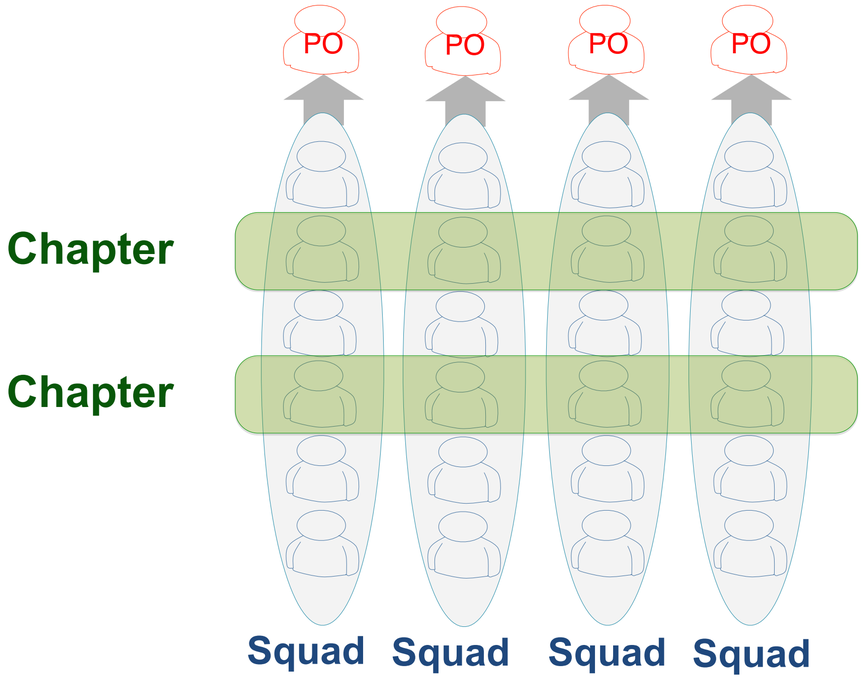
The first step to scaling is to form teams focused on features in order to maximize the user experience and speed of iterating on working software.

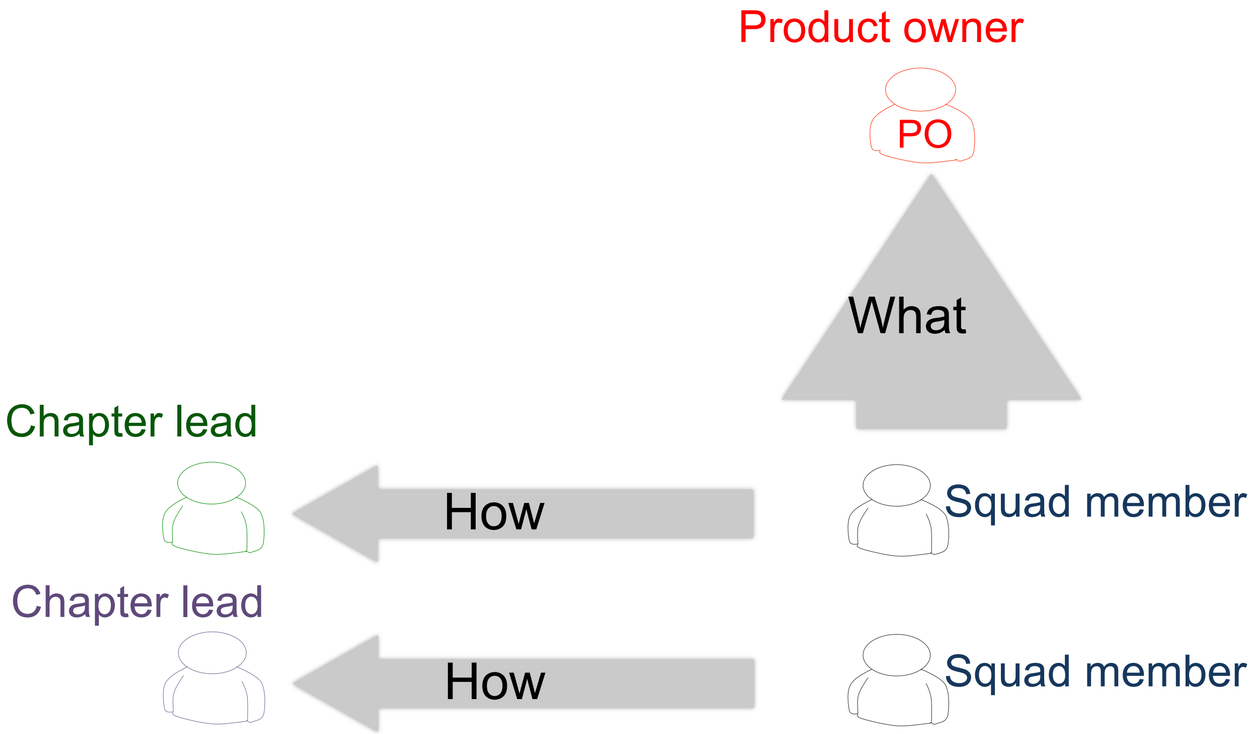


Scaling @ Spotify, Anders Ivarsson & Henrik Kniberg, Scrum Alliance Gathering Paris, 6 Feb 2013.

scruminc.

Chapters Provide Virtual Teams With Critical Expertise

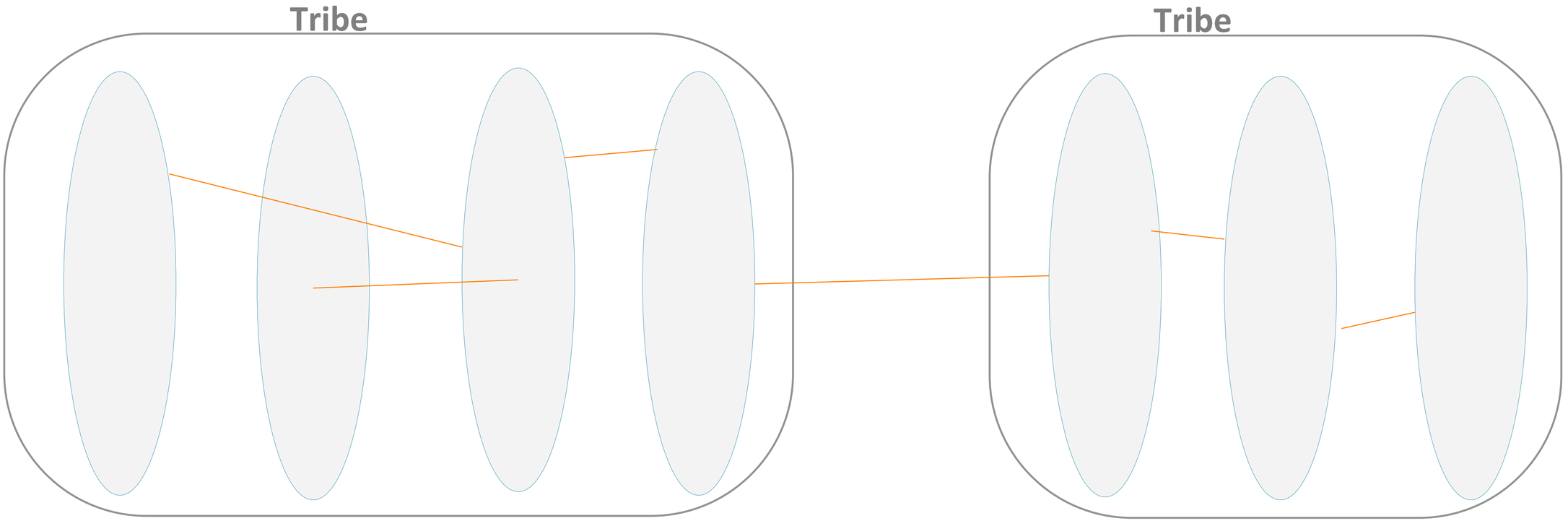




Scaling @ Spotify, Anders Ivarsson & Henrik Kniberg, Scrum Alliance Gathering Paris, 6 Feb 2013.

scruminc.

Dependency Analysis

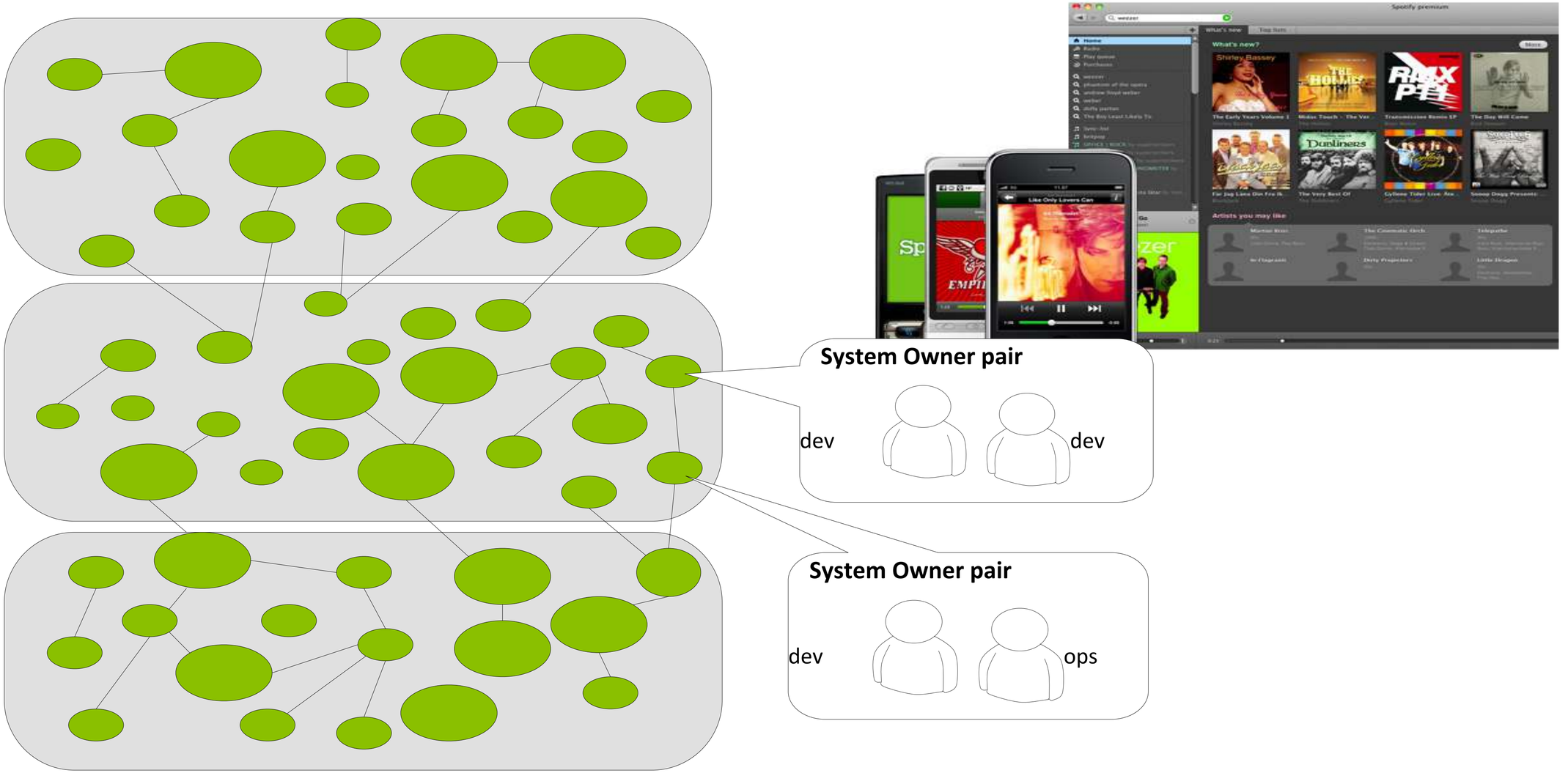


	A	B	C	D	E
1	Squad	Depends on	Dependency	Comment	Same tribe?
2	Music Player				
3	Content	Ops	Slowing	Need machines, connections, help set-up things etc. Works really well in general, but at times the workload on operations causes the lead times to grow and slow us down	No
4	Content	NeXT	No problem	Storage. Not big, mostly information/communication needs to happen.	No
5	Content	BFS	No problem	Replacement service	Yes
6	Content	Team 2	No problem	Communication around next story	No
7	Content	Team 1	Future	Content ingestion	No
8	BFS	UX	Slowing	Need UX to discuss, review and provide mock-ups	No
9	BFS	Content	No problem	Normal dependencies, sprint work.	Yes
10	BFS	Mobile	Slowing	No internal mobile developers within Squad.	No
11	BFS	Analytics	Slowing	A/B test results slowing down roll outs of features	No
12	BFS	Team 3	Blocking	Waiting for data dumps	No
13	BFS	Team 1	Future	Waiting for data dumps	No
14					

Scaling @ Spotify, Anders Ivarsson & Henrik Kniberg, Scrum Alliance Gathering Paris, 6 Feb 2013.

scruminc.

System Owners



Scaling @ Spotify, Anders Ivarsson & Henrik Kniberg, Scrum Alliance Gathering Paris, 6 Feb 2013.

scruminc.

Architectural and Technical Debt

- Any architectural decisions that prevent **solving a defect within the same sprint it is found** is worth revisiting.
- Architecture can support rapid defect locating, understanding, and resolving.
- Architecture can support rapid deployment.
- Tight Coupling may be a symptom of a missing interface.
- Additional hooks beyond those defined in the interface may be a symptom the architecture is not understood by the team, or the interface is outgrown and a candidate for iteration.

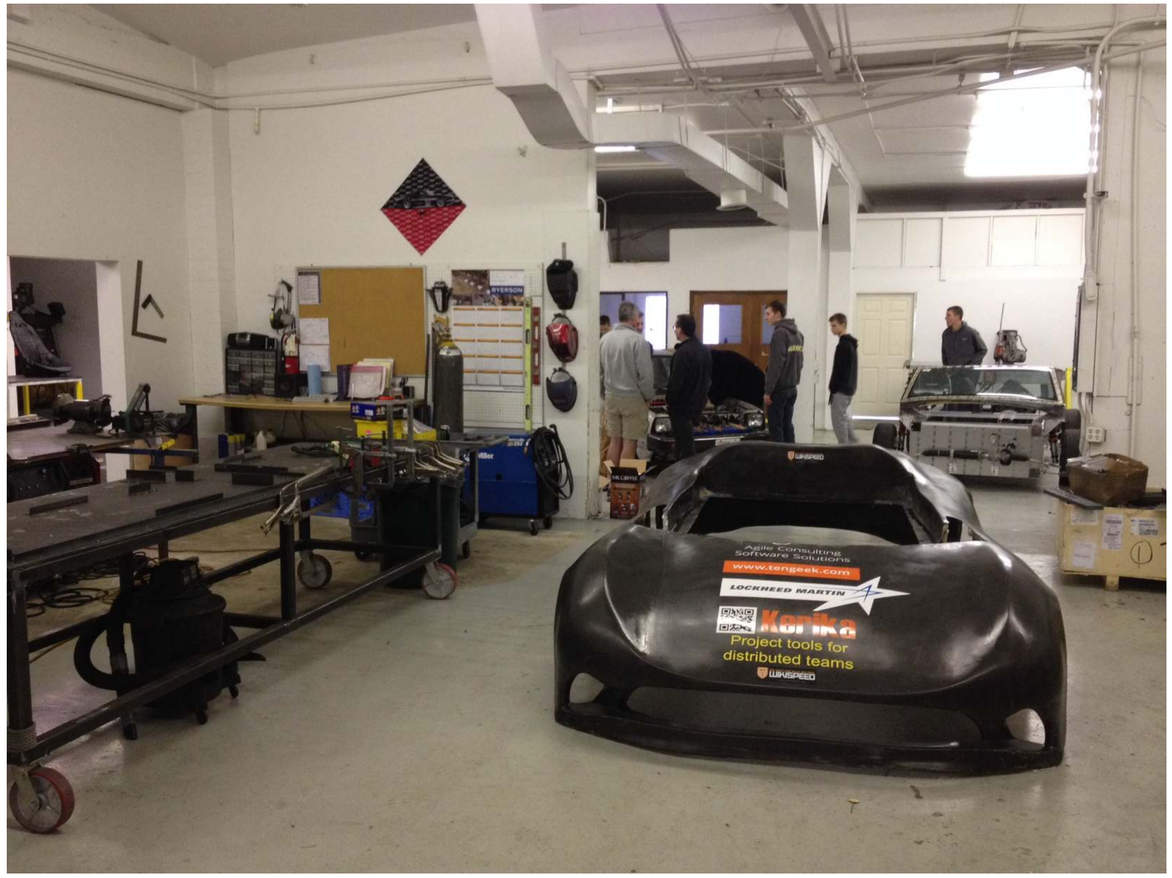
IDX: The First Scaled Architecture

- IDX (now GE Healthcare) had 8 business units and each had 3-12 products supported by over 600 developers.
- Used similar scaling mechanism to Spotify
  - Each team had an architecture representative on a Scrum of Scrum architecture team led by the Business Unit Lead Architect
  - The enterprise architecture team had Business Unit Lead Architects led by the CTO who had senior management commitment to 10% of all points in every sprint dedicated to architectural improvement (technical debt remediation, integration, branding, etc.)
  - Within 6 months all products converged to a single look and feel with common branding and workflow integration across all products. Third party products could appear in the integration framework and look like part of the global product.
  - 10 years later this architectural framework was adopted by GE Healthcare as the standard for all applications.

Scaling at Wikispeed

- Scale out versus scale up
  - Scale out is cheap, modular, redundant
  - Scale up has high initial up cost, single point of failure
  - Economy of scale available in either case

- MicroFactories
- WikiShops
- BuildParties with mobile DropKit



WIKISPEED Joint venture factory, Renton, WA, USA

How Much Architecture Up-front?

- Contract First Design
- Less than 10 modules
  - Allows “everyone on the team” to be fluent in the architecture
- Interfaces are stable
  - Each module has known mounting and I/O
  - Changing interfaces has rippling effect cost
  - Interfaces are versioned
- Treat the car like a pack of 8 web services
  - Contract is stable, what’s inside changes cheaply

AntiPattern: Architects Don’t Code

"The SystemArchitect responsible for designing your system hasn't written a line of code in two years. But they've produced quite a lot of ISO9001-compliant documentation and are quite proud of it.”

Name: Architects don't code
Problem: A large, complicated system needs to be easy and quick to build, debug and maintain.
Context: The development organization has a list of supposed junior programmers and a list of OO or relational "experts", such as senior programmers or solution architects.
Forces: Supposedly, having an expert architect will bring about consistency, cleanliness, modularity, and other characteristics of efficient software development. Expert time is expensive or rare or both.
Unfortunately, design and coding are two sides of the same coin. You cannot design if you cannot code and you cannot code if you cannot design. TheSourceCodeIsTheDesign.
Supposed Solution: Some expert, usually an authority, will architect the system on paper. The expert(s) won't be unduly bothered with technical details because their time is valuable. The team will be required to follow the architecture so that the good qualities of the system are assured.
Resulting context:
A project where the code reflects a design that the SystemArchitect never thought of because the one he came up with was fundamentally flawed and the developers couldn't explain why it was flawed since the SystemArchitect never codes and is uninterested in "implementation details".
Implementation lags and fingers are pointed; whoever has the most authority wins, usually the expert. The programmers may need to be replaced if they are deemed not to be "team players" because of arguing with the architect.
Real solution: Get architects involved at an implementation level. They will scream and moan but it's for their own good. They don't need to be writing (necessarily) an equal share of production code (if, for example, your organization is very adamant about the RoleOfASystemArchitect) but they need to get their feet wet from time to time and need to be aware of how changes in their design affect the project.
Alternately, eliminate entry-level programmers (the rank, not the employees) in favor of having the system only coded by those with enough knowledge (the MythicalManMonth solution).

http://c2.com/cgi/wiki?ArchitectsDontCode

scruminc.

Conclusion

- Create or evolve an object-oriented component architecture
- Generate autonomous teams that can upgrade their components every sprint while keeping the full system deployable
- Manage dependencies across teams
- Every team has a person responsible for architecture that is a member of a virtual architecture team
- Every team member is trained and understands the architecture of the system
- At appropriate levels in the architecture there are system owners that ensure conceptual integrity
- Every architect is on a team coding every sprint
- The architecture is emergent, constantly changing while maintaining deployability at all times
- Always remember Bell Labs – 100 million lines of code deployed to 42 countries with weekly updates and zero down time – if they can do it you can do it

KEEP UP THE AWESOME!!

RAPIDLY SOLVE PROBLEMS FOR SOCIAL GOOD

@ScrumInc
@WIKISPEED
#Agile
#Scrum

Questions?

Wallabies

scruminc.

Stay Connected!

Our Website

- check in for announcements, new content and services, book releases, and more!
- www.scruminc.com

ScrumLab

- articles, videos, papers on all things scrum
- scrumlab.scruminc.com

Blog

- scrum.jeffsutherland.com

Online Courses

- advance your learning with our interactive online courses. visit the scrumlab store to view upcoming topics.

Twitter, Facebook, and G+

- @jeffsutherland, scrum and scrum inc.

scruminc.