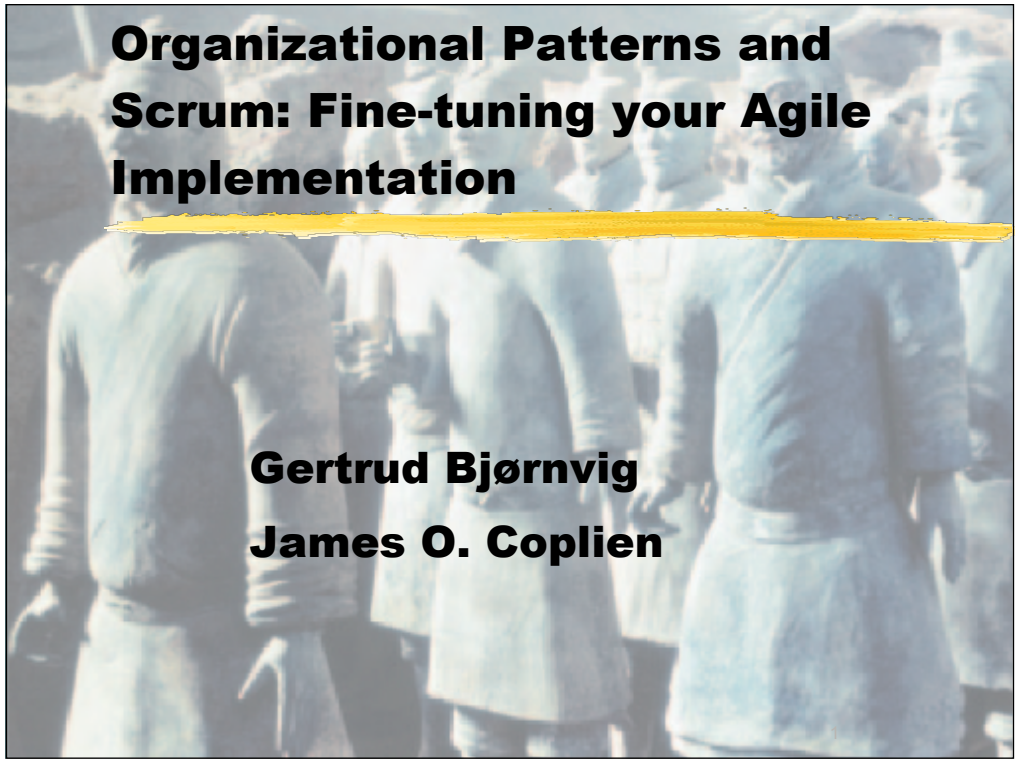
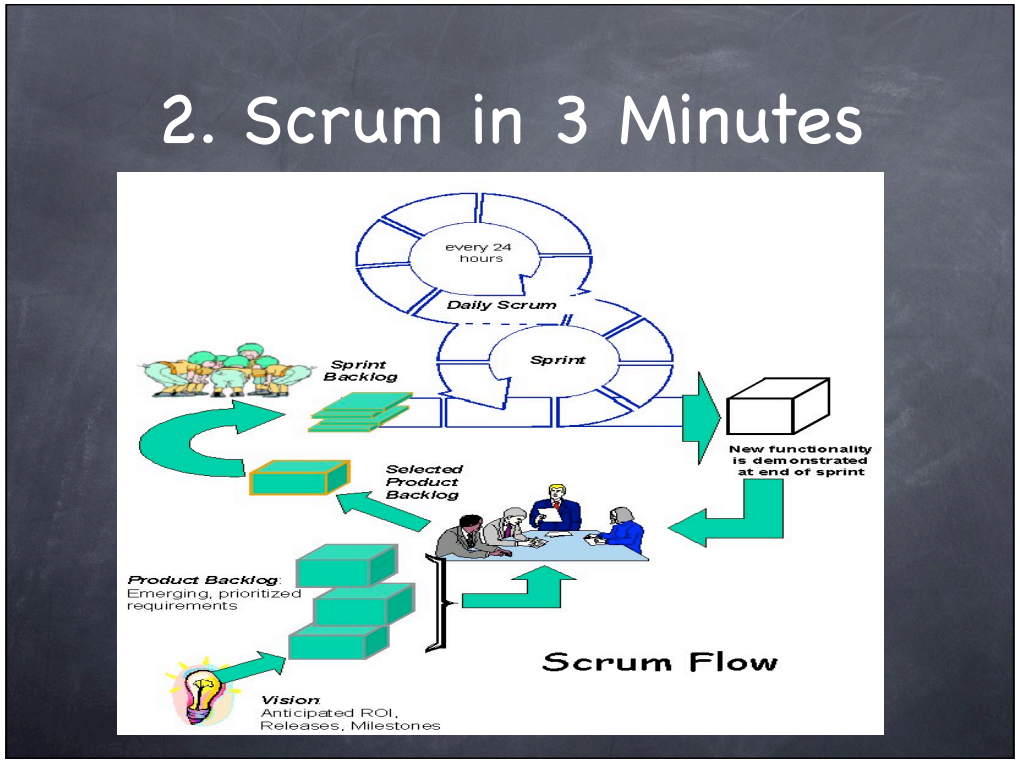


Platform time: 1 hour. 12 slides at 5 minutes per slide.

History

- 1915: Kroeber
  - Universal Patterns: Transcend Cultures
  - Systemic Patterns: common roots in an ancient culture
  - Total Culture Patterns: give a culture its identity
- 1964: Notes on the Synthesis of Form by Alexander
- 1977: A Pattern Language by Alexander
- 1993: Borland study
- 1993: Hillside: Buffalo Mountain
- 1994: First organizational patterns at PLoP
- 1997: Alistair's book
- 1998: Beedle et al. PLoP paper
- 2004: Organizational Patterns book
- 2008: scrumorgpatterns.com



So much for the roles: what is Scrum in its totality? Scrum is a process, and this diagram exemplifies the Scrum Flow.

The flow starts at the bottom left with a product vision, conceived by the Product Owner. The vision encompasses products or a product line that will generate ROI according to envisioned releases and milestones. The Product Owner articulates this vision as Product Backlog Items (PBIs): work items, products, features, and issues to be operated on by the Team.

The Product Backlog is a stack that is ordered to optimize ROI. The ordering takes place at a Release Planning Meeting in the center of the picture. All roles are involved in this meeting, where cost, business value, and all other relevant factors come to the table.

The team takes PBIs from the top of the Product Backlog until they have one Sprint's worth of work. A Sprint is a major cycle that recurs once every two weeks to once a month, more or less, and it is driven from the Sprint Backlog. The circle at the top center represents a single Sprint cycle. The Sprint Backlog contains Tasks that are broken down from the PBIs; a Task is two hours to two days. During a Sprint all the project resources are available to the Team, but the Team is not available to the rest of the project. The Team may change mapping of tasks to team members at any time.

Every day, the Team holds a meeting called a Daily Scrum. This establishes a 24-hour rhythm for the team, which is the circle at the very top.

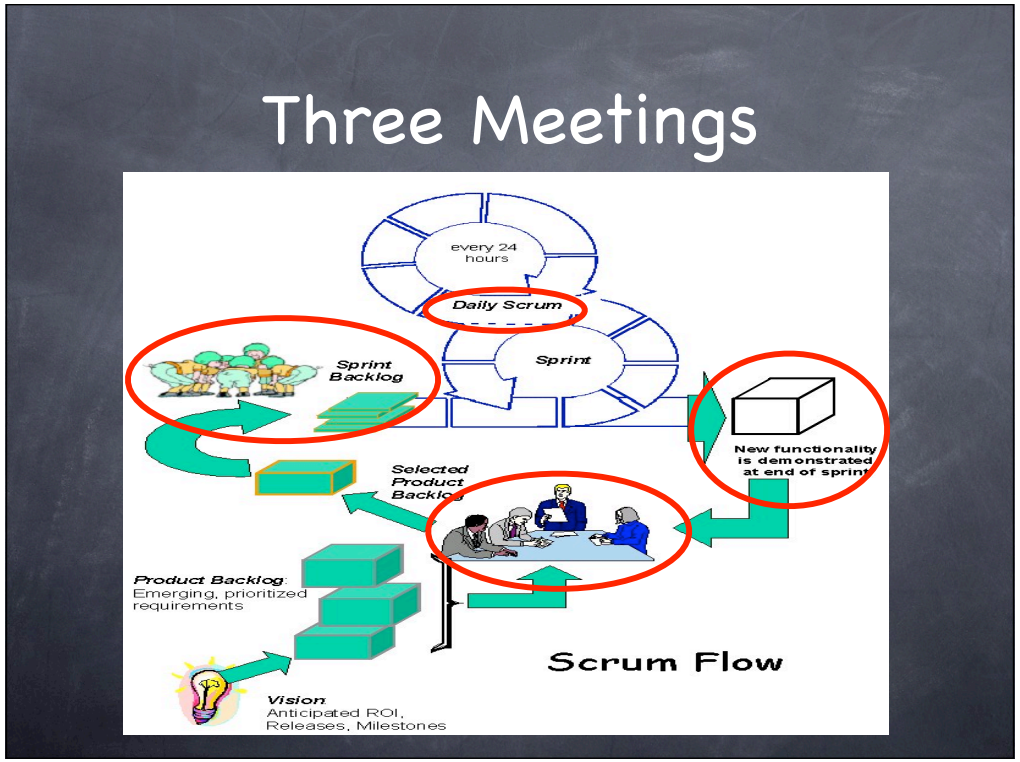
The Sprint delivers product, which is demonstrated to the Product Owner (and customer) to get feedback. New and modified work requests come from this meeting, and a new Product Planning meeting is held to complete the cycle.

There are three lists in Scrum: the Product Backlog, the Sprint Backlog, and the Impediment List.

The Product Backlog lists the features, products, and issues (Product Backlog Items, or PBIs) that need to be worked by the team. The near-term items are at the top and the more deferred items are at the bottom. Items become more elaborated and refined as they move within a two-Sprint window at the top of the backlog. Each PBI is ascribed with an effort estimate (only an estimate—not a commitment) and a business value estimate. The Product Owner is responsible for maintaining the Product Backlog, but any stakeholder may add an item to it at any time.

The Sprint Backlog contains Tasks that are broken down by the team from PBIs. The Sprint Backlog is owned completely by the Team: only the Team may add or delete items from the Sprint Backlog. The Team commits to delivering the entire Sprint Backlog at the end of a Sprint; a given Sprint Backlog is extant only for one Sprint.

The Impediment List is the ScrumMaster's own private Product Backlog. It is a list of obstacles and impediments for which the ScrumMaster is responsible. Removing these impediments helps the team achieve its goals, improve its velocity, and enjoy quality of work life. The list should be ordered, but Scrum does not define how the ordering takes place.



Scrum is four meetings.

First is the Product Planning Meeting, where all the stakeholders come together with their input for the Product Backlog. This usually happens between Sprints, or at the end of Sprint iteration n - 1 in preparation for sprint n.

The Sprint Planning Meeting requires the Team, ScrumMaster and Product Owner. At this meeting the Team selects PBIs from the top of the product backlog, counting effort numbers as they go, until one Sprint's worth of effort is exhausted. The Team then breaks down those PBIs into Tasks of two to 16 hours and puts them on the Sprint Backlog. The Sprint Backlog may be ordered but since all items must be done at the end of the sprint, the ordering is only an optimization.

During the Sprint, there is a daily meeting called a Daily Scrum. This meeting requires the entire Team and the ScrumMaster. Product Owners may attend, but may not speak (they are Chickens).

At the end of the Sprint is a Sprint Review meeting, with the Product Owner, Team, ScrumMaster, and potentially the Customer. This is where the team receives feedback on the match between desired functionality and what the team delivered. The Product Owner is the arbiter of whether a feature meets the requirements or not, but of course the Product Owner is likely to consult customers in this matter.

Pattern: A Definition

1. It's something that you build (and how to build it)
   - Changes communication paths
   - Changes organizational culture
2. It solves a problem
3. It contributes to quality of life

G&C

Stand-Up Meeting

...a project is in the e..., a period of high stress, or a period of quick change. Or it might just be a period of high stakes, even though you don't expect things to change rapidly—but change must be dealt with responsively, as during the end game.

❖ ❖ ❖

At times of fast change or high stress, it is essential that all members of the organization receive the same information.

When a project is changing quickly, information gets out of date almost instantly. People must have the latest information, or else they risk making obsolete or incorrect decisions. Early in a project; during initial architecture, decisions are made that have lasting impact on the product. These decisions tend to be based on incomplete information; on assumptions which must be validated with others. Because these architectural decisions tend to build on each other, an early incorrect assumption can cause significant directional errors.

At times, the change is dictated by stress or a crisis. Crisis management demands quick response. But quick response demands a coordinated effort—things can go terribly wrong if people don't have the latest information. Architects as well as individual developers can develop tunnel vision, and the low-level decisions can be just as important as the more visible "architectural" ones. As Ed Yourdon said, all things are deeply interwingled. Interdependencies affect both the long-term integrity of the product functionality and structure, as well as the smooth day-to-day functioning of a team that has a shared vision.

Some organizations simply operate at a high change velocity. This requires very tight communication coupling, or else chaos ensues. The most productive organizations we have seen operate this way, although this not their only distinguishing characteristic.

Yet in all these cases, because the need for communication is high, the communication overhead will also be high. And this overhead detracts from the very thing you are trying to accomplish. How can you balance this?

Therefore:

Hold short daily meetings with the entire team to exchange critical information, update status, and/or make assignments. The meet-

How do we use patterns?

- Patterns build things: "If you can't draw a picture of it, it's not a pattern"
- We're going to build...
  - an organizational structure (that relates to the structure of where people sit)
  - backlogs
- We build through one act of repair at a time

G&C

A pattern is:

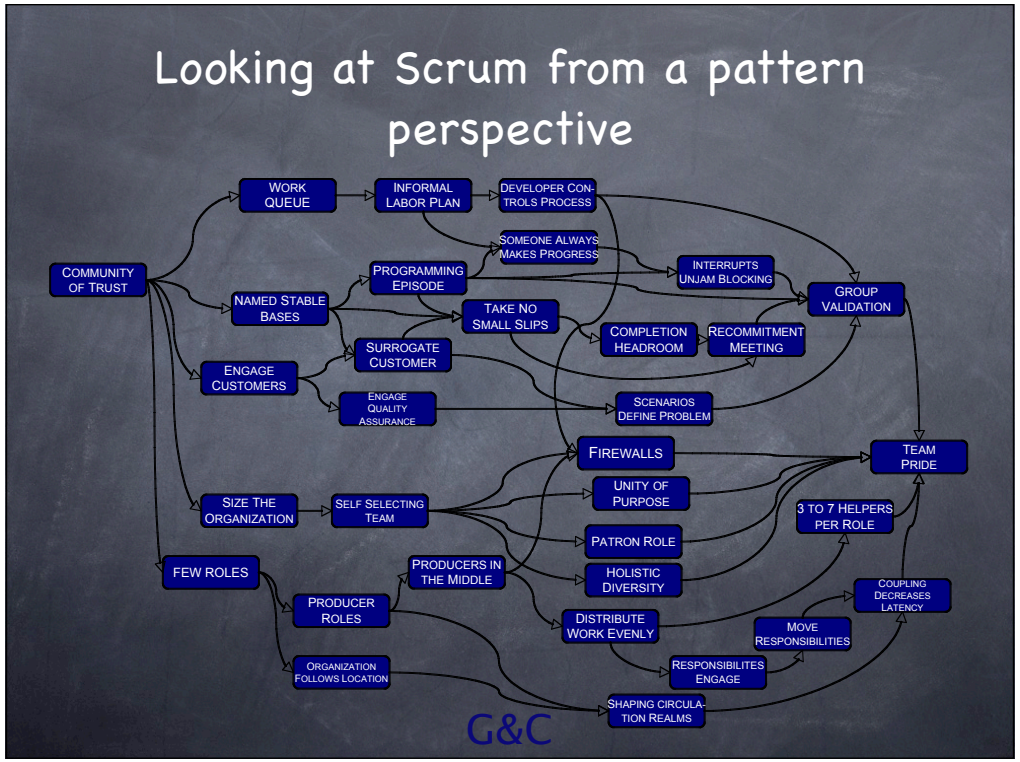
- One of a set of many patterns that work together
- A solution to some small problem of a complex system
- An encoding of proven, socially agreed elements of form
- Both the process to create the form, and the form itself
- An act of local repair to an overall system

G&C

Looking at patterns from a Scrum perspective

- You've implemented Scrum, but...
  - are missing something (maybe an ordered product backlog);
  - want to know why something isn't working
- ... and you're looking for concrete inspiration

G&C



Scrum is, in fact, very complex! Here is a breakdown of Scrum into the Organizational Patterns it comprises. This graph shows one set of paths to implementing Scrum in a low-risk way, one organizational pattern at a time. These patterns go to the deep structure of Scrum that make it work.

However, we can view Scrum from another perspective, which is closer to the process perspective more commonly adopted by process formalists and managers. That perspective is easier to understand but tends to hide the fact that Scrum is hard. We'll start with the simpler description so we have a shared model of understanding and then will come back to some of the difficult points.

Surrogate Customer

- It is sometimes impossible to engage customers
- Developers often don't understand the needs of the customer
- ☞ A person explicitly represents the customer

Developer Controls Process

…an organization has come together to build software for a new market in an immature domain, or in a domain which is unfamiliar to the development team… The necessary roles have been defined and initially staffed.

* * *

A development culture, like any culture, can benefit from recognizing a focal point of project direction and communication. Successful organization s work in an organic way with a minimum of centralized control…

Because developers contribute directly to the end-user-visible artifact, they are in the best position to take accountability for the product. Of all roles, they have the largest stake in the largest number of phases of product development. And there should be no accountability without control…

Therefore: **Place the** *Developer* **role at a hub of the process for a given feature,** in the spirit of Organization Follows Market**.** …

The *Developer* **is the process information clearinghouse.** ...

G&C

Work Flows Inward

... an organization is in place and has been doing work long enough that it can introspect about its structure and workings. ...

* * *

An organization must seek a structure tat best insures that the most authoritative roles make the decisions and carry out the work that adds value directly to the product. ...

During software production, the work bottleneck of a system should be at the center of its communication and control structure. If the communication center of the organization generates more work than it does work, then organization performance can become unpredictable and sporadic...

Therefore: **Work should be generated by stakeholders, especially *Customers*, filter through supporting roles, and be carried out by implementation experts at the center.**

You should not put managers at the center of the communication grid: they will become overloaded and make decisions that are less well-considered, and they will make decisions that do not take day-to-day dynamics into account. ...

G&C

Firewalls

…an organization of developers has formed in a corporate or social context where they are scrutinized by peers, funders, customers, and other “outsiders.” Project implementors are often distracted by outsiders who feel a need to offer input and criticism.

* * *

It’s important to placate stakeholders who feel a need to “help” by having access to low levels of the project, without distracting developers and others who are moving toward project completion. ...

Isolationism doesn’t work: information flow is important. But communication overhead goes up non-linearly with the number of external collaborators.

Many interruptions are noise.

Maturity and progress are more highly correlated with being in control than being effectively controlled.

Therefore:

Create a Manager Role, who shields other development personnel from interaction with external roles. The responsibility of this role is “to keep the pests away.”

G&C

Summary

- Scrum is a pattern language
- Patterns provide insight and encouragement to fix broken Scrums
- Patterns are an incremental path to Scrum adoption if Scrum Shock Therapy shocks you
- The patterns are proven and published

G&C

Online Reading

- Sutherland, Jeff. SCRUM: Another way to think about scaling a project. 11 March 2003, on the web at Jeff Sutherland's SCRUM Log. On how the Organizational Patterns work is the foundation of SCRUM. http://jeffsutherland.org/scrum/2003_03_01_archive.html
- Schwaber, Ken. Scaling Agile Processes. In the Agile Project Management E-Mail Advisor, 3 April 2003. QPW as an example of scaling Agile processes. http://www.cutter.com/project/fulltext/advisor/2003/apm030403.html
- Coplien, James. Borland Software Craftsmanship: A New Look at Process, Quality and Productivity. Proceedings of the 5th Annual Borland International Conference. http://users.rcn.com/jcoplien/Patterns/Process/QPW/borland.html
- Harrison, Neil, and James Coplien. Patterns of Productive Software Organizations. Bell Labs Technical Journal 1(1), Summer 1996. http://users.rcn.com/jcoplien/Patterns/paper11.pdf
- Cain, Brendan, et al. Social Patterns in Productive Software Organizations. Annals of Software Engineering, December 1996. http://www.baltzer.nl/ansoft/articles/2/ase004.pdf
- WikiPedia Organizational Patterns: http://en.wikipedia.org/wiki/Organizational_patterns

References & online resources

- James O. Coplien. Organization and Architecture. In *1999 CHOOSE Forum on Object-Oriented Software Software Architecture,* pages 5-1 - 5-25, March 1999. Bern, Switzerland, Swiss Informaticians Society. A keynote on the architectural impact of organizations. http://www.bell-labs.com/user/cope/Talks/Arch/CHOOSE99/.
- Organisatorisk Agility Program, http://www.nordija.dk/da/Konsulentydelser/OrganisatoriskAgility.html
- Neil B. Harrison and James O. Coplien. Patterns of Productive Software Organizations. *Bell Labs Technical Journal,* 1(1):138-145, Summer (September) 1996. A good summary paper on the techniques and findings in the organizational pattern work. http://www.lucent.com/minds/techjournal/summer_96/paper11/.
- James O. Coplien, Neil Harrison, and Gertrud Bjørnvig. Organizational Patterns: Building on the Agile Pattern Foundations. http://www.cutter.com/offers/orgpatterns.html. Free, but requires signup.
- James O. Coplien. A Development Process Generative Pattern Language. In James. O. Coplien and Douglas C. Schmidt, editors, *Pattern Languages of Program Design,* chapter 13, 183-237. Addison-Wesley, Reading, MA, 1995. http://www.easycomp.org/cgi-bin/OrgPatterns.

References & online resources

- Gabriel, R. Patterns of Software: Tales from the Software Community. New York: Oxford University Press, 1998. For the case study Cope presented. See the chapter on the re-engineering of ParcPlace Systems.
- Neil B. Harrison. Organizational Patterns for Teams. In John Vlissides, James O. Coplien, and Norman L. Kerth, editors, *Pattern Languages of Program Design 2*, chapter 21, 345-352. Addison-Wesley, Reading, MA, 1996.
- Brendan G. Cain and James O. Coplien. A Role-Based Empirical Process Modeling Environment. In *Prodeedings of Second International Conference on the Software Process (ICSP-2),* pages 125-133, February 1993. Los Alamitos, California, IEEE Computer Press.
- Brendan G. Cain, James O. Coplien, and Neil B. Harrison. Social Patterns in Productive Software Organizations. In John T. McGregor, editor, *Annals of Software Engineering,* 259-286. Baltzer Science Publishers, Amsterdam, December 1996.