Course "Softwareprozesse"

OSS and Self-Organization

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik

Part 1:
- What is OSS?
- Who builds it?
- Value

Part 2:
- **Self-organization**
  - Basic infrastructure
  - Typical process
  - Leadership
  - Process innovation patterns

Part 3:
- Quality assurance
- Comparison to agile
- Inner Source

Questions

- What is Open Source SW?
- How important is it?
- Who builds it? Why?
- What is 'value'?
  Who is the 'customer'?
- **How does self-organization work?**
  - Basic infrastructure
  - Typical process
  - Leadership
  - Process innovation patterns
- How does quality assurance work?
- Is this agile?
  Is it modern view?
- Is an open process useful *within* companies?
  - Inner Source

Q

Self-organization: Basic infrastructure DVCS, issue tracker, build, web pages

- Most OSS projects live on a forge, usually github.com

Key infrastructure:

- Distributed version control
  - usually git
  - allows enormously loose coordination to work well
    - pull requests
- Issue tracker
  - product backlog, bug tracker
  - provides a bit more coordination where needed
    - assignee, state, target release
    - all dialog in one place
- CI: automated build
  - just like for XP
- Web pages
  - e.g. GitHub pages
    - often absent for small projects
  - presenting work to public
  - advertising to future contributors
  - display policies

- High variety of **processes** on top of this infrastructure
  - but many typical features:

Self-organization: OSS process: What's typical?

For community OSS, less so f. commercial OSS

[Johnsson01], [CroWeiHow12],

Why?

Driving forces:

- Prototyping is closed
  - Most projects start as closed-source or by an individual
  - Joint prototyping has too many possibilities.
    - Motivations too heteroge-neous for self-organization to work
- User-driven requirements, developers are often users
  - For infrastructure SW; less so for vertical applications

Organization view:

- Collaboration is decentralized
  - not much hierarchical communication
- Planning is informal
  - less so in large projects with heavy company involvement

Self-organization: OSS process: What's typical? (2)

Development style:

- Requirements elicitation:
  - From semi-formal to implicit (by reacting to user requests)
- Iterative process
  - Maintenance is basically bug fixing plus arbitrary re-invention
- Communication is asynchronous, written:
  - too little joint work time
- Strong reliance on technical infrastructure
  - version archive, issue tracker
- Architectures are designed for modularity:
  - To minimize coupling and hence coordination effort
    - e.g. modules in Apache, plugins in Eclipse etc. etc.
- Release:
  - Wide variety, from *"release early, release often"* to fixed intervals with explicit stabilization phases

Self-organization: OSS process: What's typical? (3)

Social processes:

- New-member socialization:
  - mostly driven by would-be member
    - acts as a people filter
  - sometimes: entry scripts
- Decision-making/leadership:
  - centralized or decentralized styles (see later)
    - a project trends towards decentralized over time
  - leadership is often implicit and often shared
- Coordination, collaboration:
  - task self-assignment
    - "do-ocracy"
  - collaboration mostly implicit (see next slide)
- Knowledge management:
  - difficult (distribution!)
  - community of practice
    - people as institutional memory
  - media: ad-hoc (mailing list) or permanent (e.g. wiki) or in-between (issue tracker)
    - Each type has its own downsides

Self-organization: Collaboration and coordination

[HowCro14]:

- OSS projects work such that individual tasks are solved by individuals, not teams
  - almost always,
  - greatly reducing coordination effort.
- Consequence for large tasks:
  - They often get deferred for a long time,
  - which would hardly be acceptable for a commercial organization.
- But eventually a work breakdown is usually found that makes them possible,
  - namely after enough enabling work has been finished.
- Strong SW modularity helps this process
  - but is not strictly a prerequisite,
  - so the process may or may not produce highly modular designs.

Questions

- What is Open Source SW?
- How important is it?
- Who builds it? Why?
- What is 'value'?
  Who is the 'customer'?
- How does self-organization work?
  - Basic infrastructure
  - Typical process
  - **Leadership**
  - Process innovation patterns
- How does quality assurance work?
- Is this agile?
  Is it modern view?
- Is an open process useful *within* companies?
  - Inner Source

Q

Self-organization: Leadership

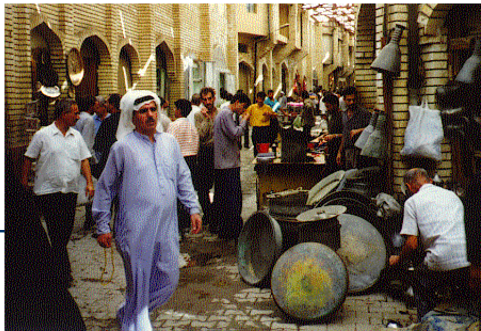
Source:

- Eric Raymond: *"[The Cathedral and the Bazaar](#)", 1997-2000*

Describes two styles of software development:

- **Cathedral style**: (=classical-view commercial world)
    - (now less strongly so with agile processes)
  - integrated groups of skilled individuals plan thoroughly and implement with care and no haste
  - *"built like cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation, with no beta to be released before its time"* Q

- **Bazaar style**: (=most of the open source world)
    - (now often less strongly so with more and larger companies involved)
  - open for participation by everyone, hardly any central planning, no competence guarantee, quickly evolving
  - *"resemble a great babbling bazaar of differing agendas and approaches"* Q



Self-organization: OSS leadership and decision-making

- By and large, OSS projects tend to have a meritocratic leadership model
  - Influence is won by making valuable contributions to the project
  - and by exhibiting technical and judgmental competence
- (exceptions possible when corporate sponsoring is present)

This statement raises two questions:

1. What is the process (in terms of milestones) of gaining influence for an individual?
   - Put differently: Are there clearly different degrees of influence that can be easily observed? (An "OSS career")
2. How does actual decision-making work?
   - Given that influence cannot easily be quantified

Self-organization: The OSS career

The typical career of an active OSS project participant:

1. Knows product
    - User
2. Knows process/project
    - Mailing list member: 2.1. Follows and 2.2. participates in the discussions in the project
3. Contributes suggestions to product
    - 3.1. Sends in defect reports or helps clarifying issues
    - 3.2. Sends in defect corrections ("bug fixes", "patches") to be checked and accepted by the developers
4. Has write-access to product
    - Developer status: can modify the source code version archive
5. Has meta-write-access to product
    - Can grant others write-access. Called differently in different projects (core developer, maintainer, leader)

Perhaps more stages here

Self-organization: The OSS career (2)

- In small projects there is often a single person with meta-write access who makes the decision at his/her own discretion
- Some large projects define various roles and behavior explicitly and may have formalized decision-making rules and even bodies for granting write-access (*join-scripts*), e.g.
  - http://httpd.apache.org/dev/guidelines.html , http://docs.python.org/devguide/ , https://wiki.documentfoundation.org/Development/GetInvolved
  - Some large projects also discriminate many different kinds of contributions (and corresponding roles) more clearly
    - e.g. Development, QA, Localization, Marketing, Documentation, Website Dev.

- See also https://opensource.guide/how-to-contribute/
  - general, project-agnostic advice

Self-organization: OSS decision-making (1)

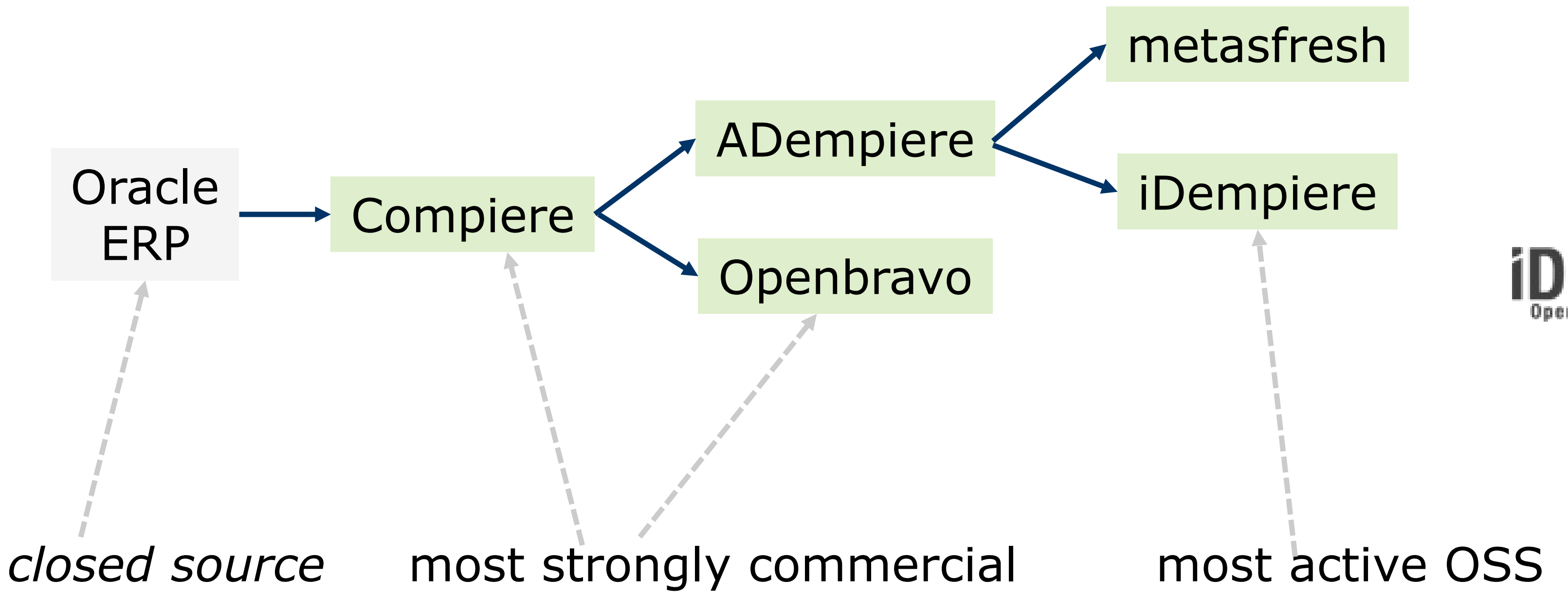
The leadership structure (formation of opinion) of OSS projects is spread over a spectrum with the following poles:

- Egalitarian:
  - In any issue, the influence of an individual depends mostly on convincing argumentation.
- Leadership group:
  - The influence depends mostly on the individual's general reputation
    - which may be formalized or not
- Note: A leadership group without merits could not persist or would lead to *forking* *(next slide)*. Thus, the difference between the poles is <u>not</u> huge.

also: https://opensource.guide/leadership-and-governance/

Self-organization: Forking

- Forking: Founding a separate project based on the same code Q
  - Happens when too-large parts of an OSS community are too unhappy with the way the community progresses.
    - Possible as a consequence of OSS licencing ("free software")
- Example: Compiere ERP



Self-organization: OSS leadership type case studies

Most larger projects fall into one of the following categories:

1. Democratic model
2. Benevolent dictator model
3. Industry-based
4. OSS foundation projects

see subsequent slides

OSS leadership types 1: Democratic model

- A group of people use explicit democratic decision processes and drive the project like a society drives a democratic state
- Example: **Apache** software foundation
    - Quotes from http://www.apache.org/foundation/how-it-works.html#management (as of 2024-01)
  - "Projects are normally auto governing and driven by the people who volunteer for the job. […] ***"do-ocracy"*** -- power of those who do. This functions well for most cases.
  - When coordination is required, projects make decisions with a **lazy consensus approach**: a few positive votes with no negative vote is enough to get going. […]
  - […] a PMC member registering a negative vote must include an **alternative proposal** or a detailed explanation […].
  - […] In the great majority of cases, the concerns leading to the negative vote can be addressed.
  - This process is called ***"consensus gathering"*** and we consider it a very important indication of a healthy community."

OSS leadership types 2: Benevolent dictator model

- A single highly respected person makes all important decisions
- Examples: **Linux**, Python

Linus Torvalds

- In 1991, the Finnish student Linus Torvalds started writing an operating system kernel
  - His message on comp.os.minix in August 1991: http://groups.google.com/group/comp.os.minix/msg/b813d52cbc5a044b
  - *"I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. […] It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks"*
- Linux (kernel/arch/drivers) now consists of 15 MLOC
- Yet Torvalds' few deputies still have to accept every change to this code to make it official

OSS leadership types 2: Benevolent dictator model (2)

- Guido van Rossum started developing the programming language **Python** in 1990
  - In 1996, he wrote (in the introduction of Mark Lutz' book "Programming Python"): *"[…] in December 1989, I was looking for a 'hobby' programming project that would keep me occupied during the week around Christmas."*
- Today, Python is one of the most popular languages
  - for web, scripting, scientific programming, teaching, …
- The Python development community calls van Rossum the *"Benevolent Dictator For Life"* (BDFL)
  - (he stepped down from that role in 2018 because he found being a Dictator too burdensome)

Guido van Rossum

OSS leadership types 3: Industry-based

- Most project members come from one industrial employer
  - they often work full-time for the project
  - and are being paid for it
- Examples: Mozilla Firefox, JBoss/WildFly

Where does the money come from?

- **Firefox**: Mozilla Foundation (Google search box fee)
- **WildFly**: Red Hat Inc. (professional services)
  - formerly JBoss Inc., sold for US$ 420 mio after 7 years

OSS leadership types 4: OSS foundation projects

- A formal organization (often called a foundation) is build in order to host a significant group of related projects that have something important in common
  - such as technology, leadership/governance principles, or philosophical principles
  - May or may not have a main sponsor

Example:

- Apache Software Foundation (ASF)
  - is a non-profit corporation with 501(c)(3) U.S. charity status
    - members are individuals, new ones accepted by current member vote
  - Goals: Support OSS projects , create a reputable receiver for donations , provide legal shelter to project participants , protect the "Apache" brand
  - Runs >350 projects, including many highly regarded ones
  - Runs an "incubator" for systematically integrating further projects into the foundation

Apache Incubator

- As of 2024-01, has 19 candidates
- Has a detailed formalized process for how a project can become an ASF project:
- 1. To become a candidate, a project must write a proposal and must have the support of
  - a Champion: An ASF member
    - http://www.apache.org/foundation/members.html
    - as of 2024-01, at least 679 individuals were members
  - a Sponsor: Either the ASF Board or an Apache Top-Level Project or the Incubator Project Management Committee
- 2. To become an ASF project, the candidate must
  - put all code under Apache license, resolve trademark issues
  - work in "the Apache way" (large community, voting, meritocracy, conflict handling, release planning, etc.)
  - create synergy with other Apache projects

OSS leadership types 4: OSS foundation projects (2)

- The Free Software Foundation (FSF, home of GNU)
  - Original goal was a completely free Unix OS
    - GNU built system utilities, shell, compilers, C library etc.
  - Main Principle is that of Free Software (GPL license)
  - Now mostly rallying free software, not developing it
- Eclipse Foundation
  - Initially an industrial consortium around IBM
    - Borland, MERANT, QNX, Rational, Red Hat, SuSE, TogetherSoft, Webgain
  - now a foundation with many members in different membership types
- Others: OpenStack, Linux, Gnome

Apache, FSF, and Eclipse are *super*-different!

Self-organization: Leadership type suitability

- Could Linux or Python be led in Apache style? Q
  - The extreme quality requirements of an OS core or a clean programming language are easier in a BDFL model
- Could Torvalds or van Rossum lead the whole ASF? Q
  - The extreme scale (volume and diversity) of the ASF projects can only be handled by a larger set of leaders.

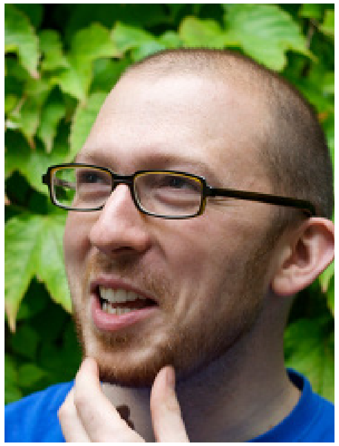
Questions

- What is Open Source SW?
- How important is it?
- Who builds it? Why?
- What is 'value'?
  Who is the 'customer'?
- **How does self-organization work?**
  - Basic infrastructure
  - Typical process
  - Leadership
  - **Process innovation patterns**
- How does quality assurance work?
- Is this agile?
  Is it modern view?
- Is an open process useful *within* companies?
  - Inner Source

Q

Self-organization: Process Improvements and OSS

This part is concerned with research performed in AG Software Engineering


Christopher Özbek

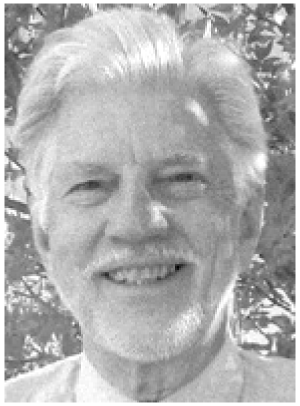
- Assume we want to perform process improvement
  - important part of self-organization
- We know that this requires a lot of effort and time
- In a company, a decision will be based on hierarchy (classical view) or joint company interest (agile)
  - Neither exists in "real" OSS projects
- How does the equivalent process work in an OSS context? Q
  - No central authority over project members, different interests → decisions are more complicated
  - Members are distributed → asynchronous discussion
  - Some improvements that are useful conventionally may not be useful here

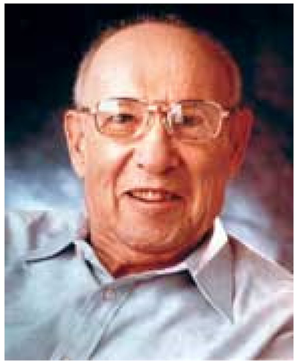
Definition "innovation"

- Definition:
  Innovation means that a group adopts a new practice
  - Conforms to the usage by important authors, e.g. Everett Rogers, Peter Drucker, Harold Evans
  - This definition is operational: observable, executable

- "Practice" refers to
  - habits, routines, and other forms of embodied recurrent actions
  - that are chosen and performed without conscious thought.

- In this sense, software process improvement is innovation


Rogers


Drucker

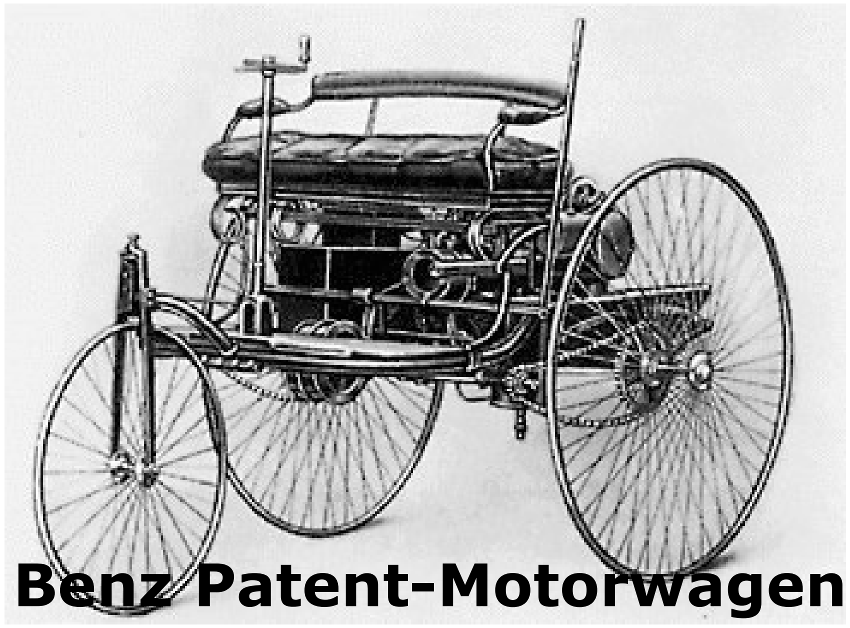
Innovation vs. invention

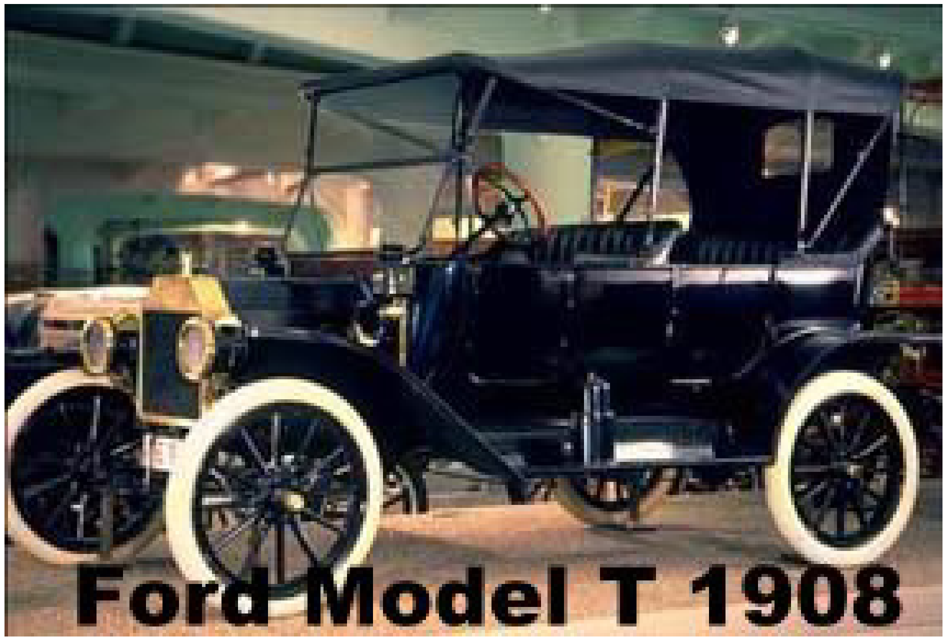
- Invention is different from innovation.
  - Invention means to create something new,
  - but does not require that anyone accept or adopt it.

1. Most inventions never become (or lead to) innovations
2. Many innovations are not brought about by the inventor
3. The same invention can lead to many innovations
   - one per group adopting it
4. Innovation need not be unusual, widespread, or radical
   - and can happen slow or fast

Invention vs. innovation

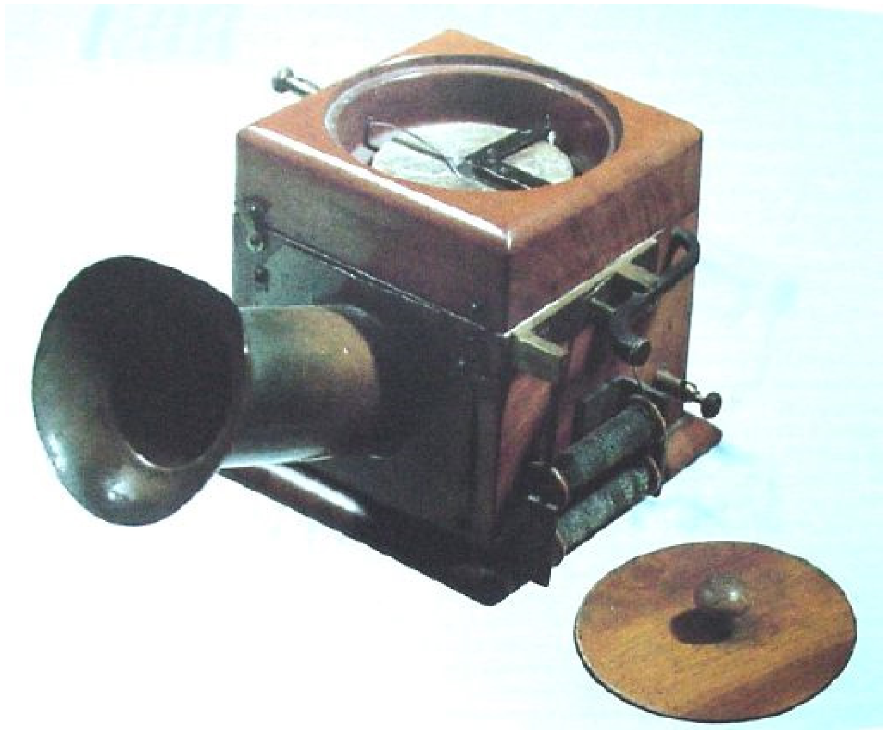
- Carl Benz's first car was an invention
- but only Henry Ford's Model T brought the innovation
  - it was sufficiently cheap, reliable, available


Benz Patent-Motorwagen 1886


Ford Model T 1908

Invention vs. innovation

- Johann Philipp Reis invented the telephone 1860
  - others followed: Antonio Meucci, later Elisha Gray
- Alexander Graham Bell did it again 1876,
  but then founded the Bell Telephone Company



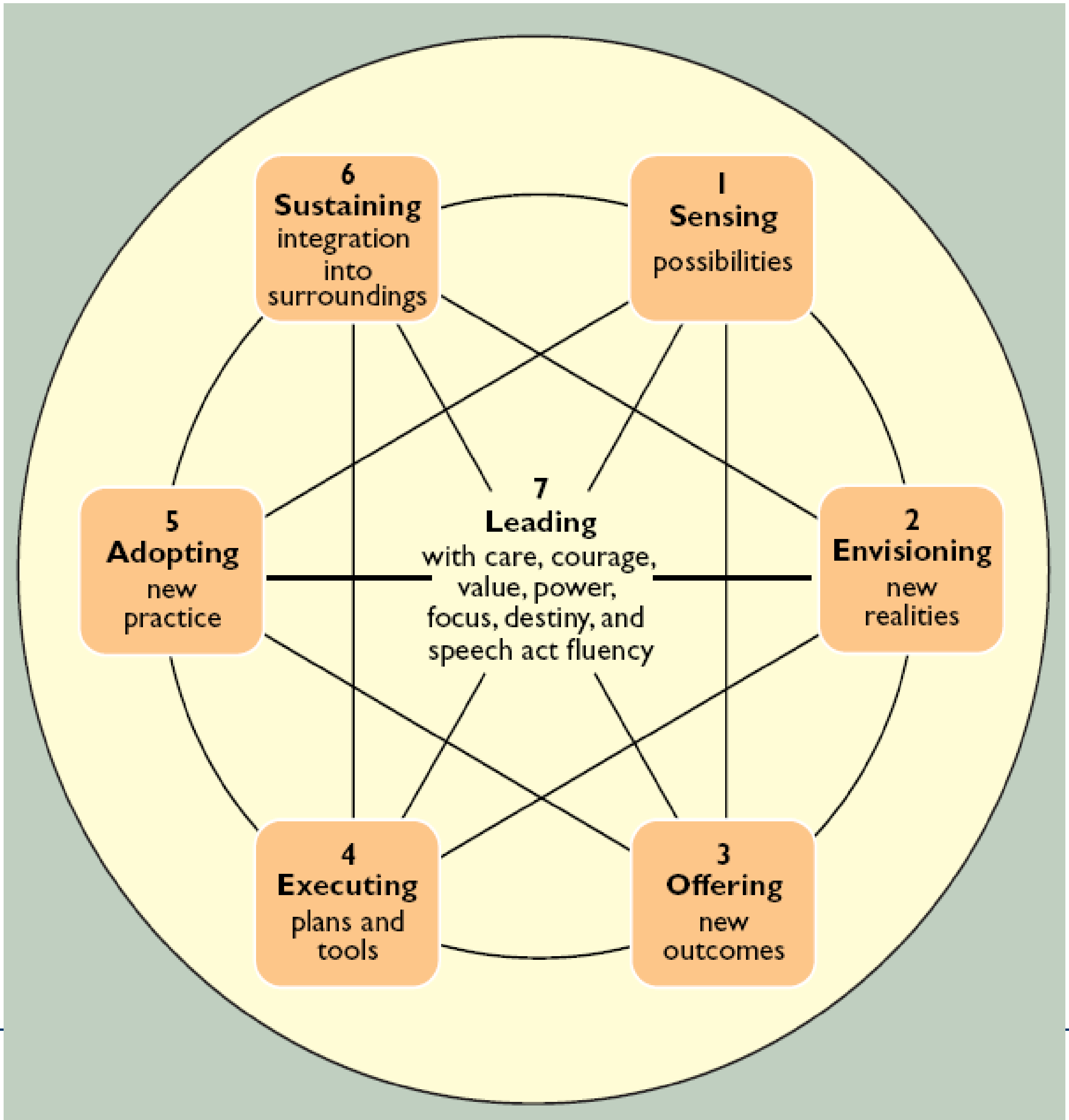
Innovation as an active social process

[DenDun06]

- Successful innovation is performed by following certain practices
- These practices can be trained and learned
  - presented in the form of a generative framework
  - (Relevant for OSS participants who want to improve projects)
- Technical capabilities are <u>not</u> at the heart of these practices

[DenDun06] The generative framework: "Personal Foundational Practices"

- 1 to 2: invention
  - 3,4: transition
- 5 to 6: adoption
- Not sequential steps!
  - more like parallel processes
- Each practice has both verbal and non-verbal aspects


6 Sustaining integration into surroundings
1 Sensing possibilities
5 Adopting new practice
7 Leading with care, courage, value, power, focus, destiny, and speech act fluency
2 Envisioning new realities
4 Executing plans and tools
3 Offering new outcomes

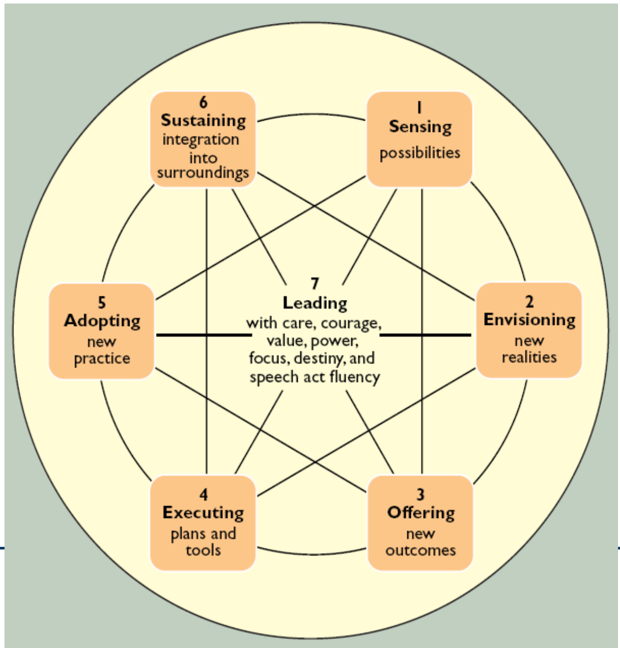
Application to OSS process improvements

- How do process innovations proceed in OSS?
  - And what can we learn from that? In particular:
- What does a would-be innovator need to do in order to maximize the chance of successful adoption?
  - How to identify candidate pairs of invention and project?
  - How to identify key people in the project?
  - How to communicate with the project?

Case study: The Moderator role

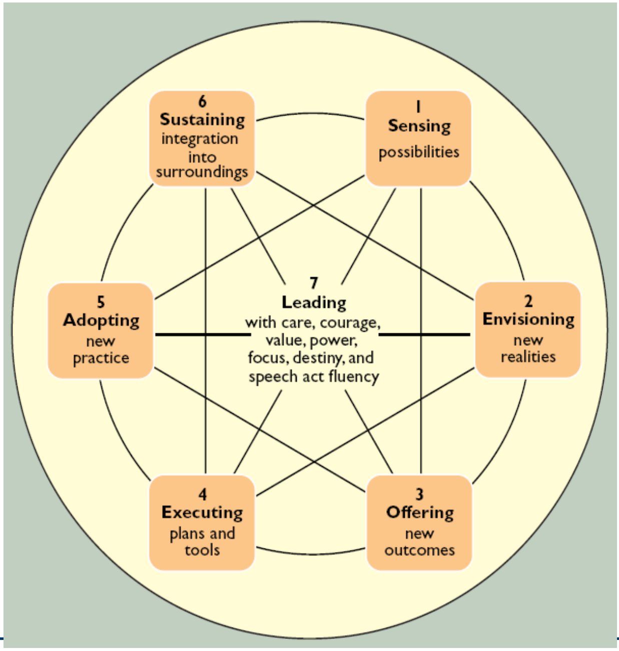
Participant observation study:

- Communication and coordination are difficult in OSS projects
- We **1_sensed** that it might be helpful to actively and explicitly promote coordination-related information in such projects
- We **2_envisioned** a new role in OSS projects, the Moderator, whose task is information management:
  - explicitly collecting and organizing information that speeds up information access for many participants (in particular new ones) and avoids redundant questions or searches
- We **3_offered** this "invention" to a project (GNU classpath)
  - We offered to set up a wiki
  - There we could collect and structure information regarding e.g. design decisions


6 Sustaining integration into surroundings
1 Sensing possibilities
7 Leading with care, courage, value, power, focus, destiny, and speech act fluency
5 Adopting new practice
2 Envisioning new realities
4 Executing plans and tools
3 Offering new outcomes

Case study: The Moderator role (2)

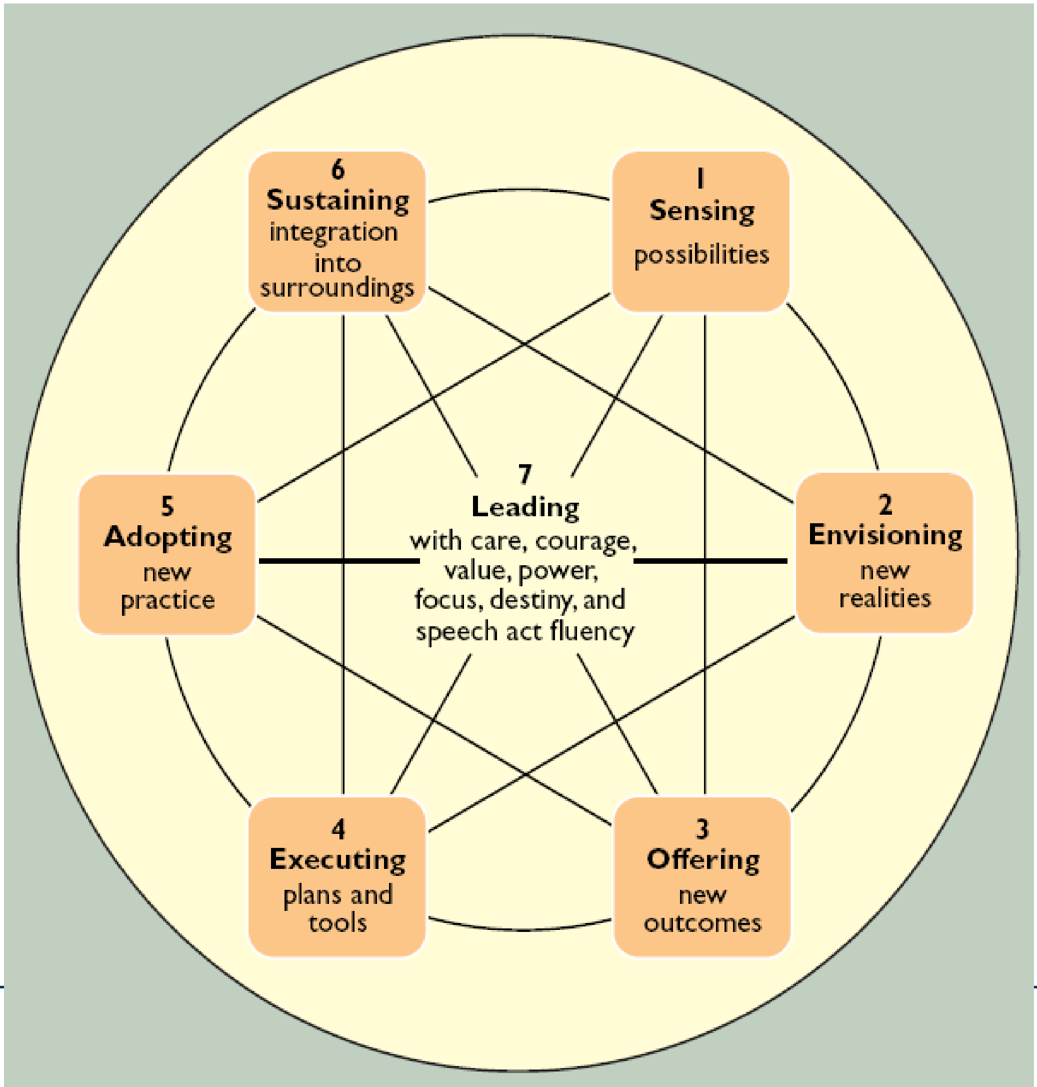
- The offer was accepted. We **4_executed**
  - by actually setting up the Wiki
  - by actually compiling initial information found in the mailing list archive and putting it there regarding (a) design decisions, (b) newbie instructions, (c) current development topics
- We continued maintaining this information, adding more from time to time and announcing it via email, thus triggering **5_adopting** the new practice
  - After some time, a few other project members started using the platform, too
  - Also for new purposes, such as arranging physical meetings
- Specific actions for **6_sustaining** the practice did not appear to be necessary
  - The Moderator role has apparently been distributed and filled since


6 Sustaining integration into surroundings
1 Sensing possibilities
5 Adopting new practice
7 Leading with care, courage, value, power, focus, destiny, and speech act fluency
2 Envisioning new realities
4 Executing plans and tools
3 Offering new outcomes

Freie Universität Berlin

Case study: The Moderator role (3)

- The details of our **7_leading** that made the effort successful still need to be understood
  - analyzing who did what when why
  - or not
- In order to understand the causation in the process, we need more examples of it



1 Sensing possibilities
2 Envisioning new realities
3 Offering new outcomes
4 Executing plans and tools
5 Adopting new practice
6 Sustaining integration into surroundings
7 Leading with care, courage, value, power, focus, destiny, and speech act fluency

Process Improvements and OSS: Research method

- We performed participant observation once
  - but that is far too time-consuming
- We switched to searching for process innovation episodes on project mailing lists
  - chose medium-sized projects (10 to 50 members)
  - scanned the mailing lists of several hundred projects
    - and picked **12 projects** for analysis
  - scanned thousands of emails for innovation episodes
  - extracted the messages for about **100 such episodes**
  - analyzed them in detail using GTM to find innovation **patterns**

- Innovation episodes:
  - variable size (#messages, #participants, #days)
  - very different topics, some types of them recurring
  - often unsuccessful

Process innovation pattern 1: Partial migration

- Context:
  - A process change was proposed
  - Many find it reasonable
- Forces
  - The change involves a lot of work for one person
    - and some work for everybody
  - It is risky or some members do not like it yet (are change-averse)
- Example:
  - Switch the version mgmt. from CVS to Subversion or from Subversion to a decentral system (e.g. git)
- Solution:
  - The change is made only for a fraction of the project at first
    - e.g. new repository created for one subsystem only
  - then tried out and adapted gradually
    - in order to distribute the workload and allow members to adapt slowly

Process innovation pattern 2: Adapter innovations

- Context: A sensible process change was proposed

- Forces: Some members cannot or do not want to accomodate the future situation.
  - → Resistance.

- Example: ditto, change of version management software

- Solution: Create an adapter that allows those members to more or less stay in the previous mode
  - at least for a while

Process innovation pattern 3: Reduce enactment scope

- Context: A sensible process change is proposed
- Forces: It involves a lot of work compared to its importance (or at least many members perceive it that way), or the benefits are unclear
- Example: Clean up bug tracker database after a release.
- Solution: Frame the suggestion as a one-time activity only. Wait and see how it worked out. Only then introduce it as a process change

(We found a few more such patterns, also smaller tactical ones.)

Common theme of the patterns?

- Partial migrations, Adapters, Reduced enactment scopes
- Reduce amount of resistance
  - by reducing the attack surface
  - "leading without coercion" (Raymond)
- Why is that needed?
    - Isn't the proposed process indeed better? Q
  - Because process change is cultural change ("*community-specific ideas about what is true, good, beautiful, and efficient.*")
    - culture sticks!

Summary

- OSS projects strongly rely on typical technical infrastructure
- Processes vary a lot
  - but have a core of typical elements
- Leadership is typically meritocratic
  - with sometimes huge influence of admired top people
- Process innovation is difficult
  - because of heterogeneity of players and
  - because process change often means culture change
  - but behavior patterns for reducing change resistance exist

Thank you!

HOME ORGANIZATION TIP:
JUST GIVE UP.