Apache CON

Open Development in the Enterprise

November 11, 2011

Leading the Wave
of Open Source

Introduction

What can corporate IT learn from leading open development communities?

Both ***principles*** and ***techniques*** offer value

Challenges must be overcome to realize value

Principles

Transparency

- Decision-making and actions are observable
- Events of interest are published and recorded

Meritocracy

- Influence on decisions is based on merit
- Merit is earned in public

Community

- Common interest
- Common experience / identity
- “Community before code”

Techniques

Open Standards

Using open standards in systems design and standards-based tools for development

Collaboration Infrastructure

Systems supporting communication and coordination: *repositories, trackers, forums, build tools*

Meritocratic Governance

Merit determines influence on decisions

Community-based governance structures

Challenges

- Choosing the right opportunities
  - "Open everything" does not work
- Clue
- Resistance
  - to change
  - to loss of control
- Rewarding merit
- Business focus
  - Accountability
  - Control

Principles

Principles - *Transparency*

Collaboration

Transparency invites collaboration

Reuse

You can only reuse what you can see

Quality

More eyeballs mean better quality

Measurement

Transparency enables measurement

Transparency ⇨ *Collaboration*

Transparent processes enable people to ***see connections***

Transparent decision-making encourages contribution and improves ***leverage***

Transparency encourages ***feedback*** and dialogue

Example: infrastructure planning and development

Transparency ⇨ *Reuse*

People can reuse assets they can **find** and **understand**. Transparency increases both the likelihood that people will find assets and that they will understand them.

Transparency means you can see where development is headed so you can **plan** reuse.

Transparency can give confidence in **quality**, making reuse more likely.

Example: Transparent bug / enhancement tracking, feature lists, release plans makes reuse of shared components easier to plan and execute

Transparency ⇨ *Quality*

"Too enough eyeballs, all bugs are shallow"

Open discussion can lead to better decisions

Errors may be seen early

Transparency encourages and facilitates peer review

“Lurkers” sometimes see things that closed group may miss and they ***learn*** things by observing

Examples: Opening source and dev process for common components; peer reviews; problem management; operations

Transparency ⇨ *Measurement*

Transparency makes process visible

Development events can be tracked

Active and inactive people, code, sub-communities can be discovered

Metrics can be less subject to debate when derived from transparent processes

Examples: Contributions to shared assets; roadmap progress; source quality metrics; test execution metrics; problem management

Principles - *Meritocracy*

- Technical decisions made by technical experts
  - Better informed decisions
- Role models
  - Merit provides examples
- Earned authority
  - "Natural" leadership

Meritocracy ⇨ *Better Decisions*

Better information

Better analysis

Faster consensus

Less prone to vendor / hype distortion

Less prone to stupid groupthink

Not the rule in corporate settings today, more common in startups / small companies

Examples: technology product selection; technology standards; organization structure

Meritocracy ⇨ *Role Models*

People with community-earned merit provide **examples of behaviors** that the community likes

Merit-earning people provide **examples of community contributions** that have merit

Imitating people with merit **helps others develop** and usually leads to good results

Examples: peer reviews; open certification process; formal mentoring; committer / PMC member concepts for shared assets

Meritocracy ⇨ *Earned Authority*

When leaders earn ***merit*** and gain authority as a consequence, their authority is perceived as ***earned*** rather than by title or circumstance.

People follow these leaders. People follow leaders whose authority is based on merit that they know about.

People support decisions made by people with earned authority.

Examples: major refactoring / uplift; service restoration; process change; talent pipeline / promotions; strategy and standards

Principles - *Community*

- Loyalty
  - Community breeds loyalty
- Durability
  - Communities can create durable assets, processes and culture
- Shared vision
  - Vision grounded in common experience and personal connection

Community ⇨ *Loyalty*

Loyal people contribute

Loyal people support one another

Loyal people work out conflicts and compromise to support the community

Loyalty begets loyalty

Examples: retention; service restoration; crunch time; managing conflict

Community ⇨ *Shared Vision*

Community shares experience

Vision can be developed in the community

Vision guides community and holds it together

Vision helps community adapt to change

Examples: transformation plans; refactoring; innovation

Community ⇨ *Durability*

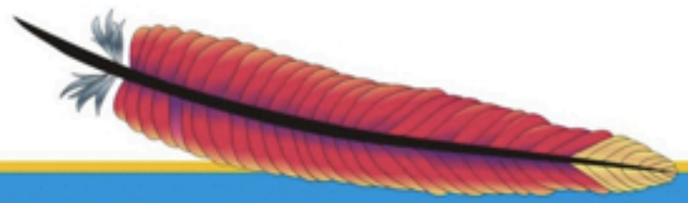
Communities can survive individual departures

Community developed assets can be revived / restarted when time is right

Community developed assets are robust against changing requirements

Examples: shared code components; architecture communities

Techniques



Techniques

Open Standards

Using open standards in systems design and standards-based tools for development

Collaboration Infrastructure

Systems supporting communication and coordination: *repositories, trackers, forums*

Meritocratic Governance

Merit determines influence on decisions

Community-based governance structures

Techniques - *Open Standards*

Faster ramp-up

- Standards provide common background

Easier setup

- Easier to get started, get up to speed

Interoperability

- Key to success in heterogeneous environments

Techniques - *Collaboration Infrastructure*

Communications

Support asynchronous, geographically dispersed collaboration, **fewer meetings**

Repositories

Enable transparency, discoverability

Trackers

Coordinate collaborative work, transparency

Build tools

Enable consistent, independent, repeatable builds; support process discipline, quality assurance, productivity,ramp-up

Techniques - *Meritocratic Governance*

Decisions

Influence on decisions determined by merit

Structures

Governance structures supporting merit-based decision-making

Examples: PMC managing roadmap / stds, shared components; user/contributor/committer roles for common code as well as strategy / standards content; review and approval of changes to standards, roadmaps, shared assets; peer voting on releases

Challenges

Choosing the Right Opportunities

Good	Bad	Ugly
Open development of shared assets	Open development in specialized areas with small teams	Building communities that have nothing to do with day jobs
Meritocracy principles integrated into performance management	Meritocratic decision-making process, but decisions not binding	Merit earned and acknowledged, but not rewarded
Open development infrastructure introduced as part of process improvement	Open development process introduced with no infrastructure support	Open development principles mandated with no process or infrastructure support

Choosing the Right Opportunities

Transparency

Almost always a **good thing**, but need to be careful to avoid distraction / low signal/noise

Meritocracy

The more technical the domain, the more valuable this is. Needs to be inclusive and harmonized with hierarchy.

Community

Like transparency – always good in principle, but can have low / no value if not conceived and nurtured correctly

Open Standards

Always

Collaboration Infrastructure

Take “appropriate technology” approach. Can be a vector for planting the open development meme

Meritocratic Governance

Apply selectively, starting in technical domains with leaders who already operate consistently with the ideas. Formalize criteria.

Clue

Community is not the same as team

Contribution is work

Community requires investment

Transparency is not a threat

Collaboration means compromise

Driving results means driving consensus

Resistance to Change

If it ain't broke...

Communication can be annoying at first

Need to learn new tools and processes

Closed processes and decision-making are the norm

Administrivia can get in the way and provide a convenient excuse to defer / delay

Resistance to Loss of Control

Open development can be directed, but can't be micro-managed

Schedules and timelines

"Owning" decisions

Fear of failure

Accountability, reporting, leadership communications

Meritocracy / Rewards Mismatch

Reward system may not be based on merit

Merit needs to be rewarded to proliferate

Merit needs to be rewarded to be respected

Maintaining Accountability

Community ownership does not guarantee owners are always available and responsive

Not always clear who owns decisions or when decisions have been made

Easy to blame lack of engagement / community support for bad decisions or work products

Control and support responsibilities need to be managed explicitly

Maintaining Control

Communities are harder to direct and focus than individuals

Company value needs to drive community, not vice-versa

Roadmap needs to be explicit and directive (another reason it is good for this to be an open development product)

Timelines, feature sets, quality, packaging and deployment objectives have to be explicit and accepted as largely “exogenous”

Maintaining Business Focus

Community interest must align with company interest

Business leaders have to be welcome and engaged in community

Merit is not just technical and has to be linked to business results

Open development projects need to deliver value – "**show value early, show value often**"

Open development should not be used as a means to invest in projects that have weak or no business case

Questions?