Martin Brandtner
Harald Gall

Open software development: an overview

TECHNICAL REPORT – No. IFI-2012.04

2012

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Martin Brandtner, Harald Gall
Open software development: an overview
Technical Report No. IFI-2012.04
Software Evolution and Architecture Lab
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
http://seal.ifi.uzh.ch

Open Software Development: An Overview

Development Processes and Partner Programs

Martin Brandtner, Harald Gall
Department of Informatics
University of Zurich
8050 Zurich, Switzerland
`{brandtner,gall}@ifi.uzh.ch`

Abstract *- The interlinking of software systems requires software vendors to implement standardized data exchange formats or an API to allow direct access to the software system. In both cases, a software vendor has to open the development in terms of communication with partners and customers. This report provides an overview of open development processes and discusses key findings for such a process.*

Contents

1 Introduction

The development of software systems can take place in different environments (e.g. in a community, in a company, etc.) and under different degrees of openness to the outside world. The last point, the openness to the outside world is often misunderstood and wrongly equated with the openness of source code. In the last years, more and more software projects are developed in a community or at least with the support of a community. Openness of software development can be split into openness of the development process and openness of the source code. Many companies opened their development process to gain innovative ideas through a community. In this technical report, we analyze community-driven and partly company-driven development processes for software systems. We provide an overview of common commercial partner programs and of three community-driven software development processes, namely: Eclipse Development Process, Java Community Process 2, and Khronos Group Development Process. Based on this, we discuss key findings for a successful open development of software systems.

2 Community-driven Development Processes

Many successful and well-known software products such as Apache or Eclipse are developed under open source with a development process influenced by open development. Many new or recent closed source software projects (e.g. Java-Runtime etc.) rely on such open and transparent development processes. This section provides an overview of three open development communities with focus on management, participation and innovation handling.

2.1 Eclipse Development Process

The Eclipse development process is the driving force behind the Eclipse project. It describes basic technical handling, project management and communication between participants. Especially cultivation of a transparent and profitable ecosystem for all community members (individuals and companies) is a key principle of the Eclipse project.

"It is an explicit goal of the Development Process to provide as much freedom and autonomy to the Projects as possible while ensuring the collective qualities benefit the entire Eclipse community." [1]

Participation Model

An Eclipse project consists of three major groups of stakeholders: contributors, adopters, and users. A project member might be an individual person or a company. Figure 1 shows an overview of interaction between each group. The contributor group is responsible for the core technical development and coordination of the project. This group has to define the development roadmap, do the implementation and keep the community alive. The second group is the users group.

This group uses the software and provides feedback. A big user group offers a great attraction to companies and developers. These companies may become either a member of the users group or an adopter. Members of the adopters group develop plug-ins or use the underlying project as framework (e.g. Eclipse RCP) for their own products. The benefit for an adopter is an active and working community, which can be directly addressed.

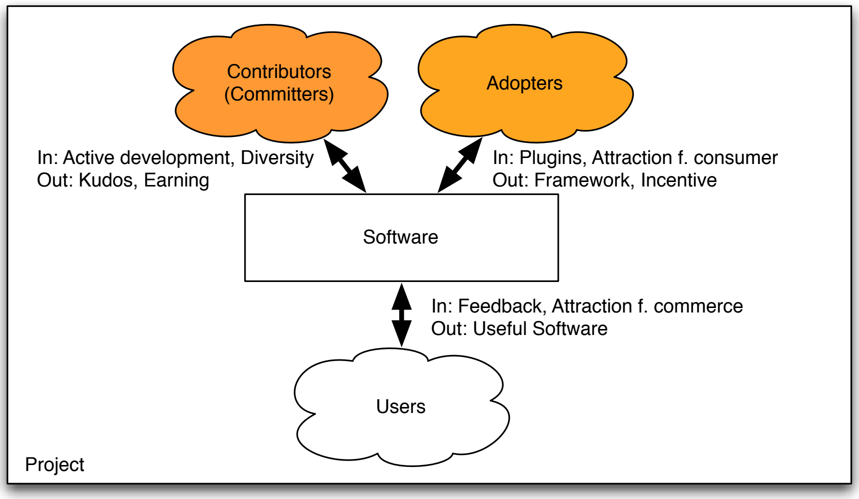
Contributors (Committers)
Adopters
In: Active development, Diversity
Out: Kudos, Earning
In: Plugins, Attraction f. consumer
Out: Framework, Incentive
Software
In: Feedback, Attraction f. commerce
Out: Useful Software
Users
Project

Figure 1: Participation Model with information flow of Eclipse Development Process

Management of Development Process

The management of the development process is influenced by so called councils. A council is comprised of strategic members and representatives of the project management committee. Three councils support and guide the project. The requirements council takes account of the roadmap and has to define the feature set of a software version. The definition of a feature set is based on reviews and incoming requests. The planning council is responsible for the cross-project planning, coordination, and integration issues. The purpose of the architecture council is to ensure principles of the development process and maintenance of the Eclipse platform architecture. Only members with sufficient experience can be part of this council. There is a more detailed definition for the main tasks of each council (see [1]).

Depending on the type of a project, a lead is either called project lead (in case of a sub-project) or project management committee (in case of a top-level project). A committee consists of one or more leads and zero or more members. The lead of a project keeps responsibility that all members follow the development process, which is influenced by the councils.

To keep the whole management process transparent regular reviews take place. There are nine different types of reviews and each applies to a fixed review process. In contrast to other parts of the development process the description of these review processes is quite strict and detailed (see [1]).

Promotion and Handling of Innovation

The best way to introduce innovation into an existing project under the Eclipse development process is a so-called Incubator project. An incubator project is always a sub-project of a top-level project and it must be covered by the scope of the owning project. The rule set is smaller and less restrictive as for a regular Eclipse project.

At least one committer of the incubator project has to be a member of the top-level project and an incubator project never has releases, but only builds and downloads are allowed. The result of such a project can either be used as base for a new project or for integration into the top-level project. To avoid uncontrolled development only top-level projects in a mature phase are allowed to own incubator projects.

For short, these projects provide a playground for innovation, new ideas or growing functionality with a loose but controlled binding to a related project. The creation or termination of an incubator project happens through a review of the current top-level and incubator project.

2.2 Java Community Process (JCP) 2

The Java Community Process is mainly used for development and evolvement of the Java platform. Experts and individuals with deep knowledge of a specific topic are the basement of this process. Before a draft is released to the public at least two different groups of experts have reviewed it. It is also important to clarify that this process only supports the establishment of a specification and not the development of a product it self.

"The JCP produces high-quality specifications in internet time using an inclusive, consensus building approach that produces a specification, a reference implementation (to prove the specification can be implemented), and a technology compatibility kit (a suite of tests, tools, and documentation that is used to test implementations for compliance with the specification)." [2]

Participation Model

The participation model of the Java Community Process defines and knows a large set of different roles and user groups. In the following only roles that have a major impact to the whole process are discussed.

The so-called Spec Lead and members of the Java Specification Process (JSP) group provide almost everything for a new specification. Beside the specification they have to provide a Reference Implementation (RI), a Technology Compatibility Kit (TCK) and an adequate license for each of this. A TCK mainly consists of different testing suites to prove if an implementation fulfills the according specification. Feedback and reconsideration ballots from each review round have to be processed by the responsible JSP group.

The Executive Committee (EC) is the main communication partner for groups with a new specification request. The EC takes a first review before the draft of a new specification request is published to the JCP web site for public review.

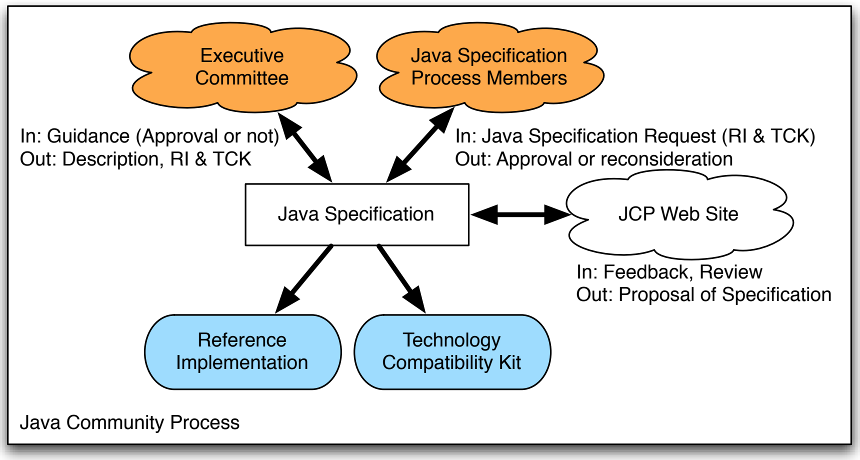
Executive Committee
Java Specification Process Members
In: Guidance (Approval or not)
Out: Description, RI & TCK
In: Java Specification Request (RI & TCK)
Out: Approval or reconsideration
Java Specification
JCP Web Site
In: Feedback, Review
Out: Proposal of Specification
Reference Implementation
Technology Compatibility Kit
Java Community Process

Figure 2: Participation Model with information flow of JCP

The third major group of the JCP is the Internet user group. Each individual can participate through the public web site. An actual browser and Internet access is sufficient to take part in the process.

Management of Development Process

The process for a new Java Specification starts with a Java Specification Request. Within 14 days an Executive Committee has to decide whether the request is approved or not. If the request is not approved it is possible to adopt and file it a second time. Otherwise the Spec Lead has to define an Expert Group that can start working on the Early Draft.

There are no rules for a special working style except that the process should be transparent and use Internet-based technology. Based on the output of this step the Community Review and Early Draft Review take place. The review phase lasts between 30 and 90 days. The difference of these coexisting reviews is the user group. A Community Review addresses Java Community Process Members whereas an Early Draft Review is accessible for everyone. It is also possible for an Expert Group to update a draft during this review process. All members have to be notified about such an update. If the Early Draft Review ends up in an approval the Expert Group will complete the specification for a Public Draft. Such a Public Draft contains a full specification and has again to be reviewed for 30 to 90 days. In the phase before Final Draft Review the Expert Group completes TI and TCK. After the Final Draft Review and approval by the Executive Committee a new specification is established.

Promotion and Handling of Innovation

The maintenance and innovation in a project happens under the lead of an Expert (Maintenance Lead). Without a Maintenance Lead at least no maintenance is carried out and the specification is marked as Dormant. Maintenance Leads collect feedback and ideas via an email feedback address from the public.

The Maintenance Lead decides to initiate a Minor Change or more depending on the relevance of a feedback to specification. Each correction or new feature gets listed on an according section of the specification. Depending on the status or impact of a request it gets marked as Proposed (for not yet implemented), Accepted (for change made), or Deferred (for change items to be considered in a new JSR in the Change Log section). A revision of the specification must be reviewed in a Maintenance Review before it can be released. Any new feature or changes that do not address issues of an existing specification always lead to a new JSR.

2.3 Khronos Group Development Process

The Khronos Group Development Process is the stepping-stone for all standards under the OpenGL brand and many other visual computing standards.

"All Khronos members are able to contribute to the development of Khronos API specifications, are empowered to vote at various stages before public deployment, and are able to accelerate the delivery of their cutting-edge 3D platforms and applications through early access to specification drafts and conformance tests." [3]

Participation Model

The Khronos Group is a non-profit organization with almost every big player of the computer and mobile device industry being a member. A member can take part as Promoter or Contributor. The annual membership fee is twice as high for a Promoter as for a Contributor member. The main difference (except the fee) is a higher degree of influence towards the future direction of working groups and the Khronos Group. The development of a new specification begins with the initialization and ends with the ratification through members of the Promoter group. In a working group both parties have equal rights on marketing, frequently meetings, early access on specification, voting, ratification and the right to chair a group.

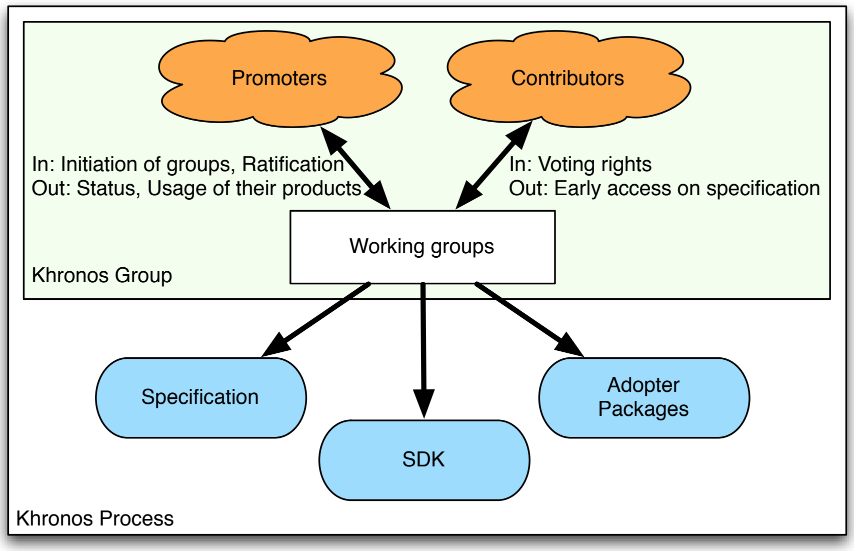

Figure 3: Participation Model with information flow of Khronos Group Development Process

Based on the ratified output of the working groups all other stakeholders can use and implement the standards. Depending on the kind (soft- or hardware) and conformance of an implementation a developer is allowed to use the official trademarks on the product.

Management of Development Process

The Khronos process is rather focused on standardization than on the development itself. Technical development takes place in the working groups. A working group can be initialized through a member of the Promoter group. After the initialization a Working group starts working on its first draft. The actual state of work is transparent and accessible through the Website of each group. A group provides at least a forum to receive feedback from public. Many of them also offer a public wiki or a mailing list to advocate discussion between the group and developers or users. For the coordination inside the Khronos group teleconferences take place every week. Additional face-to-face meetings between a working group and Promoters/Contributors enrich the communication. The release of a specification needs ratification through the promoters and contributors. The ratification period should last at least 30 days but no longer than 60 days.

Promotion and Handling of Innovation

A Promoter mainly influences the direction in a new working group. Based on this direction and an early access to the specification other Promoters or Contributors can enhance or influence the output of a working group. Weekly meetings or teleconferences should support the Promotion of Innovation. The decision if a feature or change request is integrated occurs through a voting of Promoters and Contributors. Each member of this group has exactly one vote. A majority can be reached if more than 50% of the attending groups vote for it. Last but not least a public forum offers a possibility for the public to contribute new ideas or change requests.

3 Partner Programs

As valuable information about open development and especially the establishment of an open development communities with commercial background is rare, a closer look onto partner programs of leading companies might provide a useful input for further research. In the following sections an overview of following three well-established partner programs (and paradigms) is provided:

- Value-added reseller (VAL) and independent software vendor (ISV)
- Object Management Group (CORBA, UML, etc.)
- Certified partner program (Microsoft)

3.1 Value-added Resellers (VAR) and Independent Software Vendors (ISV)

The greatest benefit of a value-added reseller program is the ability to offer a product that covers a wide range of particular industries and multiple markets. With a high coverage it is possible to sell more systems and to reduce the development costs per unit. This also means that customization takes place outside of the kernel and only general functionality is encapsulated in the kernel (see [5]). To open the customization and localization for partners it is favorable or even necessary to provide an integrated development environment, which is relatively easy to use. As different partners can do the verticalization and localization a simple exchange between the partners is highly desirable (e.g. one tool for everything). A separation of these two layers forces an interaction and information exchange between partners.

A company can satisfy a wider range of customers and offer ready to use solutions through value-added partners. The second channel to attract partners and customers are independent software vendors. These ISVs develop and service modules that offer additional functionality to the core system. In case of banking solutions a switch from one provider to another is not usual. So the decision finding for the best offer is mainly depending on the extensibility and the total cost of ownership and not on the initial price. A marketplace with a large selection of additional modules and providers is a big impact factor for the decision.

There are two major reasons why companies prefer to sell directly and without partners. One reason is the low portion on the total earnings (around 15-20%). The second reason is the risk to lose the direct contact to customers and therefore the ability to fulfill their needs.

Open development and a central marketplace offer potential solutions for these two major problems. The low portion on total earnings can be compensated through more sells of the product via value-added partners (see [5]). Furthermore a company earns the annual member fees, revenues from the certification program (e.g. certification expires after a year) and fees of every sold module in the central marketplace.

3.2 Object Management Group (OMG)

A challenge of many software products is the Return on investment (ROI). Basically there are two possible solutions to maximize the ROI of a software product. A common solution is the reduction of development costs through outsourcing in countries with lower wages. A second solution is to expand the lifetime of a product. An expansion of the lifetime requires the ability to adopt the product on changing markets. Therefore it is necessary to model a software product before it is built. This second solution is one major driving force behind the Object Management Group.

The OMG has many software vendors as members as well as end-users. The following explanation for the high amount of end-user members is suggested through the OMG: Our end-user members are the leaders; the companies that drive their IT, instead of letting it drive them. [6]. This statement can provide an important input for the establishment of a community and probably a marketplace. Because this force to drive the future development of a product might have a positive impact in terms of motivation to use and support such a community or marketplace.

OMG lines out following six reasons to join their group as end-user (see [6]):

- Help shape industry standards and vendor products to your company's needs
- Leverage the work and knowledge of the industry's best minds
- Plan, purchase, and implement in front of the curve, instead of behind it
- Keep your competition from jumping ahead of you
- Let your company, and your department, be seen as a leader
- Cover your costs (and more!) with savings

As OMG tries to establish technology-free standards many of these points are applicable to different kinds of industries.

In terms of community there is an interesting fact on the amount of provided input because there is no obligation. Each community member can influence different projects with different efforts. The concentration of industry players and solution providers will usually fill gaps if a member decides to reduce or stop supporting a project (see [6]).

3.3 Classic Certified Partner Program

Classic certified partner programs usually offer different levels of a membership. Those levels are categorized by annual fees, turnover, competence level, and number of certifications or a combination out of these. To allow companies a first look onto the program without paying any fees it is possible to join a partner program as a registered member. Such a membership is free but it is already necessary to provide some personal data and to confirm the official membership agreement.

The main target of a certified partner program is to spread the usage of own products and encourage sells of the program owner. If a partner sells only the products or uses it as fundamental for own products is not matter. Additional support on topics such as sales, marketing as well as technical resources and knowledge bases attacks potential partners. Certifications are needed for partners mainly as unique selling proposition and for customers as an indicator for the proficiency.

Typical promoted benefits of a partner program are (see [8]):

- Build your business and increase profitability
- Reduce costs and increase operational efficiency
- Realize your full business potential

These benefits can be achieved if a partner knows your product and has the ability to configure it. For a partner it is necessary to have clear structured hierarchy of the different membership levels. Again, each of these levels need something unique (e.g. a badge) for the selling proposition. The requirements for a level are typically defined as a combination out of revenue and received certifications. A possible hierarchy of partner levels can be seen as follows (see [8]):

- Registered member (3rd level)
- Certified partner (2nd level)
- Gold certified partner (1st level)

This classical idea of a certified partner program is probably outdated and will be replaced through community oriented partner programs. Microsoft for example tries to establish a network between all partners, the Microsoft Partner Network (MPN) (see [7]).

Each partner has to join the MPN (similar to registered member) and to choose a specific competence program. These specific competence programs probably should address a major issue of the classic partner program. In the classic partner program it was possible to achieve a level through specialization or through diversity.

4 Open Software Development

This section discusses key findings of open software development processes based on the three community-driven development processes introduced in Section 2. In cooperation with a software development company we elaborated the following categorization:

- Governance
- Architecture
- Tooling
- Culture

For the company, these four categories are the most important ones to prepare a switch from closed software development to an open software development.

4.1 Governance

Committers

Eclipse differentiates levels of projects from top-level to sub-level ones. Committers can contribute only to the project they are member of but not to others. Furthermore, existing committers of a project always have to agree on a new committer.

In JCP 2 committers can be divided into two groups: JCP members and non-JCP members. The only way to submit a new specification request is as a JCP member. After the initial review and acknowledgment through the Executive committee everybody is allowed to comment on the Java Specification Request (JSR). The development of a new specification is driven by a group of JCP members. Outside people can only provide comments.

In Khronos, the development of a new specification takes place in working groups initiated through a member of the Promoter group. A second group called Contributors has early access to the specification and can influence the development through voting rights. Outside persons can comment on the (final) work of a working group through the public website.

Project Types

Eclipse: Of the two different types of projects, namely top-level projects and sub-projects, an incubator project is a special kind of a sub-project. Incubator projects are used for the promotion of innovative new projects (in the context of the top-level project) or as a sandbox for major reengineering of an existing project.

JCP 2: The only project type known by JCP 2 is a Java Specification Request (JSR). A JSR is opened for every new specification or for an adoption of an existing specification.

Khronos: The development of new Khronos specification takes place in Working Groups established by the Promoters of the specification. Similar to JCP 2 it is not possible to adopt an existing specification.

Standards

The standardization of Eclipse, JCP 2, and Khronos processes mainly rely on restrictive review processes. In contrast to many other requirements of the development process (e.g. documentation) these reviews must strictly follow the regulation.

Key Findings

Governance is a key enabler for communities to work in an open and business oriented environment. All reviewed community processes choose a similar approach to face the challenge of openness and profit-orientation in the development process.

- Openness through coordinated contributions and decision transparency:
  One might think openness stands for free access and everybody is allowed to contribute. None of the reviewed development processes allows such random and uncoordinated contribution or access to a project. The openness of these processes is given through the possibility that every member can trace decisions and easily raise a comment on an issue. If a member raises a comment he/she can be sure that it is heard because the addressed receiver has to reply on the comment. Such a conversation is normally traceable by any member of the process. Transparency and traceability promote the growth of trust between members of a community.
- Simplicity of project structures:
  Simplicity is another key success factor besides transparency, traceability, and trust. The project structure in community-driven development processes is simple and has a hierarchy with no more than two levels (project and sub-project). Every project should work stand-alone to avoid dependencies to other projects. Dependencies to other projects might stall the innovation in a project. A dependency can be everything that is in more than one project. Even a human can count as dependency. So it is clear that every project should be able to decide who is allowed to contribute to the project. In some places all these stand-alone projects should fit together and build one big solution. The synchronization of multiple projects to one big product should not impact the innovation process of a project.
- Multi-stage, community-driven reviews:
  Community-driven development processes use reviews for the synchronization of the single projects. Reviews take place on multiple stages (e.g. creation, graduation, etc.) of a project. Each of these reviews is fully specified through the development process description.

4.2 Architecture

General

In Eclipse the so-called Architecture Council is responsible for the development, articulation and maintenance of the Eclipse Platform Architecture. This council consists of about 50 members for about 35 mentored top-level projects. The communication and voting take place mainly on a public mailing list. An issue tracker is used for the communication between the council and outside people.

JCP 2 does not have a specific technical architecture such as Eclipse. In JCP 2 it is possible to establish a technical specification that may also contain a specification of the used architecture itself. It is even possible that a JCP request constitutes the definition of architecture such as JSR244 (Java EE 5 specification).

Khronos is similar to JCP 2 in that the technical architecture is defined through specifications.

APIs

The Eclipse project offers a large API to plugin developers. This API is well documented and expects that developers take some usage rules into account. These rules are a kind of extended coding conventions. If a plugin developer obeys the rules the possibility that something might be broken in a new version of the API is minimal (for the detailed rules see [9]).

A JCP 2 specification has to contain a specification of the API and a Technology Compatibility Kit. The Technology Compatibility Kit consists of tests to ensure that an implementation meets the requirements of the implemented specification.

Khronos specifications normally specify a standardized layer between hardware and software products. The final specification is available both as document and implementation in an SDK.

Modularity

Eclipse heavily depends on the Open Services Gateway initiative framework (OSGi) standard as driver for modularization [11]. OSGi is a module system for Java that implements a complete and dynamic component model.

JCP 2 and Khronos do not have a special mechanism to achieve or ensure modularity. The focus of these two processes is on a correctly working implementation with little interest on the implementation. In JCP 2 it is possible to establish a specification to improve modularity, e.g. JSR 294 - Improved Modularity Support in the Java Programming Language.

Key Findings

Well-defined evolution of the architecture by strict compliance: The architecture of a system must be well defined and evolutionary changes should take place in a predictable manner. Every reviewed community-driven development process takes a different approach of how to specify the architecture but one thing is common for every process: a strict compliance test of the public interface. Every API or SDK has to have at least 100% signature test coverage.

- Established API evolution rules:
  One major challenge in software development is the evolution of an API. JCP 2 and Khronos solve the problem of API evaluation with a new specification. This makes sense because it is not possible to adopt every existing product that is based on a specification. The Eclipse process has established a set of API evolution rules. These rules are a kind of coding convention. They describe how to use the API based on the underlying extension strategy of the API. This extension strategy heavily depends on to the modularity of the product.

4.3 Tooling

Knowledge Management and Social Media

Knowledge management in Eclipse, JCP 2 and Khronos currently takes place via mailing lists, forums or Wikis. There are discussions in the communities to find a more structured solution but no concrete approach yet how this problem can be solved.

Testing

Eclipse: The Eclipse Test and Performance Tools Platform (TPTP) Project provides an open platform that supplies powerful frameworks and services for testing. TPTP addresses the entire test and performance lifecycle, from early testing to production application monitoring, including test editing and execution, monitoring, tracing and profiling, and log analysis capabilities. The platform currently supports a broad spectrum of computing systems including embedded, standalone, enterprise, and high-performance and will continue to expand support to encompass the widest possible range of systems. [10]

JCP 2: Testing is an essential part of every JCP 2 specification. Tests of a JCP2 specification have to fulfill 100% signature coverage. For all other tests a specification has to provide specification coverage and how this coverage was achieved (such as techniques, criteria, etc.). Depending on this information it should be possible to evaluate the quality of the tests.

Khronos: Based on the kind of hardware and its certification, the testing process is subject to a specific set of testing criteria (e.g. identical rendering pipeline like some reference product).

Key Findings

Tooling as enabler for a working community: Tooling builds the key platform for an optimal working community. Current IDEs used in community-driven development processes support different views for almost every group of technical users. Designers, developers and testers use the same tool with different views to work on one common project.

- Knowledge management has not adopted Social Media yet:
  However, knowledge management outside the source code is still pretty old fashioned. Most common knowledge management tools used in community-driven processes are mailing lists and bug trackers. Actual scientific research has the focus on social media and social networks as knowledge and information exchange platform coupled to the source code (see [12]).

4.4 Culture

Roles

Next, we provide a brief overview of the roles in different open development communities. Eclipse and JCP are more rigorous in having strategic decision bodies; Khronos works with less structure and is driven by initiatives.

Roles in Eclipse:

- The Council takes management and strategic decisions for the development process
- A Project Management Committee leads a project under the Eclipse development process
- A Project lead that leads a sub-project in the Eclipse development process
- Contributor contributes to the Eclipse platform
- Adopters use the Eclipse platform for their own products
- Users of Eclipse or a product based on Eclipse

Roles in JCP 2:

- The Program Management Office is the responsible group in Oracle for JCP
- An Executive Committee guides the evolution of the Java technology
- An Expert Group develops or makes significant revisions to a specification

- Java Community Process Members take part in the process
- A Maintenance Lead is responsible for maintaining the specification

Roles in Khronos:

- A Promoter initiates Working Groups and ratification
- A Contributor can influence the work in Working Groups
- Working groups devise new specifications
- Adopters build conformant implementations
- Developers program applications using the Khronos API

Transparency

Eclipse: The information exchange in each of the three councils is transparent as it is publicly accessible. Voting and decision finding only takes place inside each of these councils. Therefore the process is transparent and targeted but not influenced from the outside. The transparency of information exchange on levels under the council depends on the kind of project (open/closed, free/commercial, etc.)

JCP 2: Transparency is an essential part of the JCP 2: Expert Groups must operate in a transparent manner, enabling the public to observe their deliberations and to provide feedback. All feedback must be taken into consideration and public responses to such feedback must be provided. [2]

Depending on the degree of transparency even the voting process is influenced: The EC should take the Expert Group's transparency record into consideration when voting on its JSR. [2]

Khronos: Compared to Eclipse and JCP 2, the Khronos process is not really transparent to the public. Only the final specification is available for the public. During the specification creation process information is only available to Promoters and Contributors. Both types of membership require a fee ($30.000 for Promoters and $10.000 for Contributors).

Key Findings

Strategic bodies versus bottom-up working groups: The role models in the three communities vary quite a bit. Eclipse and JCP run top-down by strategic initiatives and the respective bodies (e.g. councils); the Khronos process runs bottom-up with working groups that are initiated. However, the essence is conformance that is the major goal for platform evolution and usage.

- Establishing a culture of transparency and trust:
  For an open and innovation-driven culture it is necessary to strengthen transparency and trust. Firstly, long-term architectural strategies must be available to every member of a community. A second step is the opening of the strategy finding process.
- Strategy finding process open to major partners:
  Depending on the business targets of a company or institution the strategy finding process is open for major partners only or the whole community. Also the amount of a parties influence needs to be derived from the main business target of the leading company behind the community process.

5 Conclusion

In this technical report we provided an overview of open software development processes and commercial partner programs, and key findings for a successful open development of software systems. The opening of software development processes is necessary due to the increasing interlinking of software systems and the indirect need to inform partners and customers about upcoming changes in the software system. This opening allows partners and customers not only to react on upcoming changes, it opens them a new way to have impact on the future development of the software system. For example, a customer needs an implementation of a new data export interface. In a closed development process the software vendor or a partner would implement the extension to the software system. In a open development process the customer can implement the extension himself and supply it to other customers, partners and the software vendor. Open development allows a software vendor to focus on the development of the core system and customers to adopt the software system to their needs.

References

[1] Eclipse Development Process (Last visited: April 2, 2012) `http://www.eclipse.org/projects/dev_process/development_process_2010.php`

[2] Java Community Process 2 (Last visited: April 2, 2012) `http://jcp.org/en/procedures/jcp2`

[3] Khronos Group - Member Agreement (Last visited: April 2, 2012) `http://www.khronos.org/files/member_agreement.pdf`

[4] Khronos Group - Member Levels (Last visited: April 2, 2012) `http://www.khronos.org/members/benefits/`

[5] Michelle Antero, Niels Bjorn-Andersen, A Tale of Two ERP Vendors – and the Crucial Decision of Choosing the Right Business Model, CENTERIS 2011, Part I, CCIS 219, pp. 147-157, 2011

[6] Jon Siegel - Vice President Object Management Group, Why Should My End-User Company Join OMG?, 2010

[7] Microsoft Partner Network, Quick Reference Guide for U.S. Microsoft Dynamics Partners, 2011

[8] Microsoft Partner, The Microsoft Partner Program Guide for all Microsoft Partners, 2005

[9] How to Use the Eclipse API (Last visited: April 11, 2012) `http://www.eclipse.org/articles/article.php?file=Article-API-Use/index.html`

[10] Eclipse Test and Performance Tools Platform Project (Last visited: April 11, 2012) `http://www.eclipse.org/tptp/`

[11] Eclipse and OSGi (Last visited: April 11, 2012) http://www.eclipse.org/osgi/

[12] Andrew Begel, Robert DeLine; Microsoft Research; Codebook: Social Networking over Code; Association for Computing Machinery, Inc.; Proceedings of ICSE 09 (New Ideas and Emerging Results) `http://research.microsoft.com/pubs/81052/codebook-icse2009.pdf`