The VistA Ecosystem:
Current Status and Future Directions

James Herbsleb **Claudia Müller-Birn** **W. Ben Towne**

October, 2010
CMU-ISR-10-124

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Table of Contents

Executive Summary i

1 Introduction 1

2 The VistA ecosystem 2

2.1 Research Methodology 3

2.2 Brief history of VistA 4

2.3 Technology and main distributions 6

2.4 VistA community 8

2.4.1 WorldVistA 9

2.4.2 VISTA Expertise Network 12

2.4.3 VistA Software Alliance 13

2.4.4 Open Health Tools 13

2.4.5 Commercial firms 15

2.5 Perceived Challenges for the VistA Ecosystem 20

2.5.1 Challenges seen by Community members 20

2.5.2 Discussion of Challenges 23

3 Possible Paths for VistA 24

3.1 General observations: unification or fragmentation 25

3.1.1 The key choices are: 26

3.2 General observations: The role of VA 26

3.2.1 The key choices are: 29

3.3 Technical platform architecture 29

3.3.1 The key choices are: 31

3.4 Collaborative infrastructure 31

3.4.1 The key choices are: 33

3.5 Governance 33

3.5.1 The key choices are: 35

4 Conclusion 35

Acknowledgments 37

References 37

Appendix A: Dimensions of platform-based ecosystems 41

Appendix B: Four case studies of established ecosystems 46

Appendix C: Commonalities and variabilities of four open ecosystems 66

Appendix D: Tabular ecosystem comparison 71

Appendix E: Brief overview about licenses 77

Appendix F: VistA discussion groups 78

Appendix G: Red Hat and Fedora 79

Executive Summary

With the recent national focus on health care and the infusion of funds that HITECH will bring to Health IT (HIT), it is hard to imagine a more appropriate time for the U.S. Department of Veterans Affairs (VA) to formulate policy aimed at making its HIT system, VistA, more widely available. In fact, judging by the transformative impact that open platforms have had in many technology domains, it is likely that if done correctly, the VistA software could form the basis of a thriving ecosystem that would drive down cost and unleash innovation. The kernel of a socio-technical ecosystem based on VistA is functioning today outside of VA, but it needs help and support from VA policies if it is to thrive, grow to its full potential, and help to transform HIT.

Socio-technical ecosystems are extraordinarily complex, but research has recently begun to reveal the keys to software platforms that spark successful ecosystems. The primary difficulty is charting the course to growth and en route there is a set of fundamental issues that need to be addressed: What technical work must be done to evolve VistA into a suitable platform for use outside of VA? How should an ecosystem consisting of great numbers of vendors, service providers, IT departments, many classes of users, VA, non-profits, and many other types of players be governed so that coherence can be maintained, while still encouraging innovation? What kind of cultural and technological infrastructure must be deployed in order to support information dissemination and effective decision-making? How can a critical mass of adopters be attracted quickly to create viable business opportunities?

In this technical report, we help to answer such questions by mapping out the VistA ecosystem and comparing it with four well-established, highly successful software ecosystems. We begin by providing the results of a year-long qualitative study of the current VistA ecosystem. For this study we conducted interviews with stakeholders, examined documents, e-mail lists, and other sources, to create an up-to-date view of VistA development and distribution, and the people and organizations outside VA who are involved with it. Second, we performed four smaller-scale case studies of long-lived, thriving ecosystems based on software platforms: Apache, Eclipse, GNOME, and Mozilla. We looked for similarities and differences across these four cases, and then applied what we learned to the current state of VistA. Based on this analysis, we discuss ways of moving VistA forward, capitalizing on lessons from successful platforms. We identify the key choices facing VA and other participants, summarized below.

The architecture of the ecosystem: Our observations of the four established ecosystems lead us to note several obvious differences between the architecture of the VistA ecosystem and all the others. The established ecosystems all have a single main distribution and code that is centrally managed. Licensing is uniform or at least compatible across the ecosystem, and each has a central foundation that owns the intellectual property or the community is kept together by one shared license. Moreover, VA plays a role not present in other ecosystems. In particular, VA accepts very little outside code, and does not release key parts of the VistA system. There are good reasons for both of these behaviors, but they create a powerful damping effect, as we explain.

We believe it is possible for VA to meet its unique needs, reduce costs, and support HIT adoption nationally, and create business opportunities to forward-looking firms by pursuing an open source ecosystem strategy. It will require successful resolution of several issues, outlined below, for evolving the technical architecture, establishing a governance regime, and creating an effective

collaborative infrastructure. From a business perspective, we believe the appropriate incentives can readily be put in place to establish the four key elements of a VA ecosystem-based strategy:

- VA performs or directs the vast majority of platform maintenance and enhancement.
- Vendors will provide small, focused platform enhancements for free.
- Vendors will provide the bulk of application development, enhancement, and maintenance for a free distribution.
- VA will fund application enhancements required by VA, and release the enhanced versions to the ecosystem.

Key choices about the structure of the community and the role of VA are:

- **VA:** Will the VA commit to full participation in the ecosystem, and adopt an ecosystem-based software strategy?
- **Community:** Will the community continue to maintain several distributions or will it rally around a single distribution with uniform or compatible licenses?

The technical characteristics of the platform: VistA is currently not separated neatly into an underlying platform and applications that run on top of it. All of the established ecosystems have such a technical structure, although they differ considerably on how large the platform is relative to the entire distribution, and nature of the Application Programming Interfaces (APIs) through which the platform functionality is exposed. A closely related question concerns what will be included in a free distribution. In all the established ecosystems, a set of applications sufficient for a fairly wide range of uses is included, helping to spur adoption and grow market share. Including too much can reduce business opportunities, such as the ability to sell the included functionality. Including too little suppresses adoption and growth of the market. Key choices about the technical characteristics of the platform are**:**

- Will a platform be "cored" from VistA? What will it include?
- What APIs will be defined and developed? What functionality will be exposed for each? In what specific way will the API make the functionality available?
- Will free, open versions of applications be included in a distribution of the platform? If so, which ones? With what level of functionality (will it, for example, qualify for "meaningful use")?
- Will VA choose contractual arrangements with vendors that will make all useful platform code available to the community, or will vendor code continue to be regarded as proprietary?

Technical and social collaborative infrastructure: In each of the established ecosystems, the community converged on a common set of tools to support collaboration, communication, and coordination. They also developed cultures with norms and values about cooperation, transparency, and citizenship. The VistA ecosystem has the potential to be much larger and more interconnected than any of the established ecosystems, so having an effective set of tools and a culture that strongly encourages prosocial behavior will be particularly important. Key choices about the technical and social collaborative infrastructure are**:**

- Will all participants agree on and use a common set of tools, including a hosting service, mailing lists, source control, change management, etc.?
- Will all participants (including VA and participating firms) agree to openly discuss important technical decisions in advance in agreed-upon public forums?

- Will all participants agree to take on the values and follow the norms appropriate for open source, even when this causes some friction with organizational values and norms?

Governance structures: Finally, we note that all the established ecosystems had governance structures built around foundations. A number of key decisions were made centrally, by the foundation, often in consultation with members or advisory boards. Other decisions, such as what products to build, services to offer, and business models to pursue, were left completely up to participants. There were various levels of decision-making, with some showing a strong "meso" level, with projects having considerable authority over their own schedules, processes, and what features to build. There were also a variety of participatory mechanisms such as councils and committees that allowed participants to have a role in making decisions. The established ecosystems differed with respect to the status of firms – some had firms as members, others only recognized individuals. Key choices about ecosystem governance are**:**

- Will a foundation be recognized as the legitimate steward of the ecosystem? Will it have ownership of the intellectual property, or be granted the power to insist on compatible licenses?
- What kind of input will VA and the community have in the technical direction and evolution of the platform?
- Will an organizational structure include a meso-level with independent development projects, or will a foundation be the main decision-making entity?
- What will be the scope of the foundation's decision-making, and what kinds of decisions will be left to Councils or members?
- How would a foundation be funded?
- Will companies participate, for example, as members, board members, council or committee members in such a foundation? If yes, how?

We wish to call particular attention to the process by which decisions are made in the established ecosystems we studied. In each case, the communities are built on open debate, meritocracy, and decision-making based on consensus. They are by no means free of conflict, but they resist imposition of solutions from the outside. The nascent VistA ecosystem has a similar culture, and would be seriously damaged by decisions simply imposed upon it. We believe that making decisions the right way – in an open community-based process – is at least as important as making decisions that are correct on their merits. Incorrect decisions can be changed if the community remains intact. Community-building may be the most important activity to be undertaken.

A critical role for research. Enhancing and growing the VistA ecosystem will push the boundaries of practical experience and the current state of knowledge. For this reason it is critically important for ecosystem development to proceed in close collaboration with a research community committed to addressing the essential unknowns, anticipating issues, applying state of the art solutions, and taking on key issues that will help the ecosystem thrive over the longer term. Among the central issues research must address are the following:

- What social and technical collaborative infrastructure – tools, practices, and norms – will support an ecosystem on the scale and with the interconnectedness that VistA could achieve?
- What delivery models make sense for VistA – cloud deployment? Providing applications as a service? Providing the platform as a service?
- How can the very large-scale VistA legacy system be evolved into a desired architecture?

- For individual and corporate participants, what is the business case for contributing to the creation and maintenance of a platform and a fully-functional free distribution? How can it be modeled in convincing fashion so participants will have a sound basis for their decisions about participation and contribution?
- How can the critical performance, security, availability, and other quality attributes of deployed systems be assured?
- How can open source governance mechanisms – foundations, projects, councils, committees, user groups, vendors, service providers, health professionals, standards bodies – be orchestrated to provide effective policies, dispute resolution, and guidance for an ecosystem of unprecedented scale and complexity? Will novel mechanisms be needed?
- In order to guide overall decision-making, how can the socio-technical ecosystem be modeled so that the effects of establishing or changing various design parameters can be predicted, and the space of parameter combinations explored?

Research on these issues, critical for near- and long-term success, should be adequately funded, and mechanisms provided for frequent interaction between the research community and thought leaders in the ecosystem.

VistA approaches a critical juncture. VistA might well suffer the fate of Unix, and continue to fragment into multiple and somewhat incompatible versions, each with its own small community. This is not such a terrible fate – except by comparison to what VistA could become. Imagine a free platform and basic applications that could be used by doctors' offices, clinics, small hospitals, and medical centers. This platform is highly reliable, secure, and standard-compliant, and is naturally adopted by a substantial segment of the HIT market. Service providers of various sizes and specialties stand ready to install, configure, customize, and train. Vendors supply upmarket versions of the applications to large hospitals and hospital networks. All the participants, including VA, benefit from the huge market created by adoption of the free version, and by sharing the cost of maintaining the platform and basic applications. HIT costs drop, since duplication of effort is avoided, and each firm can focus on its differentiating competencies. Innovative companies, large or small, can sell their novel products and services with low barriers to entry because of the openness of the ecosystem. Such an ecosystem would not meet all HIT needs, and proprietary solutions would thrive alongside the VistA-based ones, just as now happens with Apache, Eclipse, GNOME, and Mozilla. Yet VistA would be at the center of the HIT agenda, driving down costs through competitive pressure, unleashing innovation, providing business opportunities for firms of all sizes, and making solutions available to those who otherwise could not afford them.

1 Introduction

We have arrived at a critical point for Health Information Technology (HIT) in the United States. The convergence of several key events presents us with a rare opportunity. The U.S. Department of Veteran's Affairs (VA) is currently considering policy options for encouraging wider use and adoption of the powerful VistA HIT system, currently in use in VA facilities around the country, and adopted by a number of hospitals and clinics around the world. As claimed by VA CIO Roger Baker (Baker, 2010), it is arguably the finest HIT system in the world. A working group of the Industry Advisory Council (IAC), commissioned by VA to look at ways of evolving VistA and deploying it to a wider community, has recommended pursuing an open source strategy (VistA Modernization Working Group, 2010). We believe an open source strategy built around a VistA platform could have a transformative impact on cost, adoption, and innovation in HIT (Yellowlees, Marks, Hogarth, & Turner, 2008), as this approach has already had in information technology areas such as web and application servers (e.g., Apache, MySql, and Perl), operating systems (e.g., Linux), mobile technologies (e.g., app stores for platforms such as Android), and development environments (e.g., Eclipse).

Platforms and the ecosystems that grow up around them have the power to *"drive innovation and transform industries"* (Evans, Hagiu, & Schmalensee, 2006). They form the context for both cooperation and competition, in ways that potentially combine the best of both (Brandenberger, & Nalebuff, 1996). Open platforms can orchestrate people, technology, and organizations into socio-technical ecosystems that remain, even on large scale (Northrop, 2006), flexible and innovative. For ultra-large, highly interconnected systems, such as HIT, socio-technical ecosystems are the only mechanism that will allow sufficient resources to be brought together in a coordinated way to produce the systems we need (Northrop, 2006).

The time for taking bold steps in HIT could not be better. The Affordable Care Act of 2010 has focused national attention on health care, and focused a spotlight on the serious, health-threatening shortcomings of our current information technology and the promise of technologies that are now within reach (Chaudhry et al., 2006; Hillestad et al., 2005; Longman, 2010). At the same time, the Health Information Technology for Economic and Clinical Health (HITECH) Act will provide sufficient funds to have a very large impact on rapid adoption and deployment of HIT systems.

Yet growing a healthy ecosystem is fraught with uncertainties. Success clearly depends on much more than the quality and the design of the software itself. How to create and sustain a critical mass of adopters that will provide a viable market? How to design the platform so that new technologies can be quickly integrated? How to ensure system-level security and regulatory compliance when multiple vendors operate independently? How can such an ecosystem be governed so that technical coordination and coherence are assured, yet novelty and innovation are encouraged? Why would firms, naturally seeking profit and avoiding uncertainty, adopt business models that are tied to the success of an overarching ecosystem over which they have limited control? These are but a few of the very difficult questions that must be addressed in order to design a viable ecosystem. Design principles to guide such decisions are uncertain at best, and lack scientific basis.

Yet there exist a number of long-lived, robust ecosystems generating products that are highly competitive, or, as in the case of Apache (see Netcraft, 2010) even dominate their markets. We believe that careful study of such ecosystems can provide guidance for anyone wishing to grow the network of users, vendors, and service providers that surround VistA, and can also begin to provide more general design principles and move us toward a more systematic understanding of ecosystems.

The goals of this report are twofold. We have the *pragmatic* goal of providing concrete advice, grounded in empirical observation and analysis, to help guide the formation of a viable VistA-based ecosystem. We have the *scientific* goal of identifying design dimensions, begin to formulate design principles, and identify choices in ecosystem design. Toward these ends, we have conducted an extensive qualitative study of the VistA ecosystem, and used a comparative case study of four highly successful open source ecosystems in order to suggest directions VistA could take to grow and thrive beyond VA.

The remainder of this report is structured as follows. Section 2 presents the results of our qualitative case study of the existing VistA ecosystem. Section 3 compares the VistA ecosystem to four established ecosystems that are long-lived, robust, and thriving. They represent plausible, albeit smaller scale, models for what the VistA ecosystem could become. In order to provide this comparison we draw on research that has recently begun to identify some of the critical decisions for the design of ecosystems. Gawer and colleagues (Gawer, 2010; Gawer & Cusumano, 2002) have identified four "levers," i.e., design parameters that firms striving for platform leadership, including the scope of the firm, product technology, external relationships, and internal organization. We adopt an analytic framework consisting of four dimensions that are critical to ecosystem design: the technical architecture, governance regime, collaborative infrastructure, and business opportunities.[1] At the heart of any ecosystem is a technical platform that supports the accumulation of the efforts of diverse contributors (Evans et al., 2006; Gawer, 2009; Gawer et al., 2002). A governance regime is the way in which a particular ecosystem distributes types of decisions over types of participants and decision-making mechanisms. Successful collaboration over distances and across organizational boundaries is widely recognized as a challenging problem (e.g., Olson & Olson, 2000), but established ecosystems have addressed this with a combination of a culture favoring openness and a collection of internet-based tools that foster communication and coordination around the product (Moon & Sproull, 2000). The final dimension we use to compare the ecosystems is the set of business opportunities that are actively being pursued in the ecosystem (see generally Messerschmitt & Szyperski, 2005).

2 The VistA ecosystem

In this section, we present the results of our empirical case study of the VistA ecosystem. We first describe our research methods, then give a brief history of VistA in order to provide context. We then present our results in several sections. First, we provide a technical overview about the VistA system, describing the main distributions. We then describe the VistA community and how it is organized. Finally, we discuss challenges for the VistA ecosystem that were identified by our interviewees.

[1] Various appendices contain the details of these dimensions along which we compared ecosystems (Appendix A), our four case studies of established ecosystems (Appendix B), our analysis of commonalities and variabilities among the cases (Appendix C), a table with our detailed point by point comparison of all the ecosystems considered (Appendix D), a brief overview of open source licenses (Appendix E), a comparison of VistA discussion groups (Appendix F), and a description of the role of Fedora in the Red Hat Linux ecosystem (Appendix G).

2.1 Research methodology

Our study of the VistA ecosystem adopts a grounded theory approach (Glaser, & Strauss, 1967). The goal is to provide insights about the functioning of the VistA ecosystem after collecting data from primary and secondary data sources and developing a rich description. Over a six-month period of time beginning in September 2009, data were collected through individual interviews, analysis of documents (e.g., research articles, news, websites) and participant observations during meetings. The first steps in our process of field data collection were identifying the existing key players in the ecosystem and becoming familiar with the role of information technology in the health care industry. Based on the findings of the document analysis, the first interviews were conducted and additional documentation was identified. The collected information yielded further interview participants.

By means of this iterative process, we identified and interviewed a total of 24 informants. All interviewees represented different stakeholders in the ecosystem. At the beginning, the interviews were semi-structured and the questions were dependent on the particular interviewee. Later in the data collection phase, the interviews became much more structured, especially the final interviews that sought to address specific holes in our understanding. The interviews were between 30 and 90 minutes long and included the following topics: (1) organizational issues regarding VistA development process, (2) technical issues regarding VistA distributions, (3) legal issues regarding VistA distributions, (4) organizational issues regarding the VistA ecosystem, and (5) the future of the VistA ecosystem.

Classification criteria	Number
Number of participants	
Total interviewees	21
Total interviews	24
Medium used	
Telephone interviews	9
In-Person interviews	15
*Affiliation**	
Software vendor	11
Non-profit organization (501(c)(3), (6))	8
Department of Veterans Affairs	4
*Roles**	
Developer	7
Implementer	8
Vendor	11

Table 2-1: Summary of interview participants. *Interviewees could have multiple roles and affiliations.

In Table 2-1 represents an overview of the participants. Interviewees are classified according to three different criteria: medium of interview, affiliation, and role. Most interviews were carried out in-person and were held mainly during the VistA Community Meeting in January 2010. Three groups of affiliations could be differentiated: software vendors, non-profit organizations, and Department of Veterans Affairs (VA). This study deals with the open source ecosystem that exists outside VA; therefore, the majority of interviews were not conducted with VA employees. There are some multiply affiliated participants; for example, a person who is involved in a non-profit organization might also work for VA. The third classification criterion is the assignment of interviewees to one of three main roles: developer, implementer, or vendor. A developer is a person who actively programs functions for extensions to the VistA system. An implementer is a person who installs the VistA system—funded or unfunded—and adapts the system, if necessary. A vendor is employed by a company that installs, customizes, or sells a VistA distribution.

In order to develop a description of the VistA ecosystem, the interview data were inductively analyzed. The interviews were transcribed and then categorized by assigning codes. These codes were based mainly on existing stakeholders and topics. In the next step, the different codes were clustered and the resulting concepts were used to identify specific themes. The latter step was applied specifically for concepts that address existing challenges in the ecosystem. All insights and findings are summarized and documented.

Based on these results, as well as our analyses of four open source ecosystems reported in Appendix B. We developed a framework (see Appendix A) that we used to describe and evaluate the VistA ecosystem, to derive recommendations for improvement, and to highlight existing similarities and dissimilarities to several thriving, well-established ecosystems (see Appendices C and D for a detailed comparison).

2.2 Brief history of VistA[2]

The first ancestor of today's system was created in 1977: VA MUMPS (Massachusetts General Hospital Utility Multi-Programming System) medical system[3] (cf. Figure 2-1). More and more VA medical centers acquired their own computing systems and VA MUMPS was increasingly adopted and enhanced (Brown, Lincoln, Groen, & Kolodner, 2003, p.137). Implemented improvements or extensions of the system had been shared between Medical Centers; however every VA MUMPS system was customized for the specific needs of each site.

In 1979, the further development of the VA MUMPS system was officially terminated. But a group of volunteers, who still believed in the capabilities of the system, continued its development. This group named itself the Hardhats[4]. Over a period of three years, they worked in the "underground" as a geographically distributed development team. This development process was a precursor to the open source development style (Trotter, 2007). However, the system was increasingly utilized within and outside the US. In 1981, during the Symposium on Computer Applications in Medical Care, VA MUMPS with its different components was presented to VA physicians and administrators from around the country. Because of the outstanding capabilities of the system, in February 1982, the VA officially legitimized the MUMPS work and founded a task group on the DHCP (Decentralized Hospital Computer Program). In the first years, the software had been developed in a user-centric manner that meant that its development was closely related to clinical processes and attuned to the specific needs of the physicians. Various independent, regional development centers existed across the US. In 1996, because of an ongoing system evolution, the Department of Veterans Affairs (VA) renamed DHCP as the Veterans Health Information Systems and Technology Architecture (VistA) (Brown et al., 2003). One major improvement was the introduction of a graphical user interface CPRS (Computerized Patient Record System). In the following years, VistA has been continuously extended and improved, for example the Barcode Medication Administration (BCMA) to electronically manage medications, and Clinical Reminders for chronic disease management.[5]

[2] For a history of VistA focusing on its role in the extraordinarily effective healthcare at VA, see generally Longman (2010).

[3] In the late 1960s, MUMPS was created as an operating system and programming language intended for and designed to build database applications for the health care industry. More information about M can be found at: http://en.wikipedia.org/wiki/MUMPS.

[4] More information can be found at http://www.hardhats.org.

[5] We are grateful to DSS, Inc. for pointing this out.

Nowadays, VA provides care to approximately 5.3 million patients (among 7.7 million enrollees or "covered lives") through 1,400 sites (including 171 medical centers and hospitals, 876 outpatient clinics, and a variety of other long-term, community mental health, and home care programs (Veterans Health Administration, 2006).[6]

VistA does not only exist in VA. Internationally, VistA is the HIT system, for example, at the Berlin Heart Institute in Germany and the Nasser Institute Hospital in Egypt[7]. The Department of Defense (DoD) installed their variant of VistA, the CHCS (Composite Health Care System) in military hospitals, and the Indian Health Service (IHS) called their variant RPMS (Resource and Patient Management System). Both variants can be seen as dialects of VistA. RPMS and VistA are most closely related because VA and IHS have repeatedly exchanged software code over the years.

In the last few years, the nature of software development in VA has profoundly changed. Today, the Office for Enterprise Development (OED) coordinates all development activities. In the OED, VistA has been in maintenance mode with a skeleton staff for the last 5 years while about 700 developers are working on a rewrite of the original system. New requirements are identified by NSR (new software requests) in VA. In order to avoid duplication of work, all change requests are collected by the OED and prioritized. Implemented changes (patches) to the VistA system are aggregated to releases and distributed in VA twice a year or on an as-needed basis. A new version of VistA is slated for a first release in 2018.

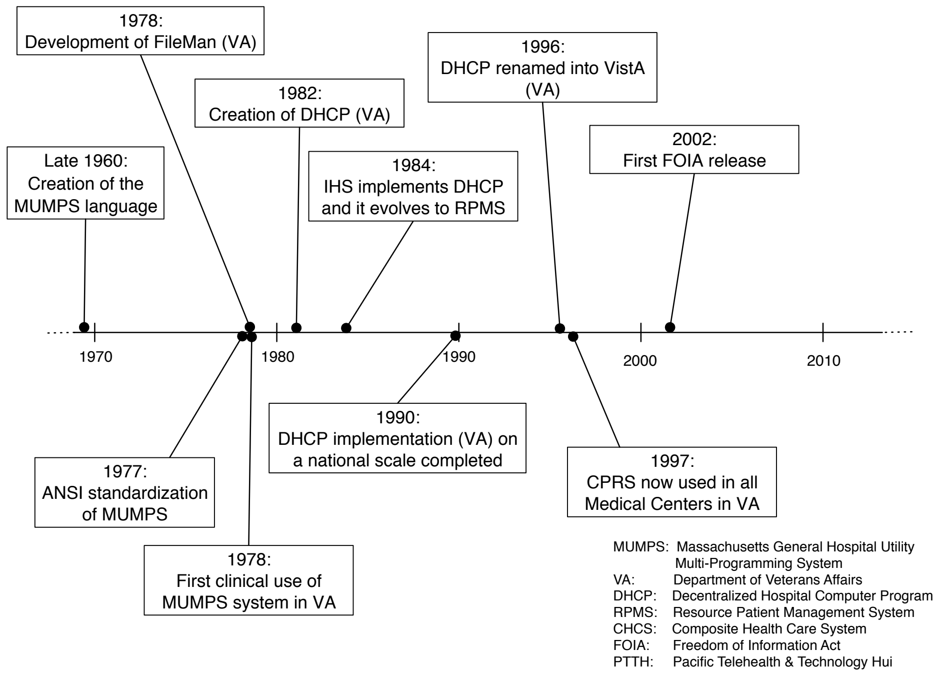

1978:
Development of FileMan (VA)
1982:
Creation of DHCP (VA)
1996:
DHCP renamed into VistA (VA)
2002:
First FOIA release
Late 1960:
Creation of the MUMPS language
1984:
IHS implements DHCP and it evolves to RPMS
1970
1980
1990
2000
2010
1977:
ANSI standardization of MUMPS
1990:
DHCP implementation (VA) on a national scale completed
1997:
CPRS now used in all Medical Centers in VA
1978:
First clinical use of MUMPS system in VA
MUMPS: Massachusetts General Hospital Utility Multi-Programming System
VA: Department of Veterans Affairs
DHCP: Decentralized Hospital Computer Program
RPMS: Resource Patient Management System
CHCS: Composite Health Care System
FOIA: Freedom of Information Act
PTTH: Pacific Telehealth & Technology Hui

Figure 2-1: Timeline of the development of VistA in VA.

[6] Numbers are given by Ben Mehling on Google Group Hardhats (http://groups.google.com/group/hardhats/msg/1d22c0ad3dce7d09).

[7] Information is given by Chris Richardson on Google Group Hardhats (http://groups.google.com/group/hardhats/msg/80adfa1a90f7da3d). Community hospitals that have installed the system are listed on the Hardhats website (http://www.hardhats.org/adopters/vista_adopters.html).

In October 2009, the Department of Veterans Affairs commissioned the Industry Advisory Council (IAC) to invite experienced healthcare and information technology professionals from IAC membership organizations to form the VistA working group (Blankenhorn, 2009; Brewin, 2009). Possible deployment models for VistA, such as open source, as well as strategies for modernizing VistA were discussed in this working group. In May 2010, the IAC working group presented its recommendations to VA (VistA Modernization Working Group, 2010) consisting of two main ideas: (a) replace VistA by a new VistA 2.0 core system that is realized by an open source software development model, and (b) build a sustainable ecosystem around VistA that is governed by a non-profit foundation.

VistA already forms the technical core of an open source ecosystem, albeit one that is far smaller and more fragmented than one might expect, based on the value of the technology. The software is made available via the Freedom of Information Act (FOIA) of 1966, which requires disclosure of governmental documents and information. VistA and unlimited ongoing updates (500-600 patches per year) are provided as public domain software[8] by the US government. Certain parts of VistA are precluded from publication because they fall into existing exemptions; for example, records that are closely related to VHA's internal rules and practices, as well as proprietary code, security related routines, or financial information are not released. However, each year VA publishes existing changes on its website. Patches are assembled in so-called KIDS builds (Kernel Installation and Distribution System). Most of these patches are written to enhance the software, sometimes also to fix existing software errors.

2.3 Technology and main distributions

VistA is not a single application; it is more a conglomerate of various packages and programs. According to the VistA-HealtheVet Monograph, VistA includes 120 applications (Enterprise Development, 2009). A common framework for VistA implementations is utilized in VA allowing third party health care providers to easily access health information (Veterans Health Administration, 2006).

The underlying technology of most of VistA's applications is MUMPS. MUMPS is both a procedural programming language and a hierarchical or multidimensional key-value database[9]. Since 1977, MUMPS has been an ANSI standard which has primarily fostered its utilization in health care information technology. During 1966 and 1967, MUMPS was developed at Massachusetts General Hospital (MGH) in Boston, and apart from health care it is also used in the financial sector. The design of MUMPS is primarily aligned to multi-user database-driven applications. Compared to widely used programming languages today, MUMPS has a very different syntax and terminology. For example, MUMPS has a single data type of string and it is not possible to freely use white space. Global variables are used for persistent storage and are accessible across all processes in the system. Almost every MUMPS command can be abbreviated to one single letter, which makes it possible to have succinct code, but at the same time reduces the readability of the code for inexperienced programmers[10]. In Figure 2-2 and Figure 2-3, a simple Hello World example is given in MUMPS, in MUMPS with abbreviated commands, and in Java.

[8] "Public domain" is not a license; it is more the absence of any license or restriction. The exact rules for public domain depend on the country. More information is available here: http://www.copyright.cornell.edu/resources/publicdomain.cfm.

[9] More information can be found at http://www.mcenter.com/mtrc/whatism.html.

[10] More information can be found at Vistapedia http://vistapedia.net/index.php?title=What_is_VistA_Really.

```
hello()
      write "Hello,␣World!",!
      quit

hello() w "Hello,␣World!",! q
```

Figure 2-2: Hello World example in MUMPS (line 1-3) and MUMPS with abbreviated commands (line 5).

```java
class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello␣World!"); // Display the string.
   }
}
```

Figure 2-3: Hello World example in Java.

Because MUMPS is a programming language and incorporates, based on its design, a database file system, all database interactions are part of the language. In VA, the proprietary object database management system Caché is employed[11]. In opposition to common relational databases such as MySql, MUMPS has a native hierarchical storage structure. The VistA systems use a VA constructed database called File Manager that has properties of both relational and hierarchical databases. MUMPS has object-oriented capabilities via Caché ObjectScript or ESI objects in order to allow for object-based programming while Caché also allows creation of SQL tables and supports SQL code to access the database[12].

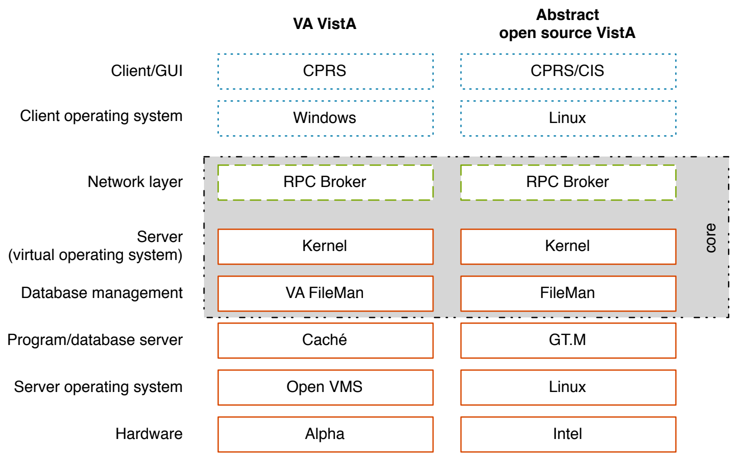


Figure 2-4: VA VistA and abstracted open source VistA technology stack (following (Mehling, 2008)).

The core infrastructure of VistA consists of two main components: VA FileMan and Kernel (cf. Figure 2-4). VA FileMan is a database management system that organizes the medical data, storing it in fields, records, and files (Kolodner, & Douglas, 1997). The Kernel provides the portability layer on top of the operating system, as well as system management tools and shared services such as

[11] More information about this proprietary version of MUMPS can be found at http://www.intersystems.com/cache/. A single user version of Caché can be downloaded from the website.
[12] There is an open source equivalent to Caché's SQL capability, although it is not as fully featured. More information can be found at: https://medsphere.org/community/project/fm-projection.

sign-on and security service, menu management, and error processing. This infrastructure enables applications to be integrated on a database level and all data between applications can be consistently shared. This common infrastructure of the core code can be updated more easily because the layer between applications and the operating system is stable (Brown et al., 2003).

VistA was originally mainframe software that allowed users to interact with the system via character-based terminals. But since the mid-90s, the Computerized Patient Records System (CPRS), which is written in Delphi, is the primary graphical user interface for much of the clinical data in VistA (Veterans Health Administration, 2006). It uses a client-server architecture, with a client user interface in Windows style[13]. However, many departmental systems (e.g., Pharmacy, Laboratory, Radiology) still use a terminal-based user interface.[14]

In 2003, under a contract from the Pacific Telehealth & Technology Hui[15], VA and DoD, in a joint effort, converted FOIA VistA into Hui OpenVista. The main motivation behind this project was the need for an affordable HIT system in the Pacific region. VA VistA runs on proprietary software and the licensing cost turned out to be a major issue for this region. Therefore, Hui OpenVista operates on an open source database management system GT.M[16], a free equivalent of Caché that runs on Linux. Hui OpenVista was the first available open source stack of FOIA VistA. A team called the Hui 7[17] realized this ambitious project; it can be seen as a starting point for the development of today's open source VistA ecosystem.

There are four main distributions of VistA outside VA: FOIA Vista (unlicensed, public domain software), WorldVista EHR (GPL-licensed), OpenVista (AGPL-licensed) and vxVista (EPL-licensed)[18]. These distributions, i.e. flavors, are described in more detail in the following sections (see also Noll, 2009).

2.4 VistA community

In this section, we describe the different stakeholders of the VistA ecosystem and their governance models. The stakeholders can be divided into two groups: organizations and software vendors.

System name	Vendor/implementer	# Patient records	# Sites
Non-government:			
WorldVistA EHR/VOE 1.0	Sequence Managers, etc.	293,195	16
RPMS EHR 1.1	Community Health Network of WV[19]	242,816	30
OpenVista 1.5	Medsphere Systems Corp.	213,948	18
vxVistA	DSS Inc.	189,106	4
Sub-total non-federal		939,065	68
Government:			
VistA	Department of Veterans Affairs	23,442,000	1,007

[13] A demo of this software can be found at http://www1.va.gov/cprsdemo/.
[14] We are grateful to Ben Mehling for pointing this out.
[15] More information can be found at http://www.pacifichui.org.
[16] More information can be found at http://fisglobal.com/Products/TechnologyPlatforms/GTM/index.htm.
[17] Team members in alphabetical order: Dee Knapp, Brian Lord, Rick Marshall, Chris Richardson, Steve Shreeve, George Welch, and David Whitten.
[18] A brief description of these licenses is given in Appendix E.
[19] The initial implementation is done in collaboration with Medsphere Systems Corp.

RPMS EHR 1.1	Indian Health Service	≈ 1,000,000	600
Sub-total government		24,442,000	1607
Total		**25,381,065**	**1,675**

Table 2-2: OSS deployments and aggregate numbers of patient records in deployed system (after Valdes, 2008).

The first group refers to non-profit corporations or two types of associations: 501(c)(3) and 501(c)(6). This group comprises WorldVistA, VISTA Expertise Network, VistA Software Alliance and Open Health Tools. The second group comprises all existing companies. Because of their prominent roles in the VistA ecosystem, two companies are described in more in detail.

Table 2-2 provides an overview of existing applications and their approximate number of implementations. One may see that governmental application of VistA and related distributions exceed by far the applications outside the governmental sphere.

2.4.1 WorldVistA

WorldVistA[20] is a non-profit organization (501(c)(3)), incorporated on March 18, 2002. It is dedicated to further development of VistA as open source health care software (WorldVistA, 2009):

"The primary purposes of this corporation shall be to further the cause of affordable health care information technology worldwide."

A key part of WorldVistA's purpose is to ensure that VistA will be available for the world even if VA eventually stops using it. Therefore, WorldVistA has extended the work of the Hardhats (cf. Section 4.1). The primary goal of the Hardhats has been fostering the virtual community of VistA users worldwide. For example, they offered help with installing and using FOIA VistA. But the Hardhats had no formal, legal existence. Necessary activities, such as applying for grants, contracting, or formal alliances, could not be carried out. Establishing WorldVistA closed this gap. The main goal was to ensure the availability of VistA code outside VA and to provide "*a neutral meeting place for VA programmers, VA bureaucrats, and those external to the VA.*" (Trotter, 2007). WorldVistA is independent from commercial interests, federal governmental interests, and foreign-government interests. During its first years, WorldVistA had been a more technically focused group; today in its self-conception, WorldVistA aspires to be an umbrella organization for all members of the community.

In October 2009, WorldVistA coordinates with 75 implementations outside VA. Its goal is to be an international organization (there are board members from England, Canada, and Jordan). At the beginning of 2010, an important adopter is the country of Jordan, which is implementing WorldVistA EHR, and there are interested parties in India as well.

Governance. The general organizational structure of WorldVistA consists of the Board of Directors, the Executive Team, and regular membership. A member can be any individual but not a company. The Board itself elects the Board of Directors. It is responsible for the management and allocation of assets, the overall strategic direction and priorities, and the delegation of decisions to the Executive Team. The Executive Team develops and implements WorldVistA's operational plan, as well as establishing working groups that are an extension of the management team.

[20] More information can be found at http://worldvista.org.

According to the bylaws, WorldVistA is not a membership organization; this means that non-elected members have no official role in participating in the decision-making process.[21] The main reasons for this model are the number of members of WorldVistA and the size of the community. It has been pointed out that a membership-based model might endanger the vision of WorldVistA's founders. A membership model can be an option with a larger and stronger membership base. However, two Advisory Councils give voice to WorldVistA's members: the User Advisory Council and the Vendor Advisory Council. Councils are self-organized within the terms of reference provided by the Board. During all Board meetings, representatives of both Councils are invited to attend. An overview of the organizational structure is given in Figure 2-5.

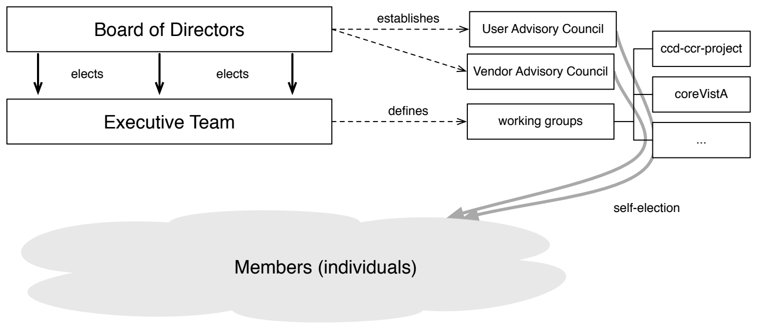


Figure 2-5: Organizational structure of WorldVistA Foundation.

Responsibilities. WorldVistA activities can be divided roughly into three main areas: community development, software development, and business development.

The community development addresses the development of a community of VistA users, supporters, and developers. It involves the organization of the VistA Community Meeting, the WorldVistA Education Seminars, developer meetings, and micro training. The VistA Community Meeting takes place twice a year at various places around the US where low or no cost hosts can be found (Robert Morris University, University of Washington, University of Arizona, the National Library of Medicine, Midland Memorial Hospital, Hewlett Packard, and Intersystems).

While the VistA Community Meeting serves as gathering point for the whole open source VistA community, the WorldVistA Education Seminar is aligned to the specific needs of the WorldVistA community. The types of technical trainings such as in MUMPS or EsiObjects training at the expert level for WorldVistA EHR reflect this. Attendees of this seminar are end users, potential users, or adaptors. Developer meetings, i.e. coding camps, comprise small, intensive programming sessions.

With the community-driven wiki VistApedia.org[22], information (installation, configuration, and usage) about existing distributions (i.e. varieties or flavors) of FOIA VistA, Astronaut (cf. Section 2.4.5), and RPMS EHR is provided (cf. Section 2.2). It is furthermore aimed to give an overview

[21] Regarding the role of community participation in WorldVistA, contrary opinions exist. This issue is more deeply discussed in Section 2.5.1.
[22] More information is available at http://www.VistApedia.org. (This website is owned by one member of WorldVistA's Executive Team.)

about the comprehensive VistA Documentation Library. Moreover, WorldVistA provides information and documentation for the whole community, such as CPT codes[23] and Lexicon files.

Software development is mainly focused on the integration of VA patches into the WorldVistA EHR systems and the development of new functionality. The latter is carried out either by WorldVistA itself or by collaborative development. Even though WorldVistA initiates those projects, the lack of funding often leads to voluntary work done by community members. However, WorldVistA provides the organizational frame to enable collaborative projects. One successful example of these efforts is the CCR/CCD project. This project addresses new standards to represent patient information from different sources in one unit document. The ASTM Continuity of Care Record (CCR) is a standard to transfer patient-related health care information between different medical applications. The Continuity of Care Document (CCD) standard also supports exchange of clinical documents, but it was proposed by HL7[24] in order to harmonize the CCR and the HL7 Clinical Document Architecture (CDA). In 2006, a proposal to add this functionality to WorldVistA EHR led to formation of a working group, which subsequently has implemented it (e.g., for the Electronic Primary Care Research Network, ePCRN).

Part of WorldVistA's business development effort is providing appropriate support for vendors, depending on their level of knowledge. For new vendors, WorldVistA recommends the participation in the Education Seminars or the Community Meeting. For established vendors, support is mainly carried out by *"bringing together others with expertise to help them"*. As WorldVistA is a non-profit organization it is not involved in any business activities. For that reason, the VistA Software Alliance (cf. Section 2.4.3) has been formed.

WorldVistA EHR. WorldVistA EHR[25] consists of GNU Linux, GT.M, FOIA VistA, and EsiObjects[26]. Besides the pure open source stack, WorldVistA can be used with Caché and MS Windows. However, it is the result of two-year effort funded by the Center for Medicare and Medicaid Services (CMS) to provide an affordable EHR that could help physicians improve their patient care. WorldVistA EHR was the first open source software product that received CCHIT certification[27].

Another important technical task of WorldVistA is the consolidation of VA single patches, i.e. KIDS builds, into multi-builds in order to enhance the existing distribution. Such multi-builds can comprise up to 600 KIDS builds and can be used to update WorldVistA EHR. Having single patches eases the identification of existing problems that can occur during their integration. Projects that exceed these activities, i.e. larger software development projects, are dependent on available funding.

Approximately 99 percent of all the WorldVistA EHR code is based on FOIA VistA. But only about one percent of the source code that has been developed by the open source community goes back into VA (quotation from WorldVista Education Seminar 10/2009).

[23] CPT (Current Procedural Terminology) codes are numbers assigned to every task and service a medical practitioner may provide to a patient including medical, surgical, and diagnostic services (http://patients.about.com/od/costsconsumerism/a/cptcodes.htm).

[24] HL 7 is an ANSI-accredited Standards Developing Organization providing widely used standards in health care. More information is available at http://www.hl7.org/.

[25] The WorldVistA EHR is available at http://worldvista.org/Software_Download.

[26] EsiObjects is a standards-compliant, object-relational, database management and interoperability system from ESI Technology Corporation.

[27] CCHIT has defined certification criteria in order to ensure that EHRs achieve measurable outcomes in patient engagement, care coordination, and population health. More information is available at http://www.cchit.org.

Extensions or modifications from the community are evaluated based on various criteria in order to decide about their integration into WorldVistA EHR. An expert group reviews these extensions based on the following considerations:

- License: extension must be GPL compatible,
- Compatibility: extensions must comply with the SAC (Standards and Conventions, maintained by the Standards and Conventions Committee (SACC)) and other technical standards in order to ensure compatibility with WorldVistA EHR,
- Quality: extensions must be suitable for use in terms of code quality (e.g., "good housekeeping" for variables), ease of code maintenance, and documentation,
- Relevance: extensions must be useful for other members of the community, and
- Feasibility: extensions must run on two different stacks (WorldVistA EHR with Caché and MS Windows or WorldVistA EHR with GT.M and Linux).

These criteria do not apply for "trusted" submitters such as VA and Medsphere. However, other extensions or modifications that do not meet these criteria are available on the WorldVistA server. One main reason that contributed source code is not approved by WorldVistA is that is has not been tested on different platforms. The testing of this source code by WorldVistA is often impossible because of a lack of funding. At the moment an important challenge is to receive a CCHIT re-certification of WorldVistA EHR.

2.4.2 VISTA Expertise Network

The VISTA Expertise Network[28] is a 501(c)(3) organization that aims: *"[...] to improve people's health through the use of technology by providing advice and hands-on assistance for new adopters of VISTA and affordable support for existing users."*

The VISTA Expertise Network has been founded by Rick Marshall and was incorporated in August 2007. The goal is to close the existing generation gap of VistA developers. The main reasons for this gap are on the one hand, missing training programs for VistA in the last decade and on the other hand, the growing number of retirements of VistA experts. In order to support the development of a new generation of VistA experts, the VISTA Expertise Network has developed a unique educational program called Paideia. In this program VistA experts teach VistA novices while working on real-world programming problems that exist in open source VistA distributions (e.g., FOIA VistA, WorldVistA EHR, RPMS). The program follows a *"learn-do-teach"* model, i.e. craftsmanship model, of education.

The network has partnerships in the whole VistA community such as with WorldVistA, with public health agencies (e.g., Department of Veterans Affairs, Indian Health Service), as well as with academic institutions (e.g., University of Washington). This cross-organizational approach creates a strong expert base and addresses the need for the revitalization of the VistA community.

The Paideia approach is a much more elaborated educational concept than WorldVistA's Educations Seminars (cf. Section 2.4.1) because of the integration of classes and long-term mentoring. But the VISTA Expertise Network supports neither community building nor development work to extend available functionality in open source VistA distributions.

[28] More information can be found at http://www.vistaexpertise.net/index.html.

2.4.3 VistA Software Alliance

The VistA Software Alliance (VSA) is a non-profit trade association (501(c)(6)), a trade organization for vendors, founded in 2004 for the purpose of promoting the VistA electronic health record system.[29]

It is committed to *"improv[e] health care by increasing the quality of care, reducing patient errors, and lowering costs"*. In January 2008, the VistA Software Alliance had more than 40 members, for example HP, IBM, and Perot (now Dell/Perot cf. Section 2.4.5). In opposition to WorldVistA, members of the VSA can be individuals and companies who are involved in the implementation, integration, and support of VistA distributions and who share a common understanding of VistA being a valuable business model. Members are therefore commercially interested in a higher adaptation of VistA. The VSA has emerged from WorldVistA members who *"felt that their perspective was not being served"* because WorldVistA is not involved in any business activities because of its non-profit status. Large hospitals especially use so-called "request for proposal" mechanisms. The VSA performs a clearinghouse function, enabling companies to apply for those proposals. Another issue has been the self-conception of WorldVistA because its activities have been more concentrated on system development rather than on promoting the system usage: *"[...] there should be more than engineers or groupies saying how great VistA is, there should be people who actually use it."* But even though companies joined the VSA, they retained their WorldVistA membership status. However, it is said that major challenges for the VSA have been the missing technological community and the costs to keep this organization running. Another issue has been the missing value because VistA software could not be sold as successfully as expected. Finally, in 2009 the organization fell apart.

2.4.4 Open Health Tools

Open Health Tools (OHT)[30], a non-profit trade association (501(c)(6)), was founded in 2007 (OHTF, 2010) in order to "*enable a ubiquitous ecosystem where members of the Health and IT professions can collaborate to build open, standards-based interoperable systems that enable patients and their care providers to have access to vital and reliable medical information at the time and place it is needed."* OHT will therefore be part of various communities in this area of Open Health IT.

[29] More information can be found at http://www.vistasoftware.org.
[30] More information can be found at http://www.openhealthtools.org.

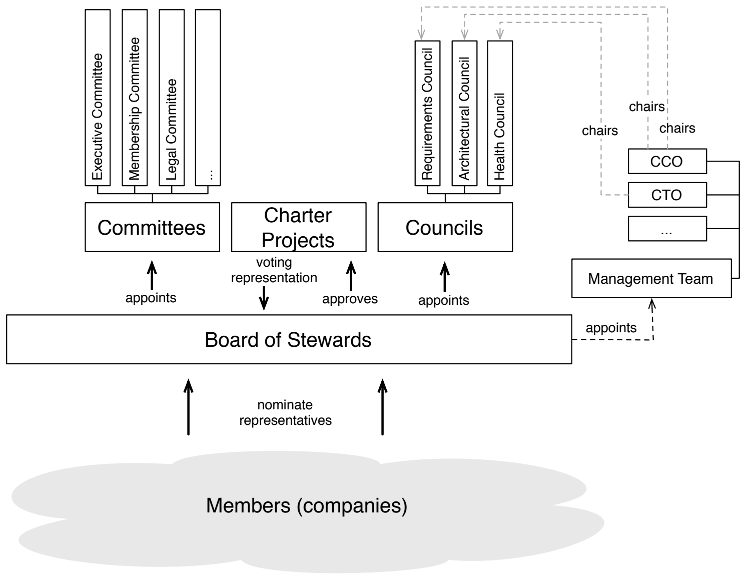


Figure 2-6: Organizational structure of Open Health Tools.

OHT is modeled on the Eclipse open source ecosystem[31] in its core areas such as governance, legal and intellectual property policies, development processes, and business models[32]. It will facilitate collaboration among diverse participants by involving stakeholders such as software companies, government health service agencies, and standards organizations. These stakeholders shall work together in a transparent software development process in order to produce software products collaboratively (Open Health Tools Foundation, 2010). The OHT will provide the needed neutral space for these activities by creating a *"common health interoperability framework, exemplary tools and reference applications"*. OHT supports all licenses that have been approved by the Open Source Initiative.[33]

In January 2009, DSS, Inc. (cf. Section 2.4.5) joined OHT, and contributed its VistA distribution vxVistA to OHT. Because of this contribution, OHT became actively involved in the VistA community and is now more widely recognized. This is the first actively promoted project of OHT, however, and it can be seen as a test case in the area of Open Health IT to verify the sustainability of its concept.

Governance. As of May 2010, Open Health Tools had 47 corporate members (with eight in process), including companies such as RedHat, IBM, and NexJ Systems, Inc., organizations such as Eclipse and the Veterans Health Administration (VHA), and academic organizations such as the Oregon State University.[34] The goal is to increase the number of members by eight in each quarter. The OHT is committed to the open source principles of meritocracy, openness, and transparency. All members of OHT have an equal status that is reflected by a single vote of each member. Individuals have the same importance as companies. Information, such as deliberations of Board and Councils, project information, software development processes, and guidelines are openly accessible and decision-

[31] More information can be found at http://www.eclipse.org.
[32] Although the Eclipse Foundation and the OHT Foundation are not affiliated in a corporate sense.
[33] The Open Source Initiative (OSI) was founded in 1998. It is a non-profit organization that stewards the Open Source Definition (OSD) and approves licenses as OSD-conformant. More information can be found on its website: http://www.opensource.org.
[34] Although OHT has only organizational members, persons are permitted as well if they *"can make substantial contributions"*.

making processes will be transparently designed. OHT follows a contribution-based membership model. There is no membership fee, but members are explicitly encouraged to make contributions such as assets, existing software, or other intellectual property, and participate in software development projects.

The organizational structure of OHT is shown in Figure 2-6. Each member can nominate representatives (Stewards) to the Board. The goal is to foster collaboration between the different board members. The Board additionally includes representatives of project leads and committers who have a vote in the Board as well. The Board is responsible for, amongst other things, policies, technology plans, and directions. It approves furthermore so-called Charter Projects and appoints Councils. At the moment there are three Councils: the Requirements Council, Architectural Council, and the Health Council. These Councils are defined to deal with specific issues of OHT, such as budget and intellectual property.

Technical Goals. OHT provides an open source technology repository. The idea is that different software with different licenses (proprietary and open source software) can coexist and can be combined into one bundle. For this, a rule-based engine is planned that checks under what conditions such a merge is possible. The objective is to maximize user choice.

The repository contains two kinds of projects - Charter Projects and Forge Projects. The Board approves Charter Projects; they are reviewed based on their goals, expected contributions, etc. and if they have been licensed under a commercially friendly license such as Eclipse Public License, Mozilla Public License, and the Apache License. A development process for those projects shall be defined in order to ensure high quality of the projects. For example, certain checkpoints shall be specified which are accompanied with certain sets of rules that are reviewed quarterly and becoming a committer in those projects shall be based on the principle of meritocracy. The OHT wants to provide a trusted environment for health care projects that might be related to each part of the EHR life cycle and can range from development of architectures over standards to trainings. Additionally, OHT supports so-called Forge Projects that are complementary to Charter Projects. For those projects, no certain quality criteria exist but there shall be a process how Forge Projects can graduate to Charter Projects. Forge Projects shall encourage innovative solutions because no pre-defined rule set exists. In April 2010, there are 9 either approved to proceed or approved in principle Charter Projects and 11 Forge Projects.

In order to harvest projects the OHT has defined two phases. In the first phase the goal is to gather as many projects as possible. This phase is almost finished as this report is written. In the second phase, collaboration within and between projects shall be enhanced. It is planned to have a release cycle of one year. Tools such as a source code management system, an issue tracking system as well as project pages, wikis, and discussion forums will enable collaboration between projects.

2.4.5 Commercial firms

It is fairly challenging to assess the number of business that are based on, related to, or that are using the FOIA VistA distribution. Because the software is in the public domain, it is not even necessary for a business to mention that their software is based on this distribution. However, based on the information we were able to collect, Table 2-2 shows selected vendors of VistA distributions. Three companies that provide open source products are briefly introduced and two vendors—Medsphere and DSS—are described in more detail.

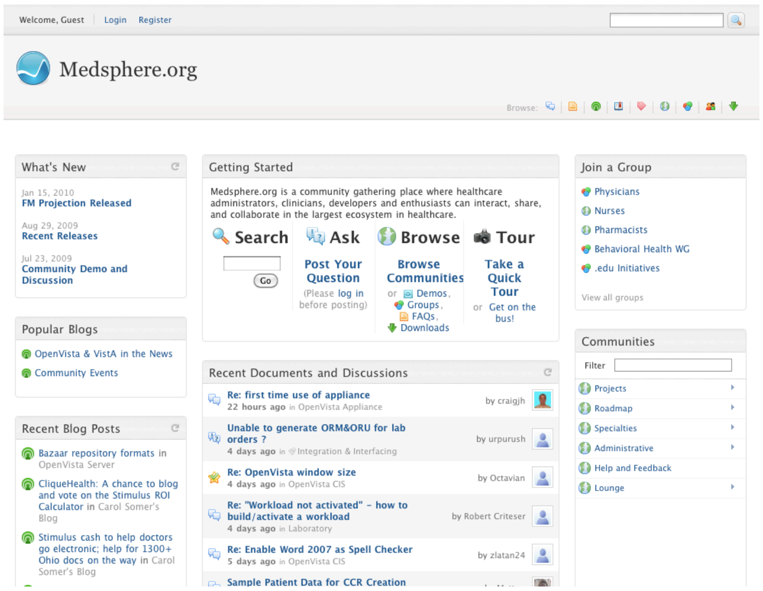
Astronaut, LLC[35] offers the VistA InstallerSuite for WorldVistA and OpenVista. Based on experiences gathered by implementing WorldVistA EHR in the IntraCare Hospital in Houston, TX, the necessity for an installer has occurred because the overall implementation process was very time-consuming and long-winded. The Astronaut installer, that packages the server, clients, and auxiliary modules, promises to reduce this process to less than one hour. Besides the Astronaut installer the company offers support, further development, training, and the Astronaut Shuttle, a pre-configured Amazon Machine Image (AMI) with Astronaut VistA pre-installed.

Dell Perot Systems[36] was founded as Perot Systems Corporation in 1988 and the Perot Systems Healthcare Group was established in 1995. Dell Perot Systems is a software integrator. Active software development does not exist. Dell Perot has been involved in various VA contracts, doing everything from configuration management to project management. One of the offered health care products is FOIA VistA and a major customer is Jordan with 46 hospitals (WorldVista EHR will be implemented). Within this project, Dell Perot Systems collaborates with Medsphere because they are using their OVID framework (cf. Section 4.3.5).

Sequence Managers Software, LLC[37] was founded in February 2004. It results from the Hui initiative (cf. Section 2.3). The company offers OnDemandCARE that is based on the WorldVistA EHR. Sequence Managers Software aims to deploy software and services for underserved areas, for example small communities clinics, rural hospitals, and custom development work.

In 2002, two brothers founded *Medsphere Systems Corporation*[38]. The company has various governmental contracts, amongst others, contracts with VA (e.g., several years for FileMan) and the Indian Health Service (IHS). For IHS, Medsphere provides the VueCentric front end for RPMS[39]. The VueCentric front end is based on a componentized CPRS (cf. Section 4.2). As of May 2010, Medsphere has around 100 employees; and 20 implementations of the company's OpenVista EHR solution[40]. Medsphere and/or its partners are currently in the process of implementing five additional non-governmental OpenVista hospital sites.

[35] More information is available at http://astronautvista.com.
[36] More information is available at http://www.perotsystems.com.
[37] More information is available at http://sequencemanagers.com.
[38] More information about the company and its mission is available at http://www.medsphere.com.
[39] Additional details are available at http://www.ihs.gov/cio/ehr/index.cfm?module=faq.
[40] The number of implementations does not include approximately 200 IHS sites.



What's New

Jan 15, 2010
FM Projection Released

Aug 29, 2009
Recent Releases

Jul 23, 2009
Community Demo and Discussion

Popular Blogs

OpenVista & VistA in the News

Community Events

Recent Blog Posts

Bazaar repository formats in OpenVista Server

CliqueHealth: A chance to blog and vote on the Stimulus ROI Calculator in Carol Somer's Blog

Stimulus cash to help doctors go electronic; help for 1300+ Ohio docs on the way in Carol Somer's Blog

Getting Started

Medsphere.org is a community gathering place where healthcare administrators, clinicians, developers and enthusiasts can interact, share, and collaborate in the largest ecosystem in healthcare.

Post Your Question (Please log in before posting)

Browse Communities or Demos, Groups, FAQs, Downloads

Take a Quick Tour or Get on the bus!

Recent Documents and Discussions

Re: first time use of appliance
22 hours ago in OpenVista Appliance — by craigjh

Unable to generate ORM&ORU for lab orders ?
4 days ago in Integration & Interfacing — by urpurush

Re: OpenVista window size
4 days ago in OpenVista CIS — by Octavian

Re: "Workload not activated" – how to build/activate a workload
4 days ago in Laboratory — by Robert Criteser

Re: Enable Word 2007 as Spell Checker
5 days ago in OpenVista CIS — by zlatan24

Sample Patient Data for CCR Creation

Join a Group

Physicians

Nurses

Pharmacists

Behavioral Health WG

.edu Initiatives

View all groups

Communities

Projects

Roadmap

Specialties

Administrative

Help and Feedback

Lounge

Figure 2-7: Screenshot of OpenVista community portal (April, 2010).

In 2004, the company acquired its first contract and deployed FOIA VistA outside VA for the first time in seven state veterans' homes in Oklahoma for the Oklahoma Department of Veterans Affairs[41]. In collaboration with Hewlett Packard (who provided hardware and project management experience) and Department of Veterans Affairs (provided consultative expertise), FOIA VistA has been installed with minimal customization. The Oklahoma project gave Medsphere the opportunity to develop a series of methodologies and tools for rapidly deploying VistA systems; many of these were the roots of the implementation methodology and tools in use today.

In 2005, Midland Memorial Hospital in Midland, Texas, requested FOIA VistA with extensive customization[42] for a private hospital setting. The OpenVista release contains the commercial changes made to the software in order for it to more closely match the workflow and processes of the private hospital setting. The adaptation includes removal of VA-specific elements/functions, small commercial enhancements, and a suite of interfaces for connecting OpenVista to departmental systems. In its first years, Medsphere mainly concentrated on software development, in an effort to commercialize FOIA VistA and move the applications onto an open source stack, but has now shifted the focus to providing hospitals with EMR implementations.

[41] More information isavailable at http://www.medsphere.com/company/partners/customer-partners/379-oklahoma-department-of-veterans-affairs.

[42] The hospital reported their cost were less than half of estimated costs for proprietary EHRs [Goth, 2006] or an independent case study at http://www.medsphere.com/midland-memorial-case-study.

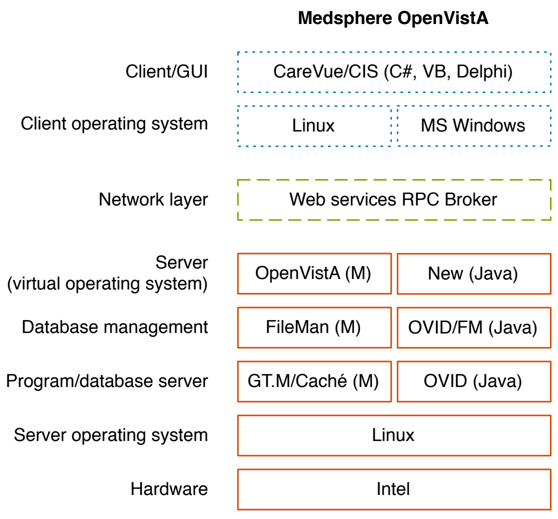


Figure 2-8: Medsphere OpenVista distribution.

In mid 2006, WorldVistA announced the coming release of a 2006 CCHIT-certified open source distribution of FOIA VistA. But because that distribution was not released for some time, the certification was ambulatory only, and the existing commercialization targeted for hospitals, Medsphere decided to release its own distribution - called OpenVista - under a AGPL license (cf. Figure 2-8). With its products and services, Medsphere is mainly targeting hospitals with a minimum of one hundred beds, whereas WorldVistA especially addresses small physician offices.

In 2007, Medsphere launched a community website[43] containing information regarding existing open source projects. This website mainly consisted of static information, release notes and software downloads. At the end of 2008, this website had been comprehensively re-designed in order to encourage more collaboration in the community (cf. Figure 2-7). The community portal especially supports collaborative development projects in the VistA community. It provides a development platform for OpenVista and periodic releases. Many projects have an open development process with code reviews and bug tracking, and in the future more and more projects will be managed in this way. Medsphere sees the community portal as providing a central location for collaboration and information, especially on OpenVista, but also for VistA in general, and it contains various channels of communication such as blogs, forums, wikis. Community development has become a more relevant part of Medsphere's overall business strategy (Medsphere):

"Medsphere also actively nurtures a Healthcare Open Source Ecosystem. This vibrant global community of customers, partners, developers and other online collaborators is growing daily as it helps drive OpenVista innovation, and provides a parallel development and support structure."

This is expressed for example by the collaboration of WorldVistA and Medsphere. In the last two years, WorldVistA and Medsphere have worked closely together on projects that are important for both organizations such as the CCR/CCD project (cf. Section 2.4.1).

[43] More information is available at http://www.medsphere.com/community and http://www.medsphere.org.

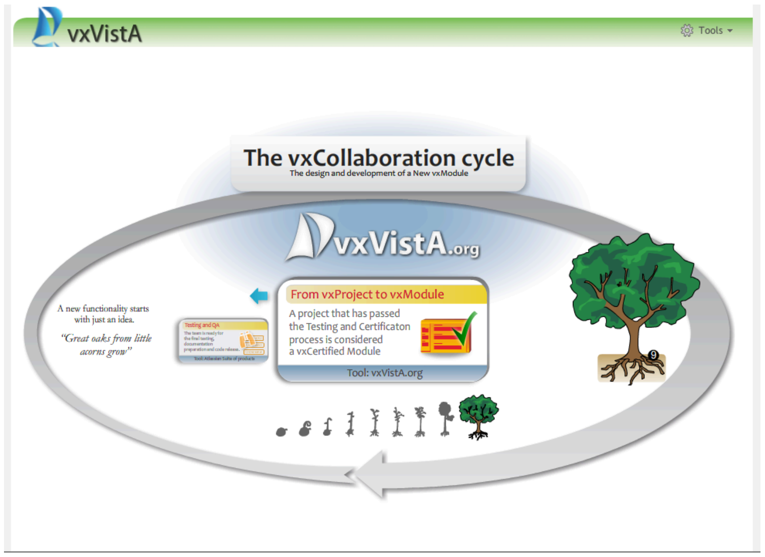

vxVistA
Tools
The vxCollaboration cycle
The design and development of a New vxModule
vxVistA.org
A new functionality starts with just an idea.
"Great oaks from little acorns grow"
Testing and QA
From vxProject to vxModule
A project that has passed the Testing and Certificaton process is considered a vxCertified Module
Tool: vxVistA.org

Figure 2-9: vxVistA collaboration cycle (April, 2010).

Major products of Medsphere are OpenVista Server, OpenVista CIS, OpenVista Interface Domain (OVID), and FileMan Projection. Core and foundation modules in OpenVista are are licensed under the LGPL license, application modules are licensed under the AGPL, and the server distribution is released under a mixed LGPL/AGPL license (cf. Appendix E), in order to improve collaboration in the community and to avoid further forking. The OpenVista Server is based on FOIA VistA. Existing errors in FOIA VistA were eliminated and commercial enhancements and interfaces were included. A cross-platform frontend for the OpenVista Server is the OpenVista Clinical Information System (CIS) (cf. Figure 2-4). It is based on VA's Computerized Patient Record System (CPRS) and includes commercial enhancements such as image viewing. The OpenVista Interface Domain (OVID) is an RPC[44] resource-messaging interface with Java bindings. It comprises a set of development tools to ease the access to OpenVista data using Java[45]. A number of platform technologies have been developed using OVID (or on its underlying technology), including FM Projection (an SQL-like projection tool), FMQL (a semantic web end-point), OpenVista REST (a RESTful web service layer) and others.

In 1991, *Document Storage Systems*, Inc. (DSS) was founded and today the company has around 250 employees. DSS had been developing only proprietary modules for VistA in VA for over 14 years and conducted over 2,300 VistA installations. At the beginning of 2009, DSS announced its first open source release of an enhanced version of FOIA VistA - vxVista; for example, vxVistA uses medical record numbers instead of social security numbers to identify patients, and is CCHIT certified. At the moment there are four implementations (e.g., Texas Tech Health University, Bayside Clinic) and four are in progress (e.g., Ramsey County Mental Health). vxVistA is not a single, monolithic solution. It is rather a modular system that can be customized depending on existing requirements.

[44] Remote Procedure Call (RPC) is an inter-process communication technology.
[45] For example it is used by the company Contineo to provide ubiquitous portable access to critical information on mobile devices such as the iPhone.

This VistA distribution is specifically designed to enable collaboration between companies based on a plug-in model. The Eclipse Public License (EPL)[46] was chosen to ensure that open source and proprietary products could be integrated. vxVistA can be seen as a core platform. On top of this platform companies can develop additional products by programming and packaging plug-ins, modules and extensions (DSS, 2009). More than 20 companies are already working with DSS to run their products on top of vxVistA. vxVistA is integrated in the VA's patch stream in order to maintain the FOIA VistA basis.

Another major effort of DSS is to create and sponsor a community (*"melting point"*) around vxVistA and therefore a community platform[47] was launched at the beginning of 2010. The goal is to build a collaboration cycle (cf. Figure 2-9) that is triggered by users' needs and their ideas. Developers whose efforts are supported by Atlassian donated collaboration tools such as Confluence and Jira implement best practices. The community platform will help customers to identify specialists and will support companies and specialists to promote their knowledge. At the end of 2010, it is planned to provide the complete collaboration functionality[48] on this website.

The overall development process consists of four stages. In the first stage, any community member can propose a new module but the whole community rates this suggestion. After identifying a project leader and development team, the Community Board will define this project as vxCommunity Project. A project will remain in this status until the new module has passed a pre-defined testing and certification process. A passed module reaches the third stage—vxCertified modules. The vxVistA Governance Committee will finally decide which vxCertified modules are included in the core stack of vxVistA. However, customers can individually decide which modules they want to use. vxCompatible and vxUnsupported modules will be provided as well. A description of the whole process is at the moment only available on this meta-level but further refinements are planned.

DSS has joined the Open Health Tool Foundation (cf. Section 4.3.4) and vxVistA will be an OHT Charter project (DSS, 2009).

2.5 Perceived challenges for the VistA ecosystem

In order to provide one global picture of the VistA community, in this section we consolidate responses from our interviewees (cf. Table 2-1) concerning the challenges facing the VistA ecosystem. In the first part of this section, responses containing existing challenges are summarized, and in the second part, these perceived challenges are discussed in relation to the literature.

2.5.1 Challenges seen by community members

In the previous sections, various members of the VistA ecosystem were briefly introduced. This information and additional data collected in the interviews are now utilized to describe existing challenges.[49] The main challenge for providing an open source health care system is to build a successful ecosystem, because the *"[s]urvival of VistA is closely connected to the survival of the VistA*

[46] More information regarding the EPL is available in Appendix E.
[47] More information is available at http://www.vxvista.org .
[48] DSS cooperates with Atlassian (http://www.atlassian.com) and is using products such as Confluence, Crowd and Jira (with GreenHopper).
[49] Unless otherwise indicated, quoted passages are taken directly from our interviews with VistA stakeholders.

ecosystem." Only an ecosystem that mirrors the specialties of VistA, therefore, will support its further development. In the following paragraphs, further challenges are discussed.

At the beginning of 2010, three main community models were differentiated by community members: the WorldVistA model, the OpenVista model, and the vxVistA model. Because of these three models, the ecosystem does not have one coherent organization. Our interviewees noted that one solution to overcome this dilemma might be an *"umbrella"* organization or consortium that unifies the community and involves all members of the ecosystem. However, a balance is needed between the different interests and the type of license, which is seen as a major pitfall. Although people want this change, they fear *"[...] differences that can pull the community apart."* However, different members of the ecosystem are striving towards a unified community. WorldVistA, for example, is trying to *"provide the framework to bring together diverse points of use"*.

But there are divergent views regarding the role of WorldVistA in the ecosystem. According to one group of interviewees, the *"WorldVistA community is [...] very disorganized; the organization should be improved."* They do not view WorldVistA as a neutral mediator, instead they see a need for an organization that supports open collaboration and feedback loops. Another challenge seen by our informants is that WorldVistA does not offer a membership model to its community. The Board of Directors makes all strategic decisions, and possibilities to actively participate in the general strategy of the Foundation are very limited. Neither decision-making processes nor development processes are transparent. On the other hand, even though WorldVistA is not a membership organization, participation is highly encouraged by the Board. In the past, new Board members were mainly recruited from these voluntary participants. The Board itself just oversees community activities and provides strategic direction when needed, but this strategic direction is strongly influenced by the community and its needs.[50] The contrasting perceptions of WorldVistA can be seen in the following comment: *"there are gatekeepers [in WorldVistA] that make sure that VistA does not change, but [at the same time it is] positive in terms of protecting [VistA] from vandalism and providing stability."* Besides these differences in perceiving the role of WorldVistA, another challenge WorldVistA is facing is funding: *"I don't see that we have a real good funding source, right now."* There are only a few volunteers who keep this organization running. One WorldVistA representative explained that there should be goals to appoint a grant writer on WorldVistA's Board of Directors and to develop strategies to get more people involved.

We were told that Medsphere has become an important role because WorldVistA has some difficulties in coordinating and directing existing development initiatives: *"[t]here is a vacuum and WorldVistA has not filled that vacuum [...] Medsphere is [...] essentially trying to fill that vacuum."*. Medsphere's community portal not only contains general information on VistA but also gives an overview about existing development projects. Medsphere even plans to provide an open but formalized development process in order to increase transparency for its internal processes and to attract more people to engage in the development. Even though the production processes will be more openly designed, our informants told us that Medsphere does not offer community members the chance to participate in the overall decision-making process of the product. Medsphere applies the AGPL license in almost all its products; this might be one reason why Medsphere and WorldVistA are partners in community projects. Our informants told us that WorldVistA has often assumed the role of initiator in those projects. Although Medsphere designed a fairly open environment around its product, it hinders real participation because it lacks a formal concept to invite external contributors.

[50] We are grateful to Joseph dal Molin for pointing this out.

Another challenge for participatory software development within the community is the unavailability of a source code management system for VistA. The whole development of VistA is based on convention and patch sequences and therefore the community must constantly re-integrate contributions manually. We were told that *"investment in tools (or adaption there of) like distributed revision control [...] would be a massive accelerator for the VistA codebase."*

However, our interviewees told us that they would like to see a more active involvement of VA in the community because *"VA is been the mother ship of the VistA technology"* but *"VA has not done a good job of developing their software"*. In the past, VistA has been seen as legacy system in VA and there were only low investments in the system. Therefore, VA did not hire new MUMPS developers and new expertise was mainly built up outside VA (e.g., see, VistA Expertise Network, Section 2.4.2). We were informed that a main issue working with VA has been the missing consistency in their actions *"[...] in the last 10 years the reporting structure within the VA has changed several times"*. These organizational changes might have caused that for example two years ago, new patches for VistA were released every two months but since then it is only once a year. Our informants told that at the same time, there are approximately up to 800 changes a year within VA. However, we were told that other governmental organizations such as the Department of Defense (DoD) approached this challenge differently as they support research projects, providing grants, and release new technology in order to support new businesses that can in turn support the DoD.

Another difficulty is the lack of a sufficient number of MUMPS developers, although the WorldVistA GPL license might attract other developers to join the community. Young developers are generally more interested in "*modern*" languages such as Java. MUMPS is often seen as an ancient, dying technology albeit it has many unarguable advantages for HIT systems. Recruitment is therefore seen as one of the major challenges in the community. The success of attracting new members will heavily influence the survival of the whole community. Universities such as Robert Morris University in Pittsburgh, Pennsylvania are very important because they are educating students to use MUMPS.

Our informants fear that the different distributions of VistA lead to higher fragmentation of the ecosystem. Reasons for this fear are, on one side, the community is fairly small, especially when considering the developer pool. Each organization finds the process of attracting experienced programmers more challenging. On the other side, adopters are insecure about the survivability of the different distributions. This situation endangers the survivability of the whole ecosystem. At the same time, available information is distributed over the whole community and therefore, *"[there are] ten places were you find information"*. Especially people that are new to the community *"[...] are having a hard time to get access to information they need"*. Specific sessions during the VistA Community Meeting specifically addresses those needs but it is still challenging to get a first overview about available distributions and their differences: *"we need to be organized open source wised in that same way, that everybody understand and we can clearly communicate [that way]"*. Even though VA provides their comprehensive VistA Documentation Library, there is no document available that shows existing differences between VistA used within VA and the FOIA distribution.

The overall complexity of VistA makes it very challenging to use in small offices. In VistA, existing user interfaces are often too complicated for small offices because they offer unnecessary functions. Our interviewees informed us that configuration tools are needed for small hospitals and that a better documentation of VistA configuration would be helpful. At the same time, FOIA VistA does not provide critical functions such as billing and e-prescribing: *"Real stopper for small doctor's office is the [missing] billing [system]."*.

We were told that companies such as Medsphere and DSS could fill this gap by providing their own proprietary solutions. But those missing modules are a drawback for VistA's functional capabilities because interested parties are less likely to invest in such an incomplete solution. The question remains whether a joint community initiative could provide a complete open source solution. At the moment, there is no clear path to a single distribution or efforts underway to minimize variations between the distributions of VistA. There are also people who see a need for a completely new infrastructure for VistA. Regarding the age and the size of the software it was pointed out that this requirement is difficult to realize.

Communication processes in the ecosystem are mainly carried out on the hardhats mailing list[51]. This list seems to be used by all members of the community. However, all community members additionally use their own communication tools. Interviewees see an improved overall transparency of the community if all used only one set of communication channels.

2.5.2 *Discussion of perceived challenges*

The overall framework of this ecosystem is unique because VA does the vast majority of development work even though it is not part of the ecosystem. Knowledge transfer between VA and the open source community mainly occurs because employees of VA are actively involved in the community in their spare time. Additionally, VA has opened up attendance to the Veterans E Health University, their annual educational conference, to people outside VA, and provides access to the VistA Document Library (VDL), an online repository of user and technical documentation[52].

However, both sides are losing existing opportunities such as sharing source code, collaboratively developing extensions, or exchanging expertise. Moreover, the VistA ecosystem revolves around several different distributions of the same source code. The unique situation of having almost completely governmentally-developed software seems to cause certain challenges for the development of a successful and sustainable ecosystem.

One important dimension to create a successful ecosystem is the governance regime (cf. Appendix A). A governance model reflects expectations of its community members. It defines how potential contributors should and can engage with the project and shows who exerts what kind of control. A governance model can range from centralized control of a single person, a small group of people or an organization (benevolent dictatorship) to a distributed control that honors contributions (meritocracy) (Ljungberg, 2000). These forms can be seen as polar opposites in a spectrum. A governance model can be at any given point in between. Depending on the maturity of the open source project, the governance model might change over time.

WorldVistA's organization can be seen as close to Linux kernel development (high tendency to benevolent dictatorship) because the majority of power is centered on a few people i.e., the Board of Directors. Such an organization is often chosen to prevent a takeover by another organization or firm. The management of community-initiated projects, such as WorldVistA, normally changes organically. In the start-up phase of those projects, decision-making is mainly carried out by its founders in a cathedral-like central control (Raymond, 2001). Because WorldVistA emanated from the Hardhats group, the usual community development pattern can be seen. Although the software itself is quite mature, its initial open source implementation is relatively young. In community-

[51] An overview about existing mailing lists and their number of members and posts is given in Appendix F.
[52] More information is available at http://www4.va.gov/vdl/.

initiated projects, the selected license heavily influences the attractiveness for other developers to join the project (West, & O'Mahony, 2005).

Despite community-initiated models such as WorldVistA, there are sponsored open source models (cf. West et al., 2005; West, & O'Mahony, 2008b). Medsphere with OpenVista and DSS or OHT with vxVistA belong to the latter group. Both companies have made their source code available to the public.

In our interviews, the role of Medsphere was widely discussed but vxVistA/OHT and its role are more difficult to assess because it is relatively new in the open source VistA sphere even though it has extensive experience with VistA within VA. In comparison to Medsphere, DSS decided to contribute its source code to OHT which, as a non-profit foundation, serves as a neutral body. Modeled after the Eclipse Foundation, its form of governance encourages contributors to actively participate in decision-making processes. OHT wants to ensure that no single entity is able to control the strategy and that the process of meritocracy drives the community structure. OHT gives every single community member the possibility to actively engage in the community. The technical architecture of vxVistA is designed in a more modular manner. By providing the core VistA system, interested parties can add additional functionality in independent modules by using existing APIs. This increases existing incentives for potential developers to join the community and to participate in the development process (cf. Baldwin, & Clark, 2006). DSS wants to bring together various open source solutions on the same platform. However, the chosen EPL license hinders possible collaboration with other stakeholders in the ecosystem, especially WorldVistA, because EPL and GPL code can not be mixed (cf. Eclipse, 2010).

In this section, we have reported the results of our case study of the current VistA ecosystem. In order to enable a comparison with several larger, more established ecosystems, we next develop a set of dimensions on which ecosystems can be compared.

3 Possible paths for VistA

In this section, we seek to apply the lessons from existing ecosystems to VistA (see Appendices A-C for a detailed description of our cases studies). Our starting point assumes that what is common across all four ecosystems is likely to be a relatively general characteristic of successful ecosystems, and therefore something likely to be helpful or even necessary if a VistA ecosystem is to grow along the same lines. Characteristics that differ among the four present ecosystem design choices.

We derived the main dimensions of our comparison from our analysis of the literature on open ecosystems and our own research (see Appendix A for a more detailed discussion). In brief, the dimensions are

- **Technical platform**: The technical characteristics of the platform that provide common functionality and support integration of contributions from multiple participants.
- **Governance regime:** The particular way that decisions are distributed over decision-makers and decision-making mechanisms, supporting both coherence and innovation.
- **Collaborative infrastructure**: collaboration technology, practices, and culture that support effective communication and coordination of participants.
- **Business opportunities**: Viable strategies for generating and appropriating value in the context of a socio-technical ecosystem.

These dimensions should be considered mutually because interdependencies exist among them. However, we do not attempt, of course, a complete enumeration of all the choices that lie in front of the VistA community and VA. This strikes us as an impossible task. Yet there are several fundamental decisions about the characteristics of the ecosystem that we believe will profoundly influence and constrain the directions it takes in the future. After making some general observations about the VistA ecosystem, we address the choices to be made along the four dimensions.

3.1 General observations: unification or fragmentation

Each of the four open source ecosystems we studied for this report has rallied around a single development branch, and a single primary distribution. Perhaps the most profound decision the VistA community has to make at this point is whether to unify the community around a single distribution or to continue in a fragmented way with many distributions. The latter path was chosen by Unix, for example, a choice that *"increased supplier and customer costs for development, application porting and system maintenance, and slowed the adoption of Unix in a number of segments"* (Jaruzelski, & President, 2007). This inefficiency is particularly damaging for the VistA ecosystem, since the platform, at least for the near term, is written in MUMPS. There are relatively few MUMPS developers, so stretching them among several distributions and duplicating each other's efforts is a serious concern.

Moreover, the benefits of product complementarity (Brandenberger et al., 1996) are limited if complementary products are available for only a subset of the distributions. For a company selling a particular application, for example, the existence of other vendors selling other applications is a benefit, since a wider and more complete choice of product configurations is likely to increase overall market share for the platform, thereby growing the market for everyone. Fragmentation limits the benefits of complementary products, since an application developed for flavor X may not run on flavor Y, and therefore cannot act as a complement for other flavor Y applications.

On the other hand, there are several forces working toward fragmentation. Several vendors have invested in particular flavors of the VistA platform, and will be loath to abandon them. Moreover, feelings run high in some quarters about open source licenses. Some individuals feel that a commercial-friendly license like EPL is essential if an ecosystem is to grow to a critical mass, where a large number of vendors produce a full range of products. Others feel that GPL-style licenses are more appropriate, since these would encourage volunteer participation, and help prevent firms from unduly profiting from the work of volunteers. While we do not try to thoroughly analyze the consequences of varying licenses, we note that among our cases, some have commercial-friendly licenses (Eclipse, and Apache, and Mozilla), while GNOME is GPL. All have substantial commercial participation. Unifying the ecosystem would be much easier, and participants would likely reap more of the benefits of unification, if a single license or compatible licenses were chosen.

As a practical matter, fragmentation is difficult if not impossible to prevent, given the nature of open source licenses and the always-present possibility of forking. The primary measures that can prevent forking are a cultural norm that discourages it (Stewart, & Gosain, 2006), and the accumulation of complementary products that act, in effect, as a barrier to entry for new distributions (Katz, & Shapiro, 1985). Even in the face of the currently fragmented collection of distributions, a sufficient investment in one of them would likely reach a tipping point, leading to a relatively unified community built around the favored distribution.

In addition to the type of license, an equally important question is who would own the intellectual property? In three of our comparison cases, ownership of the code is vested in a foundation. This is important to participants, as we found in the case of Eclipse (Wagstrom, 2009), where ecosystem success was dependent on having a non-market player in control of the platform. Profit-driven platform owners may build profitable features into the platform (Gawer, & Henderson, 2007), thereby appropriating their value at the expense of application vendors. A foundation as platform owner enhances trust, reduces perceived risk, and makes participation more attractive. A foundation has no profit motive, and can focus credibly and transparently on ecosystem stewardship.

3.1.1 The key choice is:

- Will the community continue to maintain several distributions or will it rally around a single distribution with uniform or compatible licenses?

3.2 General observations: The role of VA

While VA, as an agency of the federal government, has a general obligation to act in the public interest, it has the specific mission to *"provide veterans the world-class benefits and services they have earned"* (Department of Veteran's Affairs, 2010). Their interest in HIT, including VistA, derives directly from this mission. For this reason, promoting a VistA ecosystem external to VA will inevitably be a low priority.

Because of this internal focus, VA is a very unusual ecosystem player. Rather, it is not an actor within the ecosystem at all, except for providing the legally mandated FOIA releases. Unlike most ecosystem members, it does not make use of an open distribution, but rather maintains its own software. It does not have a regular mechanism for accepting modifications from the external distributions, so the flow of software is almost entirely in one direction, from VA to the outside. It is also unusual that its resources vastly outstrip those of other ecosystem participants, with the consequence that the other participants have essentially no leverage over its policies. The ecosystem provides no incentives – no rewards or costs – to VA.

These characteristics of VA non-participation and disproportionate size have a powerful damping effect on the ecosystem. Since the VA does not regularly accept modifications, in order for the distributions to retain these modifications, each VA FOIA release must be patched with all of the non-VA changes, and any conflicts resolved. This means that extensive modifications outside VA would be extremely difficult to maintain, as the integration effort would grow – probably at a superlinear rate – with the number of modifications. The alternative would be to stop updating external open distributions with FOIA releases, but given the resources and expertise VA has at its disposal, the VA's contributions are too valuable for this to be a practical option. The upshot is that the current mode of operation makes it difficult for the ecosystem to attract contributions from outside VA.

Yet there are substantial barriers in the way of VA incorporating external changes made to an open VistA distribution. VA has a number of requirements that many other health care entities do not. For example, VA must ensure that all changes are nationally deployable, not just deployable in a single-hospital environment. Systems must be "508 compliant", meaning that accessibility rules apply to federal government systems that do not apply elsewhere. Experience at VA, as related by one interviewee, has been that when they have attempted to bring in code developed elsewhere, they had to rewrite almost all of it. This means that bringing in external code produced by the open source community often has substantial cost and little benefit to VA.

One possible approach to this problem, suggested by the other ecosystems, would be an incubator mechanism. The studied ecosystems all have ways of bringing new projects into the distribution. Some are more formal "incubators" with clear stages and review processes, while others are a more informal mentoring arrangement. But in order to qualify for inclusion in a distribution, a new project has to meet certain quality and utility standards, and be approved after careful review. Such a mechanism could provide screening and mentoring for projects to ensure they meet the VA's needs. Projects not meeting the standards could still make their software available, but it would have to be downloaded and installed separately from the main distribution. Another possible approach to consider enhancements or changes needed outside VA is to implement configurable parameters, which can be turned on or off based on the environment.[53]

VA stands to gain substantially from greater participation. The experience of Red Hat with the Fedora distribution provides a useful lesson (see Appendix G). Red Hat is a Linux distributor whose primary product is an enterprise version. In order to be appropriate for an enterprise, the distribution must be extremely stable so that it can be certified. Red Hat maintains a free version of Linux, called Fedora, which is more volatile, and to which the open source community can much more easily incorporate changes. From Red Hat's point of view, they receive, in effect, new innovations and free testing services from Fedora. When a new feature proves to be sufficiently stable, it can be brought into the Linux Enterprise edition. VA could potentially benefit from a similar strategy, bringing new features into its software only after they have proved themselves, and only after the required VA-specific work has been done. Achieving these benefits however, would require substantial architectural changes. In particular, it would require defining and implementing a platform and stable APIs so that new functionality could be maintained separately as needed (see Section 3.3).

Besides the VistA community there are other Open Health IT initiatives such as OpenMRS[54]. OpenMRS is a software platform and a reference application, which has been developed to enable people to design customized medical record systems without programming knowledge, especially in the third world. By considering an engagement in existing open source initiatives it might be even interesting for VA to actively participate in those communities in order to exchange existing knowledge, to collaborate in selected projects, and share resources.

These mechanisms could be combined into a strategy that works to the benefit of VA as well as the ecosystem as a whole. The elements of a strategy are:

VA performs or directs the vast majority of platform maintenance and enhancement. This is almost unavoidable, as seen in the eclipse ecosystem (Wagstrom, 2009). Viewing the platform as a public good (Olson, 1971), no firm is likely to invest in platform maintenance unless the benefit of that particular investment to that specific firm is sufficient to justify it. Most platform maintenance and enhancement activities do not have this property, but rather provide a small, diffuse benefit to all ecosystem participants. No firm is likely to shoulder the cost on behalf of the community. VA, on the other hand, benefits sufficiently from the platform, relying exclusively on it (and applications that run on it) for its IT needs, that it will continue to perform or fund most platform work.

[53] We are grateful to DSS, Inc. for extending this aspect.
[54] More information can be found at http://openmrs.org/.

Vendors will provide small, focused platform enhancements for free. There will be occasions when firms whose customer base runs the VistA platform will want to upgrade the platform because the firm's business needs require it. They may need an enhanced API, improved performance, or the ability to interoperate with new hardware in order to improve their applications, create new applications, sell hardware, or provide new services. When the return to the firm is sufficient to justify this change, they are likely to provide it for the platform, as happens in many other ecosystems. Concealing their contribution by forking the platform would be far too expensive, since they would quickly lose the ability to incorporate future changes made by VA and other firms. The cumulative benefits of such small changes can be very substantial, as the highly specialized expertise of many firms is built into the platform. This can only work, of course, when such changes can be brought into the main distribution and when the culture supports collaboration and requires transparent technical discussions so that any disruptive side effects can be identified and avoided.

Vendors will provide the bulk of application development, enhancement, and maintenance for a free distribution. Firms have powerful incentives to provide fully-functional, but basic versions of applications that can be distributed under an open source license. Giving away a basic version of an application that can actually be used by downmarket customers still leaves abundant revenue opportunities for services such as installation, training, and customization. It also paves the way for sales of enhanced versions as needs grow and become more complex. Although traditional firms may be reluctant to give anything away, a form of brinksmanship is likely to create intense pressure, since there is a very large first mover advantage. The first vendor to "give away" a version of an application will see this application widely distributed very quickly. Many professionals, administrators, students, and others will see it in action, will become accustomed to its functionality, its look and feel, and will want similar systems. These expectations, and a desire not to re-learn and re-train, will give the vendor a big advantage in upmarket sales of enhanced-value versions of the application. Both the downmarket service opportunities and the upmarket sales of enhanced versions, as well as services, will be greatly diminished for second movers, particularly if the release lag is substantial relative to the rapid dissemination of the free software.

While vendors will naturally provide the bulk of the effort for maintenance and enhancement of basic application versions, the same logic applies here as with the platform. Small, specialized firms may have urgent needs for bug fixes or enhancements for their own business purposes, and will provide them to the ecosystem, to the benefit of all users of the application as well as to the benefit of the firm that is the primary application developer. As with the platform, the cumulative benefits of "long tail" contributions may be considerable.

VA will fund application enhancements required by VA, and release the enhanced versions to the ecosystem. Special needs for national deployment and regulatory compliance mean that applications developed for other customers will often fail to meet VA's needs. VA will prefer applications that have a free version available in the ecosystem in order to reduce its costs. Modifying existing applications – particularly for savvy vendors who have planned to sell applications to VA – will be much less costly than building applications from scratch. VA will, in effect, share the cost of the basic application with other customers, as such costs are recovered by the vendor in upmarket sales, services, and customization. VA will insist on contractual arrangements that provide for the enhanced versions to be released under an appropriate open source license so that other customers and government agencies have the benefit of the enhancements, and in order to maintain a single distribution that allows future enhancements by others to be incorporated.

3.2.1 *The key choice is:*

- Will the VA commit to full participation in the ecosystem, and adopt an ecosystem-based software strategy?

3.3 Technical platform architecture

Defining a platform and appropriate APIs are probably the most important and difficult steps toward building a prospering ecosystem and encouraging greater participation in the ecosystem. Platforms are typically defined in a process described as *"coring"*, or identifying elements that *"resolve technical problems affecting a large proportion of other parts of the system"* (Gawer, & Cusumano, 2008). The core would consist largely of infrastructure software that would be needed for most any implementation, and would be independent of the applications that provide specific functionality for particular departments or services. Defining this core is a difficult technical undertaking, but having a platform is fundamental to ecosystems, as we saw in all four case studies. The technical design of a platform provides the mechanisms needed for various contributors to work together. Of course, a wide variety of such mechanisms exist, such as plug-in architectures (e.g., Eclipse), standards-based protocols (e.g., Apache projects), and shared libraries (e.g., GNOME).

Platforms require interfaces in order to be useful, of course, and defining the APIs would also be a critical task. Fortunately, open standards are already available for many kinds of medical applications (see, e.g., (HITSP, 2010)). While such standards would generally guide design of APIs, the specifications would need to include additional information essential for effective interoperation of implementations.

Careful definition of stable APIs is not only essential for attracting vendors to the ecosystem, it is essential for allowing platform evolution. Stable API's can provide an abstraction layer between VistA's highly integrated data and logic tier and presentation tier. A platform, consisting of the data and logic tier, is the most stable component of an ecosystem, but the interfaces are more stable than the interior core of the platform (Baldwin, & Woodard, 2009). So long as the APIs are undisturbed, the platform code and at the same time the presentation tier can (and usually does) evolve. This would allow a migratory path from existing applications.[55]

Assuming VA decides to pursue the many advantages of a platform-based architecture, the key choices concern how to get there. VA has many pressing internal forces that will impact their decisions, and we do not attempt to assess them. Our focus is on the impact such decisions made by VA and other actors will have on the ecosystem.

VA could choose to move in the direction of closed source (e.g., contracting for proprietary code not covered by FOIA). This closed option would, of course, be devastating to the ecosystem. No other organization in the ecosystem currently has the resources to maintain and update VistA, the software would quickly become obsolete, and no party would have an incentive to participate in an ecosystem except to offer services and systems to the most cash-strapped organizations. Given the size and complexity of VistA and the dearth of resources the ecosystem would attract, the situation could not be sustained for long.

[55] We are very grateful to Ben Mehling for pointing this out.

On the other hand, VA might choose to internally evolve VistA in the direction of a platform with well-defined, stable APIs, and to resist proprietary code, perhaps by negotiating a contract calling for the release of any vendor-developed platform source code. Assuming that such a platform is released and staged appropriately, this could form the basis of a thriving, vibrant ecosystem. A critical problem would be drawing enough interested parties quickly enough to be able to build a critical mass of applications on the platform (often called *"tipping"* (Gawer et al., 2002)). This could presumably be addressed by publishing plans and specifications well in advance, and perhaps providing financial incentives to draw vendors to the platform. In this scenario, VA is essentially just another customer for application vendors (although one with great leverage given its control over the platform).

A closely-related issue is platform maintenance. It has long been known that incentives for collective action are problematic when the returns from an individual's actions are not sufficient to justify the cost (Olson, 1971). This tends to be true even in those cases where a set of actors would each be better off if all agreed to share the cost of some action. The problem is that each would receive a higher return by defecting, i.e., incurring no cost and allowing the others to do the work. Platform maintenance can suffer from this phenomenon. If each ecosystem participant would make a small investment in maintaining the platform, all would be better off. But absent some mechanism for compelling the investment, defection would be very tempting – rational in an economic sense – since no matter what the other participants do, a given participant is always better off doing nothing.

The ecosystems we examined have addressed this problem in several ways. Platform maintenance in Eclipse is performed virtually entirely by IBM (Wagstrom, 2009). IBM makes many uses of the Eclipse Rich Client Platform, and presumably benefits sufficiently from its maintenance and enhancement that the return is adequate to sustain the behavior. In Apache, as we noted, the platform httpd server is relatively small and stable, requiring only fairly minimal maintenance effort. In GNOME, several participants are distributors, who require a complete up-to-date distribution, including the platform, so have sufficient incentive to ensure it is maintained (Wagstrom, 2009). Other participants maintain small parts of the platform they require for their own business reasons.

If VA takes the open platform route, the maintenance issue needs careful consideration. It is unlikely that any other organization has the resources, expertise, or motivation to maintain and enhance the platform by itself. If VA uses the platform internally, i.e., it uses some version of the main distribution, then it may be able to provide the bulk of maintenance effort for the ecosystem, as IBM does for Eclipse. Alternatively, if VistA is owned by a foundation, it could require payment from members (perhaps on a sliding scale, as many ecosystems do) for use of the platform, and could fund maintenance from these revenues. Finally, the platform could be privatized, and provided on a market basis, perhaps with a requirement to have a free "vanilla" version available for some segment of the market.

The outcomes of a platform strategy would also be heavily influenced by product and pricing decisions by vendors. Vendors would have to make decisions about whether or not to release free (or very low-cost) versions of their essential applications. This has happened in many ecosystems, for sound business reasons. In Eclipse, for example, Actuate joint the Foundation as Strategic Developer and Board Member and released a highly functional version of their business intelligence reporting tool (BIRT) under the EPL license. By doing so, they pre-empted their competition, since all Eclipse users interested in business intelligence will likely have their first experience with BIRT, and want products with similar functionality and look and feel. Actuate sells high-end versions of BIRT, along with services such as workshops and training courses. The move dramatically grew their share of the Eclipse business intelligence market. In the case of VistA, providing free versions of essential applications is likely to have a similar effect, growing the VistA overall market share, at the expense of fully proprietary systems.

One issue that creates differences between VA VistA and VistA distributions is the use of some custom, proprietary software by VA, such as Caché. This makes evolution more difficult, since the software not supplied by VA must either be purchased, or have a free substitute in the VistA distributions, such as GT.M. These differences make the VistA software less valuable, and create potentially difficult integration problems. While this issue may not in all cases be solvable, it would greatly benefit the open source VistA community if VA were able to achieve contractual terms with its vendors that allowed it to release the source of the custom software. Given VA's extraordinary bargaining power, several members of the VistA community believed VA could readily negotiate such contracts.

3.3.1 The key choices are:

- Will a platform be "cored" from VistA? What will it include?
- What APIs will be defined and developed? What functionality will be exposed for each? In what specific way will the API make the functionality available?
- Will free, open versions of applications be included in a distribution of the platform? If so, which ones? With what level of functionality (will it, for example, qualify for "meaningful use"?).
- Will VA choose contractual arrangements with vendors that will make all useful platform code available to the community, or will vendor code continue to be regarded as proprietary?

3.4 Collaborative infrastructure

Successful collaboration over distances and across organizational boundaries is widely recognized as a very difficult problem (e.g., Olson et al., 2000). Open platform ecosystems have addressed collaboration with a combination of a culture favoring openness and a collection of internet-based tools that foster communication and coordination around the product (Moon et al., 2000).

The main characteristic of an open source project is the availability of source code to all project participants (Raymond, 2001). Collaboration tools are consequently necessary to serve existing coordination needs of distributed team work (Crowston, & Howison, 2005), as well as to support users for providing feedback, and submitting bug reports, and usability concerns (Scacchi et al., 2006). It is even hypothesized that only the availability of collaborative tools has enabled the development of open source software (Robbins, 2007). However, recent research has shown, based on an empirical study including 80 projects hosted by SourceForge[56], that the success of a development project is related to the application of a version control system and the usage of mailing lists (Koch, 2009; Michlmayr, 2005). Hence all open source ecosystems, including those in our multiple case study, provide at least a minimal set of collaboration tools including a version control system, an issue tracking (also called bug tracking or change management) system, a mailing list, and a chat tool. But these tools provide only low-level coordination in terms *"retaining*

[56] SourceForge is a web-based source code repository that provides various tools for managing open source software projects. More information can be found at: http://sourceforge.net/.

project history, tracking problems and revisions, providing and controlling remote access, and preventing change collisions (the 'lost update problem')" (Fielding, & Kaiser, 1997).

It might be useful to start with a fairly simply set of tools and to increase the range of available tools when needed. Starting with many different tools might be too difficult to handle for many potential project participants because the effort to learn specific tools might be too high. On the other hand, available tools often addresses a specific need of a group of project participants. For example in the GNOME ecosystem, in order to support a specific aspect of the software development process a build tool and a graphical debugger are provided.

Besides the provision of collaboration tools, finding the "right" fit between project size and information technology support is essential. For example, within the VistA ecosystem several mailing lists exist (cf. Appendix F). The majority of these lists are only rarely used and participation might be endangered because the target group of each list is often not apparent. The hardhats list is the mailing list with the highest activity but from time to time answers are refused because questions do not meet the purpose of the list. At the same time, the hardhat lists presents the central place for asking questions in the community, therefore restricting it to a specific set of questions seems to be artificial, especially by taking the recent size of the community into account. The composition of the offered set of collaboration tools should meet the recent requirements of the community.

Often, projects start just with one mailing list, mostly a developer mailing list, and when the necessity emerges because more and more questions are related to user specific need, a separate user mailing list is established. However, in (Cubranic, & Booth, 1999) the challenge of using mailing lists as primary communication medium are emphasized and the difficulties of information overload are discussed.

Shared values and norms have been shown to foster trust that is essential for collaboration and help to prevent fragmentation of the community through such means as social disapproval of forking (Stewart et al., 2006). Commercially owned platforms, on the other hand, seem to be based on a combination of trust and coercion, which are often functionally equivalent from an external point of view (Perrons, 2009). While open source projects can differ on the details of how people interact, and vary to some degree on level of politeness, helpfulness, and treatment of newcomers, there seems to be a common core of values and norms (Stewart et al., 2006):

- People should act for the good of the community.
- People should get credit for their contributions.
- Sharing information is important.
- Helping others is important.
- Voluntary cooperation is important.
- Reputation garnered from contributions is important.

These values and norms form a key component of the collaborative infrastructure, as they guide behavior and shape how the collaboration tools are used. They give rise to specific expectations, for example, that all substantial changes will be discussed openly before they are made.

But these values, norms, and behaviors can cause significant tension for governmental and corporate participants. Open communication threatens to signal business strategy, to the firm's potential detriment. Software engineers in commercial firms are unaccustomed to exposing their code publically and potentially receiving public critiques of it. It takes effort to summarize the results of meetings and internal discussions and conversations for the community mailing list to keep external colleagues informed. Work that is highest priority for the community may not be the work that is highest priority for a given firm, causing conflicts over how resources are deployed.

Many of the main choices here have to do with how VA and corporate participants will communicate and collaborate with the rest of the ecosystem, and whether they are willing to adjust or change their cultures to make them more compatible with an open ecosystem.

The VistA ecosystem, were it to grow to occupy a substantial niche in the HIT marketplace, would have a scale and complexity that greatly exceeds the other ecosystems we studied. Especially continuously changing regulations and standards have to be considered by every ecosystem participant, but they might especially be important for the software development itself. It seems therefore likely that the collaboration technologies of today would be inadequate for operating at such a scale. Moving in the open platform direction should be accompanied with research on large-scale, open software development environments that explore new mechanisms for awareness, communication, notification, and social computing and other approaches to address these needs.

3.4.1 *The key choices are:*

- Will all participants agree on and use a common set of tools, including a hosting service, mailing lists, source control, change management, etc.?
- Will all participants (including VA and participating firms) agree to openly discuss important technical decisions in advance in agreed-upon public forums?
- Will all participants agree to take on the values and follow the norms appropriate for open source, even when this causes some friction with organizational values and norms?

3.5 Governance

Assuming that a foundation (or other non-profit) owns the intellectual property, the governance regime determines which decisions belong to the various decision-makers, and what processes are used to make them. An important starting point within the process of specifying such a governance regime is the legal organization structure. In (O'Mahony, 2005) different organizations and their legal status are reviewed and the importance for project stability and survivability are highlighted. The GNOME ecosystem for example, started without a formal organization, every developer had a vote and decisions were discussed in the whole community. But because of the ongoing growth of the project, decision-making became more complex and less transparent. Also the interest of companies to participate in the project increased, hence a more formal organization was needed, amongst other things to ensure that the project's principles were considered in all future decisions.

Another aspect to support survivability of an ecosystem is trust, and one guarantor for trust is transparency in the decision-making process. The reviewed ecosystems showed some similarities and differences. For example, in GNOME, direct influence on Board decisions by community members is possible by submitting a referendum. Another important control is that not more than 40% of the Board can consist of employees of a single entity, in order to ensure the independence of the decision-making process. By defining decision-making processes, two main cases should be considered: the consent case and the conflict resolution case (Jensen, & Scacchi, 2010). For example in Apache, if a community members gives a negative vote, an alternative proposal or a detailed explanation of reasons for this negative vote must be provided. Following such a negative vote the process of "consensus gathering" takes place. Often, a simple majority is sufficient for decisions, but for example in Eclipse, depending on the kind of decision, different majority requirements exist, ranging from simple to super-majority consent.

The defined organizational structure and the membership rules will determine how and to which degree the different key players in the VistA sphere are represented. This is likely to heavily influence their interest in actively engaging with the ecosystem. The Mozilla Foundation, for example, is the only foundation without members. In order to create revenue the Mozilla Foundation had to establish companies to carry out their development work. Even though the community is involved in the development of the Mozilla products, the main development work and decisions appear to be carried out by Mozilla's corporations.

The importance of finding the "right" governance regime for an ecosystem can be emphasized by the historical development of the Eclipse ecosystem. In 2001, the Eclipse consortium under the leadership of IBM (largest financial contributor) was founded consisting of eight companies. Even though the source code was open and the first release of Eclipse was very well received, the people were confused about the role of IBM, which led to a relatively low rate of adoption. In order to resolve all doubts, in 2003, the Eclipse Foundation was founded and independent staff was now paid by membership fees. This was an important step, in terms of building trust, reducing the release cycles and increasing the degree of market adoption.

Another important aspect of membership status is the relationship to companies. For example, the motivation to create the Apache Foundation was *"the welfare of its customers"*; therefore, the importance of companies for the community development is already embedded in the culture of the ecosystem. In (Dahlander, & Magnusson, 2005), approaches of how companies engage in ecosystems are reviewed, and the symbiotic, commensalistic, and parasitic approach are differentiated. First, in the symbiotic approach both roles, the role of the community and the company, are accepted and companies have high influence in the decision-making process. A representative of this approach in our multi-case study is Eclipse. Here, companies are allowed to be member of the Foundation and they have right such as a vote. Second, within the commensalistic approach companies have only few possibilities to influence the community. The GNOME community is an example for this approach. Here companies are represented in an Advisory Board, but they have no decision-making authority, only an advisory function. Third, in the parasitic approach, the company can indeed use the community resources but has no influence. The Mozilla ecosystem is the closest example for this approach because companies can only donate money but other possibilities to participate in the ecosystem (except by submitting code or donating money) do not exist.

In general, the governance regime of an ecosystem can be described at three analytical levels (Jensen et al., 2010): the micro-, the meso-, and the macro-level. On a micro-level, individual participants and their actions are related to artifacts and resources. In order to coordinate their activities around those objects a collaborative infrastructure must be provided (see Section 3.4). Individuals' roles should be defined in order to determine existing rights and obligations, such as the ability to commit code to the main distribution. The design of the roles depends on the meso-level that comprises independent projects. Project teams define the meso-level. For these teams, collaboration in terms of policies and guidelines as well as leadership and control are specified. For example in Apache, Project Management Committees are established on a meso-level. These project teams are completely responsible for all technical decisions such as the release schedule within the respective project. This is unlike GNOME, where the Foundation can influence the release schedule for each project that is included in the official distribution. However, many policies and guidelines are effective ecosystem-wide.   These generally include at least all necessary procedural guidelines for most common development tasks. Importantly, from the beginning decision-making processes regarding changes in these guidelines should be considered. Finally, on a macro-level the collaboration with other ecosystems should be shaped. For VistA, collaborating with standards bodies, for example, will be critical.

Another important design element to ensure the survivability of an ecosystem is the financial arrangement of the Foundation. In GNOME for example, necessary funding comes mostly from members of the Advisory Board, as opposed to Eclipse, here members of the Foundation pay an annual fee that depends on their membership type.

3.5.1 The key choices are:

- Will a foundation be recognized as the legitimate steward of the ecosystem? Will it have ownership of the intellectual property, or be granted the power to insist on compatible licenses?
- What kind of input will VA and the community have in the technical direction and evolution of the platform?
- Will an organizational structure include a meso-level with independent development projects, or will a foundation be the main decision-making entity?
- What will be the scope of the foundation's decision-making, and what kinds of decisions will be left to Councils or members?
- How would a foundation be funded?
- Will companies participate, for example, as members, board members, council or committee members in such a foundation? If yes, how?

4 Conclusion

This technical report presents the results of a one-year research study of the VistA ecosystem and a comparison of VistA to several long-lived, thriving open source ecosystems. The result is both a description of the current state of the VistA community outside VA and an analysis of the choices facing the community if VistA is to grow and occupy a key place in the HIT landscape. More generally, we have contributed to an understanding of design principles for open source ecosystems, and how these principles can be used to help guide ecosystem design and evolution. A part of this contribution is an analysis of the dimensions of ecosystem design we identified: technical architecture, governance regime, collaborative infrastructure, and business opportunities. Each dimension provides a different view on an ecosystem, but clearly the dimensions are closely related, and the choices on each dimension influence the choices for the others.

It is also important to highlight the limitations of this work. Our selection of ecosystems is based only on successful representatives. Studying less successful ecosystems would provide additional indicators for how and when design elements interact destructively. However, by reviewing only successful ecosystems we ensure that all design elements necessary for success are present. We leave the study of unsuccessful ecosystems – a very important topic – to future research. We are also limited by our choice of only four ecosystems of many possible choices. There may be other paths to success, and future research will no doubt uncover them. However, this study serves as a starting point to improve our understanding of the VistA community, the paths it may take, and the principles of ecosystem design.

While we touched on the topic in our discussion of collaborative infrastructure, we wish to call attention once again to the process by which decisions are made in the established ecosystems we studied. In each case, the communities are built on open debate, meritocracy, and decision-making based on consensus. They are by no means free of conflict, but they resist imposition of solutions from the outside. The nascent VistA ecosystem has a similar culture, and would be seriously damaged by decisions simply imposed upon it. We believe that making decisions the right way – in an open community-based process – is at least as important as making decisions that are correct on their merits. Incorrect decisions can be changed if the community remains intact. Community-building may be the most important activity to be undertaken.

A critical role for research. Enhancing and growing the VistA ecosystem will push the boundaries of practical experience and our current state of knowledge. For this reason it is critically important for ecosystem development to proceed in close collaboration with researchers committed to addressing the essential unknowns, anticipating issues, applying state of the art solutions, and taking on key issues that will help the ecosystem thrive over the longer term. Among the central issues research must address are the following:

- What collaborative infrastructure – tools, practices, and norms – will support an ecosystem on the scale and with the interconnectedness that VistA could achieve?
- What delivery models make sense for VistA – cloud deployment? Providing applications as a service? Providing the platform as a service?
- How can the very large-scale VistA legacy system be evolved into a desired architecture?
- For individual and corporate participants, what is the business case for contributing to the creation and maintenance of a platform and a fully-functional free distribution? How can it be modeled in convincing fashion so participants will have a sound basis for their decisions about participation and contribution?
- How can the critical performance, security, availability, and other quality attributes of deployed systems be assured?
- How can open source governance mechanisms – foundations, projects, councils, committees, user groups, vendors, service providers, health professionals, standards bodies – be orchestrated to provide effective policies, dispute resolution, and guidance for an ecosystem of unprecedented scale and complexity? Will novel mechanisms be needed?
- In order to guide overall decision-making, how can the socio-technical ecosystem be modeled so that the effects of establishing or changing various design parameters can be predicted, and the space of parameter combinations explored?

Research on these issues, critical for near- and long-term success, should be adequately funded, and mechanisms provided for frequent interaction between research communities and thought leaders in the ecosystem.

VistA approaches a critical juncture. VistA might well suffer the fate of Unix, and continue to fragment into multiple and somewhat incompatible versions, each with its own small community. This is not such a terrible fate – except by comparison to what VistA could become. Imagine a free platform and basic applications that could be used by doctors' offices, clinics, small hospitals, and medical centers. This platform is highly reliable, secure, and standard-compliant, and is naturally adopted by a substantial segment of the HIT market. Service providers of various sizes and specialties stand ready to install, configure, customize, and train. Vendors supply upmarket versions of the applications to large hospitals and hospital networks. All the participants, including VA, benefit from the huge market created by adoption of the free version, and by sharing the cost of maintaining the platform and basic applications. HIT costs drop, since duplication of effort is avoided, and each firm can focus on its differentiating competencies. Innovative companies, large or small, can sell their novel products and services with low barriers to entry because of the openness of the ecosystem. Such an ecosystem would not meet all HIT needs, and proprietary solutions would thrive alongside the VistA-based ones, just as now happens with Apache, Eclipse, GNOME, and Mozilla. Yet VistA would be at the center of the HIT agenda, driving down costs through competitive pressure, unleashing innovation, providing business opportunities for firms of all sizes, and making solutions available to those who otherwise could not afford them.

Acknowledgments

We wish to thank all of the interviewees who have patiently answered questions, given helpful advice, and provided careful feedback on our cases. We sincerely appreciate their time and effort. We also thank Brian Cameron, James Howison, Fabian Lopez, Ben Mehling, Mike Milinkovich, Joseph Dal Molin, Maury Pepper, Stormy Peters, Germán Póo-Caamaño, Kevin Toppenberg, Augie Turano, Ignacio Valdes, and Steve Watson for their insightful and extremely helpful comments on an earlier draft. The authors gratefully acknowledge partial funding of this effort by the Software Engineering Institute at Carnegie Mellon University.

References

APACHE, http://incubator.apache.org/learn/theapacheway.html, accessed August 29, 2010.
APACHE, Developer Information, http://www.apache.org/dev, accessed August 29, 2010.
APACHE, Licenses, http://www.apache.org/licenses/, accessed Sept. 2, 2010.
BAILETTI, T., Open source maturity curve and ecosystems interactions, http://www.slideshare.net/guest239f177/os-maturity-curve-and-ecosystem, accessed October 15, 2010.
BAKER, M., State of Mozilla and 2008 Financial Statements, http://blog.lizardwrangler.com/2009/11/19/state-of-mozilla-and-2008/, accessed Sept. 5, 2010.
BAKER, R. (2010) *VistA Community Meeting*,
BALDWIN & CLARK (2006) The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? *Management Science,* 52**,** 1116-1127.
BALDWIN, C. & WOODARD, C. (2009) The architecture of platforms: a unified view. IN GAWER, A. (Ed.) *Platforms, Markets and Innovation.* Northampton, MA, Edward Elgar, pp. 19.
BALDWIN, C. Y. & CLARK, K. B. (2000) *Design Rules: The Power of Modularity,* Cambridge, MA, The MIT Press.
BLANKENHORN, D. (2009) VA now loves its VistA software. *ZDNet Healthcare*.
BOLOUR, A. (2003) Notes on the Eclipse Plug-in Architecture. *Eclipse Corner.*
BONACCORSI, A. & ROSSI, C. (2006) Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. *Knowledge, Technology & Policy,* 18**,** 40-64.
BRANDENBERGER, A. M. & NALEBUFF, B. J. (1996) *Co-opetition,* New York, Doubleday.
BREWIN, B. (2009) Industry group meets to consider upgrading VA patient record system. *nextgov*.
BROWN, S. H., LINCOLN, M. J., GROEN, P. J. & KOLODNER, R. M. (2003) VistA---U.S. Department of Veterans Affairs national-scale HIS. *International Journal of Medical Informatics,* 69**,** 135-156.
CHAUDHRY, B., WANG, J., WU, S., MAGLIONE, M., MOJICA, W., ROTH, E., MORTON, S. & SHEKELLE, P. (2006) Systematic review: impact of health information technology on quality, efficiency, and costs of medical care. *Annals of internal medicine,* 144, 742.
COLFER, L. & BALDWIN, C. Y. (2010) The Mirroring Hypothesis: Theory, Evidence and Exceptions. Harvard Business School, 10-058.
COPLIEN, J., HOFFMAN, D. & WEISS, D. (1998) Commonality and Variability in Software Engineering. *IEEE Softw.,* 15**,** 37-45.
CROWSTON, K. & HOWISON, J. (2005) The social structure of free and open source software development. *First Monday,* 10.
CUBRANIC, D. & BOOTH, K. S. (1999) Coordinating open-source software development. *Enabling Technologies: Infrastructure for Collaborative Enterprises*, 61 - 66.
DAHLANDER, L. & MAGNUSSON, M. G. (2005) Relationships between open source software companies and communities: Observations from Nordic firms. *Research Policy,* 34**,** 481-493.
DECREM, B. & CORRE, C., MOZILLA.ORG ANNOUNCES LAUNCH OF THE MOZILLA FOUNDATION TO LEAD OPEN-SOURCE BROWSER EFFORTS, {http://www-archive.mozilla.org/press/mozilla-foundation.html, accessed Sept. 1, 2010.
DEPARTMENT_OF_VETERAN'S_AFFAIRS, Mission, Vision, Core Values & Goals, http://www4.va.gov/about_va/mission.asp, accessed Sept. 1, 2010.
DRAGOI, O. A. (1999) The Conceptual Architecture of the Apache Web Server. \urlhttp://www.cs.ucsb.edu/~tve/cs290i-sp01/papers/Concept_Apache_Arch.htm,
DSS, I. (2009) DSS, Inc., Announces Open Source Version of vxVistA EHR Framework, Joins Open Health Tools Foundation. \urlhttp://www.vistaexperts.com/pdf/DSS%20vxVistA%20OHT%201-7-09.pdf,
ECLIPSE, About the Eclipse Foundation, http://www.eclipse.org/org/, accessed June 26, 2010.
ECLIPSE, Eclipse Marketplace, http://marketplace.eclipse.org/, accessed June 26, 2010.
ECLIPSE, Membership, http://www.eclipse.org/membership/exploreMembership.php, accessed June 26, 2010.
ECLIPSE, RCP FAQ, http://wiki.eclipse.org/RCP_FAQ, accessed June 24, 2010.
ECLIPSE, Rich Client Platform, http://wiki.eclipse.org/index.php/Rich_Client_Platform, accessed June 26, 2010.

ECLIPSE, Eclipse Public License (EPL) Frequently Asked Questions, http://www.eclipse.org/legal/eplfaq.php#USEINANOTHER, accessed Sept. 2, 2010.

EVANS, D. S., HAGIU, A. & SCHMALENSEE, R. (2006) *Invisible engines : how software platforms drive innovation and transform industries,* Cambridge, Mass., MIT Press.

FIELDING, R. T. (1999) Shared Leadership in the Apache Project. *Communications of the ACM,* 42**,** 42-43.

FIELDING, R. T. & KAISER, G. (1997) The Apache http project. *IEEE Internet Computing,* 1**,** 88 - 90.

FITZGERALD, B. (2006) The Transformation of Open Source Software. *MIS Quarterly,* 30**,** 587-598.

GAWER, A. (2009) *Platforms, markets, and innovation,* Northampton, MA, Edward Elgar.

GAWER, A. (2010) The organization of technological platforms. *Research in the Sociology of Organizations,* 29**,** 287-296.

GAWER, A. & CUSUMANO, M. (2008) How companies become platform leaders. *MIT Sloan management review,* 49**,** 28.

GAWER, A. & CUSUMANO, M. A. (2002) *Platform leadership : how Intel, Microsoft, and Cisco drive industry innovation,* Boston, Mass., Harvard Business School Press.

GAWER, A. & HENDERSON, R. (2007) Platform owner entry and innovation in complementary markets: Evidence from Intel. *Journal of Economics & Management Strategy,* 16**,** 1-34.

GEER, D. (2005) Eclipse Becomes the Dominant Java IDE. *IEEE Computer,* 38**,** 16-18.

GERMAN, D. M. (2002) The evolution of the GNOME Project. *Proc. of the 2nd Workshop on Open Source Software Engineering*,

GERMAN, D. M. (2004) The GNOME project: a case study of open source, global software development. *Software Process: Improvement and Practice,* 8**,** 201--215.

GERMAN, D. M. & HASSAN, A. E. (2009) License integration patterns: Addressing license mismatches in component-based development. *ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering*, 188--198.

GITHUB, http://github.com/, accessed August 29, 2010.

GLASER, B. G. & STRAUSS, A. L. (1967) *The Discovery of Grounded Theory: Strategies for Qualitative Research,* Hawthorne, N. Y., Aldine de Gruyter.

GNOME, GNOME Code of Conduct, http://live.gnome.org/CodeOfConduct, accessed Sept. 5, 2010.

GRÖNE, B., KNÖPFEL, A., KUGEL, R. & SCHMIDT, O., The Apache Modeling Project, chapter 3.3 Extending Apache: Apache Modules, http://www.fmc-modeling.org/category/projects/apache/amp/ 3_3Extending_Apache.html, accessed

GURBANI, V. K., GARVERT, A. & HERBSLEB, J. D. (2006) A case study of a corporate open source development model. *Proceeding of the 28th international conference on Software engineering,*

HAUGE, AYALA, C. & CONRADI, R. (2010) Adoption of Open Source Software in Software-Intensive Organizations-A Systematic Literature Review. *Information and Software Technology*.

HECKER, F. (1999) Setting Up Shop: The Business of Open-Source Software. *IEEE Software,* 16**,** 45-51.

HERBSLEB, J. D. & GRINTER, R. E. (1999) Splitting the Organization and Integrating the Code: Conway's Law Revisited. *21st International Conference on Software Engineering (ICSE 99).* Los Angeles, CA, ACM Press,

HILLESTAD, R., BIGELOW, J., BOWER, A., GIROSI, F., MEILI, R., SCOVILLE, R. & TAYLOR, R. (2005) Can electronic medical record systems transform health care? Potential health benefits, savings, and costs. *Health Affairs,* 24**,** 1103.

HITSP, HITSP enabling health care interoperability, accessed Sept. 1, 2010.

JANG, D. (2006) GNOME Architecture. Presentation, GNOME Korea, \urlhttp://www.slideshare.net/iolo/gnome-architecture,

JARUZELSKI, B. & PRESIDENT, V. (2007) Fad or Future? , Booz Allen & Hamilton,

JENSEN, C. & SCACCHI, W. (2010) Governance in Open Source Software Development Projects: A Comparative Multi-level Analysis. IN ÅGERFALK, P., BOLDYREFF, C., GONZÁLEZ-BARAHONA, J., MADEY, G. & NOLL, J. (Eds.) *Open Source Software: New Horizons*. Boston, Springer, pp. 130-142.

KATZ, M. & SHAPIRO, C. (1985) Network externalities, competition, and compatibility. *The American economic review,* 75**,** 424-440.

KERNER, S. M. (2009) Eclipse Shines a Light on the IDE's Future. *internetnews.com.* March 23.

KEW, N. (2007) *Apache Modules Book, The: Application Development with Apache*, Prentice Hall.

KOCH, S. (2009) Exploring the effects of SourceForge.net coordination and communication tools on the efficiency of open source projects using data envelopment analysis. *Empirical Software Engineering,* 14**,** 397 - 417

KOLODNER, R. M. & DOUGLAS, J. V. (1997) *Computerizing large integrated health networks: the VA success*, Springer.

KRISHNAMURTHY, S. (forthcoming) An analysis of open source business models. IN FELLER, J., FITZGERALD, B., HISSAM, S. & LAKHANI, K. (Eds.) *Making Sense of the Bazaar: Perspectives on Open Source and Free Software.* MIT Press, pp.

LAAT, P. (2007) Governance of open source software: state of the art. *Journal of Management and Governance,* 11, 165-177.

LEVY, S. (1984) *Hackers: Heroes of the computer revolution*, O'Reilly Media.

LJUNGBERG, J. (2000) Open source movements as a model for organising. *Eur. J. Inf. Syst.,* 9**,** 208--216.

LONGMAN, P. (2010) *Best care anywhere: Why VA health care is better than yours,* Sausalito, CA, Polipoint Press.

LYNN, L. E., HEINRICH, C. J. & HILL, C. J. (2001) *Improving governance : a new logic for empirical research,* Washington, D.C., Georgetown University Press.

MACCORMACK, A., RUSNAK, J. & BALDWIN, C. Y. (2006) Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science,* 52**,** 1015-1030.

MARKUS, M. (2007) The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management and Governance,* 11**,** 151-163.

MCAFFER, J. & LEMIEUX, J.-M. (2005) *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java Applications,* Reading, MA, Addison-Wesley.

MEDSPHERE, Corporation, Medsphere Systems, http://www.medsphere.com, accessed Sept. 5, 2010.

MEHLING, B. (2008) How do the various technology components of OpenVista fit together? , \urlhttps://medsphere.org/docs/DOC-1326,

MESSERSCHMITT, D. & SZYPERSKI, C. (2005) *Software ecosystem: understanding an indispensable technology and industry,* Cambridge, MA, MIT Press.

MICHLMAYR, M. (2005) Software Process Maturity and the Success of Free Software Projects. *Frontiers in Artificial Intelligence and Applications*, 3-14.

MOCKUS, A., FIELDING, R. & HERBSLEB, J. D. (2002) Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology,* 11**,** 309-346.

MOCKUS, A., FIELDING, R. T. & HERBSLEB, J. (2000) A case study of open source software development: the Apache server. *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, 263--272.
MOON, J. Y. & SPROULL, L. (2000) Essence of Distributed Work: The Case of the Linux Kernel. *First Monday,* 5.
MOZILLA.ORG, Bugzilla, http://bugzilla.mozilla.org/, accessed Sept. 5, 2010.
MOZILLA.ORG, Firefox Addons, https://addons.mozilla.org/firefox/, accessed Sept. 5, 2010.
MOZILLA.ORG, History of the Mozilla Project, http://www.mozilla.org/about/history.html, accessed Sept. 5, 2010.
MOZILLA.ORG, Module Owners, http://www.mozilla.org/about/owners.html, accessed Sept. 5, 2010.
MOZILLA.ORG, Mozilla 2008 Financial FAQ, http://www.mozilla.org/foundation/documents/mozilla-2008-financial-faq.html, accessed Sept. 5, 2010.
MOZILLA.ORG, Mozilla Firefox eBay-Edition, http://www.mozilla.com/en-GB/add-ons/ebay/, accessed Sept. 5, 2010.
MOZILLA.ORG, Mozilla Foundation License Policy, http://www.mozilla.org/MPL/license-policy.html, accessed Sept. 5, 2010.
MOZILLA.ORG, Mozilla Organizations, http://www.mozilla.org/about/organizations.html, accessed Sept. 5, 2010.
MOZILLA.ORG, Our Projects, http://www.mozilla.org/projects/, accessed Sept. 5, 2010.
MOZILLA.ORG, The Mozilla Foundation, http://www.mozilla.org/foundation/, accessed Sept. 5, 2010.
MOZILLA.ORG, What Is Mozilla?, http://www.mozilla.com/en-US/about/whatismozilla.html, accessed Sept. 5, 2010.
MOZILLA.ORG, Mozilla foundation reoganization, accessed
MOZILLA.ORG, Interview with Mike Schnoepfer, http://mozillamemory.org/detailview.php?id=7475, accessed Sept. 5, 2010.
NETCRAFT, August 2010 Web Server Survey, http://www.netcraft.com/survey, accessed Sept. 15, 2010.
NETSCAPE, Netscape Announces Plans to Make Next-Generation Communicator Source Code Available Free on the Net, http://web.archive.org/web/20021001071727/wp.netscape.com/newsref/pr/newsrelease558.html, accessed Sept. 5, 2010.
NOLL, J. (2009) What Constitutes Open Source? A Study of the Vista Electronic Medical Record Software. *Open Source Ecosystems: Diverse Communities Interacting*, 310-319.
NORTHROP, L., ET AL (2006) *Ultra-Large-Scale Systems The Software Challenge of the Future,* Pittsburgh, Software Engineering Institute.
NYMAN, K. (2005) Eclipse Architecture Council: Proposal about the updated Eclipse architecture introduction pictures.
O'MAHONY, S. (2005) Nonprofit Foundations and Their Role in Community-Firm Software Collaboration. IN FELLER, J., FITZGERALD, B., HISSAM, S. A. & LAKHANI, K. R. (Eds.) *Perspectives on Free and Open Source Software.* Cambridge, MA, MIT Press, pp. 393-414.
O'MAHONY, S. & FERRARO, F. (2007) The emergence of governance in an open source community. *Academy of Management Journal,* 50**,** 1079-1106.
O'MAHONY, S. (2007) The governance of open source initiatives: what does it mean to be community managed? *Journal of Management and Governance,* 11**,** 139-150.
OF ENTERPRISE DEVELOPMENT, O. (2009) VistA-HealtheVet Monograph. Department of Veterans Affairs,
OHTF, Vision, http://www.openhealthtools.org/about/vision, accessed October 15, 2010.
OLSON, G. M. & OLSON, J. S. (2000) Distance Matters. *Human-Computer Interaction,* 15**,** 139-178.
OLSON, M. (1971) *The logic of collective action: Public goods and the theory of groups*, Harvard Univ Pr.
OPEN_HEALTH_TOOLS_FOUNDATION (2010) Open Health Tools Overview. Open Health Tools,
PERRONS, R. (2009) The open kimono: How Intel balances trust and power to maintain platform leadership. *Research Policy,* 38**,** 1300-1312.
PHILLIPS, P., Why Mozilla Matters, http://www-archive.mozilla.org/why-mozilla-matters.html, accessed Sept. 5, 2010.
PROJECT, T. G. (2010) GNOME Foundation Referenda. \urlhttp://foundation.gnome.org/referenda/,
PROJECT., T. G. (2010) How to propose modules for inclusion in GNOME. \urlhttp://live.gnome.org/ReleasePlanning/ModuleProposing,
RAYMOND, E. S. (2001) *The Cathedral and the Bazaar,* Sebastopol, Cal., O'Reilly.
ROBBINS, J. E. (2007) Adopting Open source Software Engineering (OSSE) Practices by Adopting OSSE Tools. IN J. FELLER, B. F., S. HISSAM & K. LAKHANI (Ed.) *Making Sense of the Bazaar: Perspectives on Open Source and Free Software.* Sebastopol, CA, O'Reilly & Associates, pp. 245-264.
SCACCHI, W., FELLER, J., FITZGERALD, B., HISSAM, S. A. & LAKHANI, K. (2006) Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice,* 11**,** 95-105.
SCHROEPFER, M., Mozilla platform, http://blog.mozilla.com/schrep/2007/05/16/mozilla-platform/, accessed Sept. 1, 2010.
SMETHURST, G. (2010) Changing the In-Vehicle Infotainment Landscape. GENIVI Allliance,
SOGHOIAN, C., A dangerous conflict of interest between Firefox and Google, http://news.cnet.com/8301-13739_3-9776759-46.html, accessed Sept. 1, 2010.
STEWART, K. J. & GOSAIN, S. (2006) The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *MIS Quarterly,* 30**,** 291-314.
THE_APACHE_SOFTWARE_FOUNDATION, Apache Module Index, httpd.apache.org/docs/2.2/mod/, accessed Sept. 1, 2010.
THE_APACHE_SOFTWARE_FOUNDATION, How the ASF works., http://www.apache.org/foundation/how-it-works.html, accessed Sept. 1, 2010.
TORVALDS, L. (1999) The linux edge. *Communications of the ACM,* 42**,** 38-39.
TROTTER, F. (2007) What is VistA Really. \urlhttp://vistapedia.net/index.php?title=What_is_VistA_Really,
VALDES, I. (2008) Free and Open Source Software in Healthcare 1.0. American Medical Informatics Association, Open Source Working Group,
VETERANS HEALTH ADMINISTRATION, O. O. I. (2006) Winner of the 2006 Innovations in American Government Award. Department of Veterans Affairs - VistA,
VISTA_MODERNIZATION_WORKING_GROUP (2010) VistA Modernization Report - Legacy to Leadership. Fairfax, VA, American Council for Technology-Industry Advisory Council,
WAGSTROM, P., HERBSLEB, J. & CARLEY, K. (2005) A Social Network Approach to Free/Open Source Software Simulation. *The First International Conference on Open Source Systems*,
WAGSTROM, P. A. (2009) Vertical Interaction in Open Software Engineering Communities. *Carnegie Institute of Technology and School of Computer Science.* Pittsburgh, Carnegie Mellon University,
WEBER, S. (2004) *The success of open source*, Harvard Univ Pr.
WEST, J. (2003) How open is open enough? Melding proprietary and open source platform strategies. *Research Policy,* 32**,** 1259-1285.

WEST, J. & O MAHONY, S. (2008a) The role of participation architecture in growing sponsored open source communities. *Industry & Innovation,* 15**,** 145-168.

WEST, J. & O'MAHONY, S. (2005) Contrasting Community Building in Sponsored and Community Founded Open Source Projects. *Hawaii International Conference on System Sciences,* 7**,** 196c.

WEST, J. & O'MAHONY, S. (2008b) The Role of Participation Architecture in Growing Sponsored Open Source Communities. *Industry and Innovation,* 15**,** 145-168.

WORLDVISTA, Adoption of Restated Bylaws of WorldVistA, http://worldvista.org/WorldVistA/WV_Bylaws_Restated_06-13-09.pdf/view, accessed October 15, 2010.

YELLOWLEES, P. M., MARKS, S. L., HOGARTH, M. & TURNER, S. (2008) Standards-Based, Open-Source Electronic Health Record Systems: A Desirable Future for the U.S. Health Industry. *Telemedicine and e-Health,* 14**,** 284-288.

YIN, R. K. (2009) *Case Study Research - Design and Methods,* Thousand Oaks, SAGE Publications.

Appendix A: Dimensions of platform-based ecosystems

Our studies of these ecosystems is based on our own prior research (Gurbani, Garvert, & Herbsleb, 2006; Mockus, Fielding, & Herbsleb, 2002; Mockus, Fielding, & Herbsleb, 2000; Wagstrom, Herbsleb, & Carley, 2005; Wagstrom, 2009) and existing studies in this field. We adopt an analytic framework consisting of four dimensions that prior research suggests are critical to ecosystem design: the technical architecture, the governance regime, collaborative infrastructure and culture, and business opportunities. We explain each dimension of our framework in the remainder of this section.

A.1 Technical Architecture

At the heart of any ecosystem is a technical platform that supports the accumulation of the efforts of diverse contributors (Evans et al., 2006; Gawer, 2009; Gawer et al., 2002). The term *"platform"* can be a little tricky in this context. It can refer to an actual deployable implementation or simply specifications that will enable interoperability (see, e.g., Baldwin et al., 2009). Today's mobile platforms such as iPhone and Android, are implementations that find their way into customers' hands, and on top of which other ecosystem participants can deploy applications. The Genivi platform (Smethurst, 2010), on the other hand, is an example of a *"reference"* platform, consisting of middleware and reference applications. Vendors using the platform create their own applications, which will be certified as Genivi-compliant. In this case, the critical platform supporting the ecosystem is the specification, with the implementations primarily for reference, i.e., showing in detail how to build a conforming application. In our case studies, we gave this broader interpretation to *"platform"*, looking for technical characteristics that allowed participants to make use of the shared technical work.

Of particular concern in the role of supporting and ecosystem is the modularity of the platform (e.g., MacCormack, Rusnak, & Baldwin, 2006). By modularity, we mean that development work on different modules can be undertaken relatively independently (Baldwin, & Clark, 2000). It has long been held that modular design is critical for allowing multiple teams to work relatively independently of one another, without being overwhelmed by the need for technical coordination (Colfer, & Baldwin, 2010). This need is even greater when the ability of teams to coordinate their work is reduced by distance (Herbsleb, & Grinter, 1999; Olson et al., 2000), as is inevitably the case in ecosystems.

In addition to the overall requirement of modularity, the openness of the platform is a critical feature. For privately owned platforms, the degree of openness involves a crucial tradeoff between adoption (fostered by openness) and appropriability (made more difficult by openness) (West, 2003). Similar considerations are important even when members of a community or a foundation own the platform, if it includes a usable (not just a reference) implementation. The more complete the free version of the platform is, the more adoption one would expect. Yet greater completeness suggests that fewer business opportunities will be available for ecosystem participants. If a free platform includes a reasonably good billing system, for example, it will be harder for a vendor to sell entry-level billing systems, or any billing system that is inconsistent (e.g., has a different user interface or makes different process assumptions) with the free version. On the other hand, if the overall market penetration of the free version is sufficiently increased by virtue of its completeness, vendors may have a much greater opportunity to sell higher-margin enhanced and custom versions, as well as services such as training and installation. Decisions about what is included and what is not, what functionality is exposed via Application Programmer's Interfaces (APIs) and what is not, and the simplicity or complexity of the interfaces will all have powerful impacts on shaping what participants can do with the platform and how cost-effectively they can do it.

A.2 Governance

A governance regime is the way in which a particular ecosystem distributes types of decisions over types of participants and decision-making mechanisms. A useful definition from the literature is the following:

"OSS governance can be defined as the means of achieving the direction, control, and coordination of wholly or partially autonomous individuals and organizations on behalf of an OSS development project to which they jointly contribute." (Lynn, Heinrich, & Hill, 2001; Markus, 2007)

An effective yet open governance regime is critical for the success of an ecosystem. They are made up of multiple organizations, yet they have neither markets nor hierarchies to coordinate their activities (O'Mahony, & Ferraro, 2007). There are many decisions that have a very broad and profound impact on members, and must generally be made for the entire ecosystem, such as

- What will be included in a free distribution?
- How the platform software will be licensed?
- What standards will be adopted?
- How quality will be assured?
- The terms of the platform license?
- How the architecture will evolve?
- What is the platform release schedule?
- What technologies will be used in construction of the platform?
- What people and organizations are granted membership?

Despite this need for agreement on key decisions, too much central control threatens one of the primary benefits of open platforms – the ability of members to freely and independently innovate. Governance structures seem to arrive at a balance of bureaucratic and democratic mechanisms (O'Mahony et al., 2007).

Governance regimes do not appear full-blown, but rather evolve. The development stages of open source software projects correspond roughly to three types of OSS governance: 'spontaneous' governance, internal governance, and governance towards outside parties (Laat, 2007). In its first years, an open source project exhibits no explicit or formal coordination or control. Only the applied license defines a kind of frame in which members of the open source project can act. Often accompanied by the growth of an open source project more coordination is needed and explicit coordination and control mechanisms are introduced for the internal governance. Mechanisms are for example, modularization, division of roles, and delegation of decision-making. At the same time, open source projects have to find better means of dealing with outside parties, such as companies and organizations and they need an institutional frame, often in form of a non-profit foundation. The rights and the responsibilities of the project's members, and the right to participate in decision-making depend on each open source project. For example, West and O'Mahony (West et al., 2005) argue that in contrast to community-managed projects, in sponsored open source projects, sponsors try to retain direct or indirect control.

As we mentioned, it seems unlikely that any single private firm will come to own the VistA platform, so we focus on community-managed rather than commercially sponsored ecosystems. Participants in community-managed projects associate five particular features with this governance style (O'Mahony, 2007):

- Independence (Control over the community is independent of any one sponsor but rests with the members of the community itself),
- Pluralism (preserves multiple and perhaps competing approaches, methods, theories or points of view),
- Representation (contributing members can be represented in community-wide decisions),
- Decentralized decision-making (some degree of decision-making is decentralized), and
- Autonomous participation (welcomes participation and allows members to contribute on their own terms).

By describing the governance of each of our representatives we particularly address the following areas: governance structure, governance processes (project management, release management), and intellectual property. Within the governance structure the main governing bodies are introduced and their main responsibilities are explained (West, & O mahony, 2008a). The objective is to show the roles of foundations, members, committees, firms, and individuals, and how decisions are reached.

A.3 Collaborative infrastructure and culture

Successful collaboration over distances and across organizational boundaries is widely recognized as a very difficult problem (e.g., Olson et al., 2000). Open platform ecosystems have addressed collaboration with a combination of a culture favoring openness and a collection of internet-based tools that foster communication and coordination around the product (Moon et al., 2000).

Contemporary ecosystem culture can trace its origins to the "hacker" culture prevalent in the early days of computing, when pioneers at major universities and corporate research labs freely shared technical solutions in the form of code (Levy, 1984; Weber, 2004). Open source culture continues to value sharing, helping, and valuing one's reputation in the community (Stewart et al., 2006). Moreover, shared values and norms have been shown to foster trust that is essential for collaboration and help to prevent fragmentation of the community through such means as social disapproval of forking (Stewart et al., 2006).

As one can observe on any open source hosting service, such as SourceForge, Savannah, or GNOME, most project use a common set of tools that include a version control system (such as the Concurrent Versioning System or Subversion), a bug or task tracking system (such as Bugzilla), and a variety of mailing lists for users and developers. Some projects make use of other tools, such as wikis, chat rooms, and social computing style technologies (for an example of the latter, see GitHub).

Since the tools used are generally fairly pedestrian, and widespread in software development more generally, what is of greatest interest is the norms, beliefs, and values about what should be communicated, how, and when, as well as how the tools should be used to manage the work. The tools and the collaborative culture together create the essential infrastructure that allows ecosystems to function in a coherent and coordinated way within the governance framework.

A.4 Business opportunities

The final dimension we use to describe the ecosystems is the set of business opportunities that are actively being pursued in the ecosystem. Research literature has identified a number of ways of creating and appropriating valued in an ecosystem context. Krishnamurthy (Krishnamurthy, forthcoming) distinguishes three:

- Distributor, who provides the platform, perhaps in a bundle with additional software and services,
- Software producer, who uses the platform in its own products, and
- Third party service provider, who offers support services.

Based on a review of the research literature, Bonaccorsi and Rossi (Bonaccorsi, & Rossi, 2006) describe a number of economic and technical motivations for firms to participate in open source ecosystems, including

- Independence from price and licensing policies of large software firms,
- Making profit from complementary services,
- Selling related products,
- Exploiting R&D activity of the open source community,
- Hiring technical personnel,
- Exploiting feedback and contributions from the user community,
- Promoting standardization, and
- Addressing security issues.

Similar sets of motivations have been described by Hecker (Hecker, 1999) and Hauge and colleagues (Hauge, Ayala, & Conradi, 2010).

Figure A-I shows a high-level view of business models, each of which is briefly highlighted in the following paragraphs.

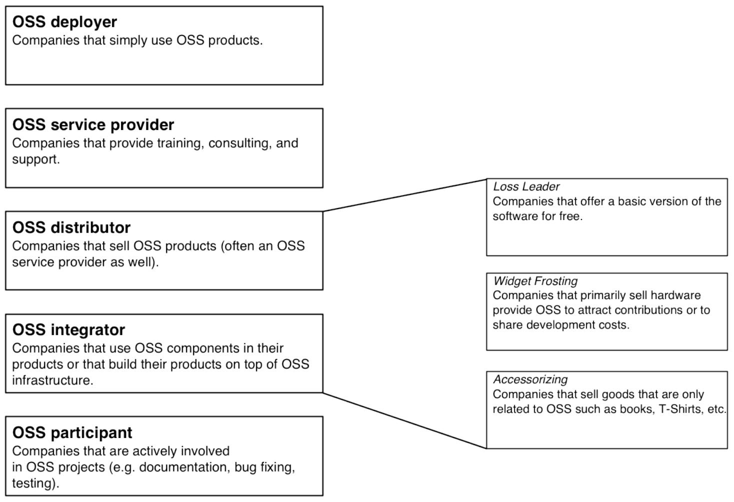

OSS deployer
Companies that simply use OSS products.
OSS service provider
Companies that provide training, consulting, and support.
OSS distributor
Companies that sell OSS products (often an OSS service provider as well).
OSS integrator
Companies that use OSS components in their products or that build their products on top of OSS infrastructure.
OSS participant
Companies that are actively involved in OSS projects (e.g. documentation, bug fixing, testing).
Loss Leader
Companies that offer a basic version of the software for free.
Widget Frosting
Companies that primarily sell hardware provide OSS to attract contributions or to share development costs.
Accessorizing
Companies that sell goods that are only related to OSS such as books, T-Shirts, etc.

Figure A-I. Overview about defined roles of companies.

There are many companies that only use the software (OSS deployer). These companies exist in nearly all successful ecosystems, and are not further considered in our analysis. Service providers are companies that provide more or less any kind of service that is related to an open source software product. Distributors sell OSS products, whereas integrators combine OSS components with their software or build systems for their clients using at least some OSS components.

Hecker (Hecker, 1999) elaborates on the distributor and integrator roles, differentiating them into loss leader, widget frosting, and accessorizing. Often, a basic version is provided for free to the user whereas the full version is commercial software. This is called loss leader. One objective is to reduce existing barriers for users for buying software because the basic version allows users to test the software and to gather experiences in its usage. In order to have additional functionality the user has to buy the complete version but then, the user might be already convinced about the usefulness of the software. Another possibility is the dual license strategy. Here, a software product is offered with an open source software license but the same software is provided by a commercial license as well in order to allow third parties to integrate this software in existing commercial solutions. Hardware producers that use existing open source to sell besides the hardware software that is pre-installed on their machines primarily apply widget frosting. The last approach can be seen as a complementary. Often open source usage and development involves a philosophy and users or developers want to show this mindset to others. Some companies capitalize on this by selling t-shirts, mugs, and books to leverage the open source brand (Fitzgerald, 2006).

Appendix B: Four case studies of established ecosystems

We employed a multiple case study design and selected four ecosystems - Apache, Eclipse, GNOME, and Mozilla - using the following criteria:

- Large scale systems,
- Relatively mature technology,
- Multiple projects or modules,
- Existence of a Foundation, and
- Both private engagement and corporate participation.

Clearly, this small sample does not exhaust the full range of platform-based ecosystems. We chose the first three attributes (large, mature, multiple projects) for their obvious relevance to VistA. While VistA does not currently have a single foundation, the open distributions that exist all have a foundation or non-profit at their hub. Moreover, it seems highly unlikely that a single corporate entity will end up with ownership of VistA, so we ruled out commercially-owned platforms. Finally, while it is possible that VA will decide to retain control of VistA and act as the open source hub, we are not aware of a precedent for a government-owned open platform. In any event, it seems that government stewardship would likely resemble foundation stewardship in that control would be exercised on behalf of the larger community, not for profit-seeking motives. Finally, we particularly wanted to examine ecosystems with strong corporate participation in order to better understand the business incentives for participation.

B.1 Apache

Apache is a conglomerate of various open source projects. Each of these projects deals with one issue in the area of "web serving". In February 1995, the Apache project was started in a joined effort by a group of volunteers ("Apache Group") to improve the existing NCSA httpd program (Mockus, Fielding, & Herbsleb, 2002). In January 1996, the first version of the Apache httpd 1.0 was released. By 1999, Apache was already the most often used web server in the world[57]. Because of this success, the technical as well as economic interest has grown and new *"sister projects"* (e.g., Ant) were founded. At present, the Apache Foundation comprises about 200 projects.

The heart of the Apache Foundation is the community management. Even though there is no main product or platform that integrates the different development efforts, the whole community is kept together by *"The Apache Way"* (Apache) that consists of the following principles:

- Collaborative software development,
- Commercial-friendly standard license,
- Consistently high quality software,
- Respectful, honest, technical-based interaction,
- Faithful implementation of standards, and
- Security as a mandatory feature.

Another important principle is meritocracy. Individual contributions are a prerequisite for being accepted by other community members and to become finally a member of the Apache community.

[57] According to the Netcraft survey, more information at http://www.netcraft.com/survey.

B.1.1 Technical architecture

Everything started with the development of the Apache HTTP Server, but today the Apache project comprises not less successful projects such as the network server (servlet container) Apache Tomcat, the search engine Apache Lucene, as well as the build management and project management software Apache Maven. All together there are 174 projects and 88 are top-level projects. Apache has no common platform but projects are encouraged to use open standards.

The principle nature of the different Apache projects can be interpreted as a Lego system with predefined forms and sizes for the knobs (presenting the standards). Each project presents a Lego block and by using the knobs blocks can be assembled in order to create customized solutions. It is even possible, because of the Apache license, to use external and proprietary blocks. Following, this principle is illustrated using various examples.

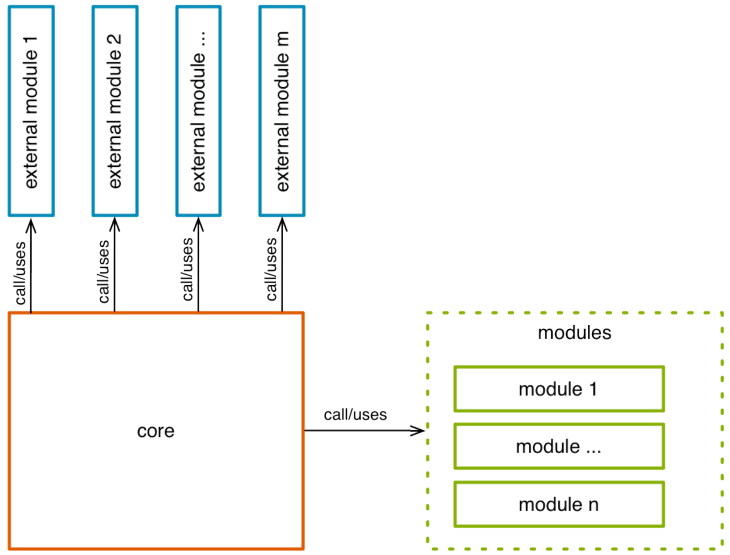


Figure B-I. Apache HTTP Server high-level architecture (following Dragoi, 1999).

The Apache HTTP Server has a modular architecture that comprises a core component and a module component (cp. Figure B-I). The relatively small core implements the basic functionality of a web server whereas the module component includes a set of modules that extend and complement the core (Kew, 2007). The Apache HTTP Server distribution consists of nine core features and multi-processing modules that are part of the core and 75 other modules (The_Apache_Software_Foundation, 2009). The server provides a module API that enables to extend easily the core functionality by other functions using external modules (Dragoi, 1999). This plug-in architecture has led to more than 400 released plug-ins by people outside the Apache project. The Apache Modeling Portal (Gröne, Knöpfel, Kugel, & Schmidt, 2004) shows the Apache module structure and the interaction of core and modules in a plain format.

The same principles have been applied to other Apache projects such as Apache Forrest. It is a publishing framework that converts different input formats into one unified presentation and provides this presentation in different output formats. It is mainly based on Apache Cocoon, a framework that separates content from presentation and uses, for example, Apache FOP

(Formatting Objects Processor) for exporting HTML and PDF documents or Apache Batik for exporting images.

Many of the projects that are part of the ASF are using open standards such DOM, SOAP, XML Scheme from W3C. For example the Apache FOP follows the Adobe PDF 1.4, PS, Microsoft RTF 1.6, XSL 1.0, XSL 1.1 standards. If other projects need an implementation of one of these standards, they can use FOP. For example, the Apache XML Graphics Commons uses Apache FOP. This library only consists of reusable components from the Apache Batik and FOP projects.

Often, projects have emerged within existing projects, providing a solution for a specific problem. Over time, the size and importance of this project has grown because the solution appeared to be applicable in other projects as well. Such a development can even end with extracting this project from the mother project and creating a new, independent project.

The distinctive feature of Apache projects is that there can be used as stand-alone applications but at the same time as combined solutions.

B.1.2 Governance

From the beginning, the members of the Apache team were very concerned about their decision-making processes. The *"Apache Group"* (later Apache Foundation), therefore, agreed early to a specific voting system that required a minimal subset of members to vote (Fielding, 1999). The main motivation behind it was the need to ensure the action-ability of the development.

Basically, the types of votes are based on the specific action; for example, if it is a procedural decision, a package release decision, or a code modification. The former two are mostly based on majority approval, whereas the latter is called *"lazy consensus"* approval because only a few positive votes with no negative vote are enough to push a project through to its nascent stages. In the case of a negative vote, an explanation should be included that contains the reasons that led to this negative vote. Based on the process of *"consensus gathering"* the community tries to address the concerns mentioned in the explanation and negotiates a solution that satisfy every member.

However, the Apache Server became very successful and more projects were founded that were related to the web server area. In June 1999, the Apache Foundation was founded because a more structured organization became necessary (West et al., 2005). The entities of the Foundation are its members, the Board of Directors (Board), and Project Management Committees (PMC).

The Foundation was founded to (The Apache Software Foundation, 2010):

- Provide a foundation for open, collaborative software development projects by supplying hardware, communication, and business infrastructure,
- Create an independent legal entity to which companies and individuals can donate resources and be assured that those resources will be used for the public benefit,
- Provide a means for individual volunteers to be sheltered from legal suits directed at the Foundation's projects, and
- Protect the "Apache" brand, as applied to its software products, from being abused by other organizations.

The Apache governance model is clearly divided into two areas: one area comprises the organizational, legal and financial responsibilities represented by the Board and its officers; while the other area comprises technical and operational responsibilities that are decentralized in PMCs.

Membership. Members of the community can be entities or individuals. In May 2010, the Foundation has 332 members (only individuals). A recent member can nominate a new member of the Foundation. Then, she/he can be elected because of her/his contributions to the Foundation. It is important that these activities should not be only concentrated on one project, but should have an impact on a number of different projects. Besides the election of the Board, members of the Foundation can propose committers for membership and new projects for incubation. There is no membership fee for members of the Foundation.

Board of Directors. The Board governs the Foundation and is especially responsible for corporate assets such as funds and intellectual property, and it allocates corporate resources such as technical infrastructure to projects. The members of the Foundation annually elect the Board that consists of nine directors. The Board of Directors defines certain quality standards for example for the release process or the project incubation. Besides the appointment of PMC chairs, the Board initiates a couple of cross-functional projects, such as Infrastructure PMC, Incubator PMC, Legal Affairs PMC.

Project Management Committees. PMCs have technical decision-making authority in their projects and they ensure that procedures are implemented, that legal issues are considered, and that the whole community participates in the release process. The Board appoints the chair of the PMC, who is an officer (vice president) of the ASF, and officers, who are responsible for day-to-day activities. Members of the PMC are developers or committers (developers with commit rights).

Project Management. All software development activities are carried out in separate autonomous projects that are managed by the aforementioned Project Management Committees. It is founded by resolution of the Board. A PMC is responsible to manage one or more software projects, which are again created by resolution of the Board. The following main roles can be differentiated: user, developer, and committer. Whereas a user just uses the software, a developer contributes to the project, for example, by providing patches or writing documentation. Based on her/his contributions, a developer can be proposed as a committer by a PMC member. A prerequisite of this application is a signed Individual Contributor License Agreement (CLA), which has to be approved by the Board's secretary. The commit process itself is decentralized in each project. The PMC finally approve (code review) the commits made by committers of the project.

New projects go through an incubator period in which they are evaluated based on the probability that they will become successful members of the community (e.g., because of the diversity of committership). Each new project has to submit a proposal to the Board that contains the project vision, the chair, project procedures such as code review and release management, voting procedures and defined processes for committer access and PMC membership. Such a proposal needs to be supported by an ASF member and a sponsor (Board member, a Top Level Project (TLP) that considers the candidate to be a sub-project), or the Incubator PMC. Existing functional overlap between an incubator project and an existing project is allowed and does not influence the approval of the Board. After acceptance of the proposal the projects goes through a so-called Podling period. The Incubation PMC takes care of the project during that period. The following activities have to be carried out: the reporting schedule, the project status page, the mailing lists, and the repository space. However, during that period projects are not full members of the Apache community and therefore, different guidelines for example for their releases apply. In order to graduate from the Podling status a project has to fulfill a number of requirements, for example the project should not rely on one main contributor, all code has to be conform to the Apache Software License, the project should use other Apache subprojects or should at least have synergetic relationships to them as well as the project should completely use the by Apache provided technical infrastructure. A project that reaches its lifetime is moved into the so-called Attic, a separate PMC.

Each project within the ASF can define their self-governing rules. A pre-predefined vision how those projects should be managed does not exist. But each project in the ASF has the following similarities:

- Communication within projects is carried out via mailing lists.
- Decision making is based on lazy consensus.
- All projects share the *"Apache way"*.
- Project members are not paid by the Foundation.
- Individuals are member of the ASF community and not institutions.
- Confidentiality and public discussion are balanced in a project.

Release Management. Even though, each project in the ASF community is responsible for its release management, the Foundation has defined requirements on reach release and the release process. These requirements include besides other things: each release must contain a source package, cryptographically signed by the Release Manager, must be tested and must comply with ASF licensing policy. Each current release can be assessed by one central website (Apache).

Sponsorship Program. A Sponsorship Program allows non-directed monetary contributions. There are four different sponsorship levels defined with certain donations and in-return benefits: Platinum Sponsorship, Gold Sponsorship, Silver Sponsorship, and Bronze Sponsorship. For example, Yahoo, Microsoft, Google are platinum sponsors; they support the Apache Foundation with US $100k per year.

Intellectual Property. All projects of the Apache Foundation are applying the Apache License Version 2.0 (Apache, 2010), which is not a copy-left license. This license therefore allows developing both open source software and proprietary software based on the licensed source code.

B.1.3 Collaborative infrastructure and culture

The Apache Foundation provides for each project that is member of the Foundation certain infrastructure services. By focusing on this single aspect, the Apache Foundation provides services such as SourceForge. But utilizing these services is only one prerequisite of becoming a member of the Apache community. The aforementioned *"The Apache Way"* is one main motif of being a member in the Apache community. The Infrastructure PMC of the Apache Foundation provides a web serving environment (web sites and wikis), a code repository, a mail management environment, a bug and issue tracking system and a distribution mirroring system. Each project in the Apache community can be accessed by its own often individually designed website. The only characteristic to identify its community affiliation is the Apache logo (i.e. the feather).

The official user conference of the Apache project is the ApacheCon. This is a general conference about existing technologies and projects in the Apache community. Sometime projects host their own conferences, addressing the specific needs of their users and developers (i.e. Apache Lucene and Soir, Cocoon GetTogether).

B.1.4 Business opportunities

There are different reasons for companies to support the Apache Foundation. One of these reasons is their reliance on Apache products. IBM has significantly impacted the successful development of the Apache community. In 1999, IBM joined the Apache Foundation and supported the further development of the web server by submitting bug fixes and features. Yahoo! supports the ASF because of the Apache HTTP Server and Lucene project. Furthermore, Google even uses the infrastructure provided by the Apache Foundation and initiated new projects such as Shindig (container and backend server components for hosting OpenSocial applications). By providing code, companies ensure that existing server components from the Foundation are used from the beginning, and the product will be well integrated in the Apache *“solution map”*.

Microsoft’s motivation to sponsor the Apache Foundation slightly differs. It is a more user-driven demand for interoperability because an increasing number of Window server’s deploy Apache-based technologies on top of it.

Companies such as Facebook, LinkedIn and Hippo (CMS) are using open source technologies and components to build their own products. SpringSource (Java application infrastructure and management), Lucid Imagination (certified distributions of Lucene and Soir) provide highly specialized (certified) solutions using ASF components and offer support, training, consulting and value-added software extensions.

The most dominant role of existing business opportunities for companies in the Apache community is the OSS integrator. Because Apache projects can easily integrated in existing products or can be assembled to new solutions, companies use this opportunity to enhance their existing product or to create new ones. Contrarily, companies such as IBM and Windows invest in Apache to ensure that their products stay interoperable with Apache projects.

B.2 Eclipse

According to the *“about”* page at eclipse.org (Eclipse), *“Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle.”* The history of this community is described as follows:

“The Eclipse Project was originally created by IBM in November 2001 and supported by a consortium of software vendors. The Eclipse Foundation was created in January 2004 as an independent not-for-profit corporation to act as the steward of the Eclipse community. The independent not-for-profit corporation was created to allow a vendor neutral and open, transparent community to be established around Eclipse. Today, the Eclipse community consists of individuals and organizations from a cross section of the software industry.” (Eclipse)

The Eclipse ecosystem evolved into its current form in response to problems encountered, many of which concerned intellectual property arrangements and the role of IBM (see (Wagstrom, 2009) ch. 2.1 for a brief history). The success of these arrangements has been such that Eclipse is widely considered to be the dominant development environment for Java (Geer, 2005).

Today, the Eclipse Foundation website lists 160 members (Eclipse), ranging from large multinational corporations such as AT&T, IBM, Google, Intel, Motorola, Nokia, Oracle, Bosch, and Siemens to a wide variety of small vendors and service providers, and a few research labs and universities. The Eclipse technology continues to evolve rapidly. New directions include *“bringing Eclipse to the web”* and expanding runtime capabilities, particularly in the direction of Service-Oriented Architecture (SOA) applications (Kerner, 2009).

B.2.1 Technical architecture

The Figure B-II illustrates several key elements of the Eclipse architecture. First, it shows that Eclipse is built on top of the OSGI framework, a module system and service platform for Java that spans devices, clients and servers. Bundles, i.e. plug-ins, can be integrated, updated, removed, etc. at runtime within this dynamic component model. Because of this platform infrastructure, Eclipse is more widely applicable, such as in the automotive industry. However, on top of this runtime platform, the Rich Client Platform (RCP) with its core plug-ins is located. Many different kinds of applications (e.g., IDEs) can be built on top of the RCP (Bolour, 2003). The Rich Client Platform RCP) is the *"minimal set of plug-ins needed to build a rich client application"* (Eclipse). As Figure B-III illustrates, on top of the base RPC, there are additional and optional plug-ins are also available (see, e.g., (Eclipse; McAffer, & Lemieux, 2005)).

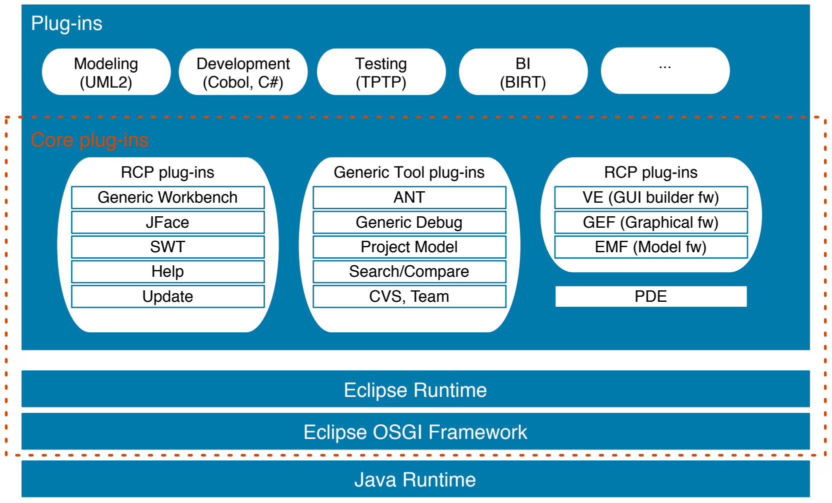


Figure B-IV. High level view of Eclipse architecture (following (Nyman, 2005))

The Eclipse platform includes additional plug-ins specific to an integrated development environment (IDE), including Java development tools and an environment for building plug-ins. The Eclipse Marketplace (Eclipse) offers over 1,000 additional plug-ins, under a variety of open source and commercial licenses, that users can install on top of these platforms.

In order to support the wide variety of plug-ins, Eclipse has a well-documented plug-in model that developers can use for extensions to the Eclipse platform or other plug-ins (Bolour, 2003). Plug-ins must be described in a specific XML format in a *"manifest"* file, that provides things like the name, id, any extension points the plug-in provides, and any extension points in other plug-ins that this plug-in will extend.

The architecture establishes conventions for communication between extended and extending plug-ins, involving schemas defined in the extending plug-ins manifest file, and callback objects created by the extending plug-in to receive specific inputs from the extended plug-in.

The key architectural points from an ecosystem view are 1) platforms are well-defined, and have broad applicability, 2) the platforms provide well-documented ways for developers to build additional functionality on top of them, and 3) all APIs use a consistent plug-in approach that provides a single flexible mechanism.

B.2.2 Governance

The Eclipse governance model ensures that no single entity is able to control the strategy, policies or operations of the Eclipse community. Eclipse has three major mechanisms: decisions made by the Eclipse Foundation, decisions made in three participatory Councils, and decisions left to participants.

Eclipse Foundation. The Eclipse Foundation (EF) consists of a Board of Directors, officers, and staff. The Foundation makes decisions in the following areas:

- Handling of intellectual property and other legal issues,
- Deciding membership and membership criteria,
- Development process and IT infrastructure,
- Branding, how the name can be used and by whom,
- Fostering community, shaping the culture, stewardship, and
- Interactions with external bodies, such as standards, specifications, other OSS ecosystems such as Apache.

The Foundation has established several basic principles or *"rules of engagement"*, especially openness, meritocracy, and transparency.

Members of the Eclipse Foundation can be firms and individuals. There are four membership classes: Strategic Member, Enterprise Member, Associate Members, Solutions Member, Committer. The first five types are for companies exclusively. Associate Members show participation in the Eclipse ecosystem by their membership. Solutions Members are engaged in the development because they offer products and services based on, or with, Eclipse. Enterprise Members influence the development of the Eclipse ecosystem because they rely heavily on Eclipse technology. Eclipse is seen as strategic platform for Strategic Members and they provide development resources or similar to support the development of the technology. Committer members are individuals who through a process of meritocracy can commit changes to project source code.

Strategic Members automatically hold seats in the Board of Directors, whereas representatives from Committers are elected annually.

The funding of the Foundation is based on annual dues from its members. Four central services are provided to the community: providing an IT infrastructure, managing intellectual property, supporting the development process, and further development of the ecosystem.

Participatory Councils.[58] The Eclipse Management Organization (EMO) consists of the Foundation staff and the Councils. The EMO is responsible for organizing and selecting the chair of Councils, enforcing policies, leading the Eclipse Platform development, marketing, and outreach. There are three Councils, one each for Requirements, Planning, and Architecture. All three Councils and the staff of the Eclipse Foundation are responsible for the Eclipse Roadmap, which describes the future directions of the Eclipse project. The Requirement Council gathers, analyzes, and prioritizes requirements, producing a set of themes and priorities, and the Planning Council provides the actual plans from the projects. The Architecture Council is theoretically responsible for describing the architecture and how it should evolve, even though it could not be realized in the last years. Existing industry trends and how the Eclipse community responds to them are added to the Eclipse Roadmap by the Foundation's staff. Beyond that, the Architecture Council is responsible for providing mentors for new projects to get started and to evolve development policies that span

[58] We are very grateful to Mike Milinkovich (Executive Director of the Eclipse Foundation) for his very helpful comments on this section.

across the community (e.g., how to manage API deprecation). The Planning Council can be seen as most important Council because it is responsible for managing the annual release train.

Member Business Decisions. Key business decisions are left completely to members. The Foundation takes no position on what members should sell, what their business models should be, whether they should contribute code to the free distribution, or whether they should participate in ecosystem governance. It is the expectation of Foundation that members will pursue their own business interests, and that contributions to the Foundation are a means to that end.

Projects and Release Management. Eclipse has a project hierarchy, with top level projects at the root, zero or more container projects, and one or more operational projects at the leaves. The top-level projects must have Project Management Committee (PMC), while subprojects have one or more project leaders. Contributors and committers perform the technical work for each project – the latter have write access to the repository. Committers elect new committers. Each project has a life cycle that goes through stages of Pre-proposal, Proposal, Incubation, Mature, Top-level, and Archived. At least two mentors (members of the Architecture Council) are necessary to propose new projects to the EMO. Only after *"graduation''*, the project is member of the Eclipse Community. A technical overlap between projects is not prohibited but is strongly discouraged. Reviews are held to determine changes in stage, as well as for other purposes such as approving a release. Project management is responsible for ensuring that the project follows Eclipse policies and rules of engagement.

B.2.3 Collaborative infrastructure and culture

The Eclipse Foundation provides an IT infrastructure that includes all the basic tools found in most open source ecosystems. They include *"CVS/SVN code repositories, Bugzilla databases, development oriented mailing lists and newsgroups, download site and web site"* (Eclipse).

The Eclipse ecosystem has evolved three core values, referred to as *"rules of engagement"*, that support collaboration among all participants (Eclipse). They are

- *Openness:* that the ecosystem excludes no one, and everyone has the same opportunities,
- *Transparency:* technical discussions are carried on out in the open, easily accessible by the public, and
- *Meritocracy:* responsibility and leadership are conferred in proportion to one's contributions.

In addition to online collaboration, the Eclipse Foundation sponsors two annual conferences, EclipseCon in the United States and the Eclipse Summit in Europe. The conferences are approximately 6 months apart, and provide face-to-face forums for Eclipse talks, tutorials, panels, workshops, and sponsored sessions focused on products; programming challenges (such as the 2010 EclipseCon e4-Rover Mars Challenge); and social occasions.

B.2.4 Business opportunities

A key insight into Eclipse business models comes from an Eclipse Foundation presentation in 2007. The basic idea is that companies should seek to avoid devoting a large proportion of engineering effort for infrastructure, which provides no differentiating value, and should seek to increase the proportion of engineering effort devoted to high-value differentiating features. Eclipse provides a way of collaborating on infrastructure, i.e., the Eclipse platform, while competing on products.

In addition to product developers, of which there are many in the Eclipse ecosystem, viable business models include

- Custom software vendors that use Eclipse (e.g., BandXI),
- Services such as
  - Supporting Eclipse development environments (e.g., Innoopract),
  - Training in Eclipse technologies (e.g., Modular Mind, Opcoach, RCP Vision, Innovent Solutions), and
  - Systems integration (e.g., OpenMethods).
- Products such as
  - Platforms for commercial distributed software development teams (e.g., Collabnet, MKS),
  - Code analysis modeling, testing, and other development tools based on Eclipse (e.g., itemis, Black Duck, froglogic), and
  - Business intelligence reporting tools (e.g., Actuate, Innovent Solutions).
- Selling products complementary to Eclipse, such as
  - Supporting projects for mobile application development, to help sell devices (e.g., Nokia, Motorola), and
  - Specialized releases for developing device software (e.g., Wind River).

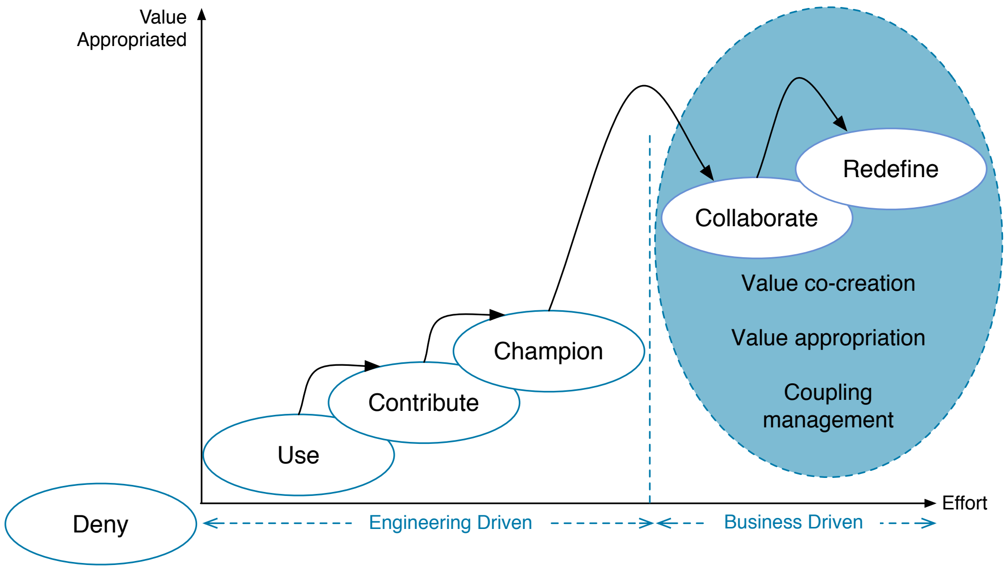


Figure B-V. Open source maturity model. (following Bailetti, 2008).

The Eclipse Foundation has promoted a maturity model of open source involvement (cf. Figure B-III) that shows a progression of business incentives for participating in the Eclipse ecosystem. At the lowest level, companies ignore or deny the existence of open source. Involvement tends to begin with simple use of open source products, and escalates through contribution and eventually championing open source. These are all engineering-driven motivations, as firms seek cost-effective tools and components. Eventually, firms may become more deeply involved by collaborating on infrastructure that has little or no differentiating product value, redefining their business model and focusing on value added products and services.

B.3 GNOME

GNOME (GNU Network Object Model Environment) is a free graphical user interface for free and open source operating systems such as Linux, BSD and Solaris. It was initiated in 1997. In addition to the desktop environment for users, GNOME provides for programmers a development framework for desktop applications (Wagstrom, 2009).

B.3.1 Technical architecture

GNOME is a graphical desktop environment and an application framework. The architecture, which is primarily realized in the programming language C and Python, consists of a collection of libraries and applications that can be roughly divided into four main groups (cf. Figure B-VI): needed libraries (e.g., GLib, XLib, CORBA), core applications (e.g., windows manager, configuration tool), applications (e.g., mail client, word processor), and third party applications (German, 2004). At the moment, GNOME is migrating from CORBA to D-Bus, therefore the architecture described following, might be soon outdated.

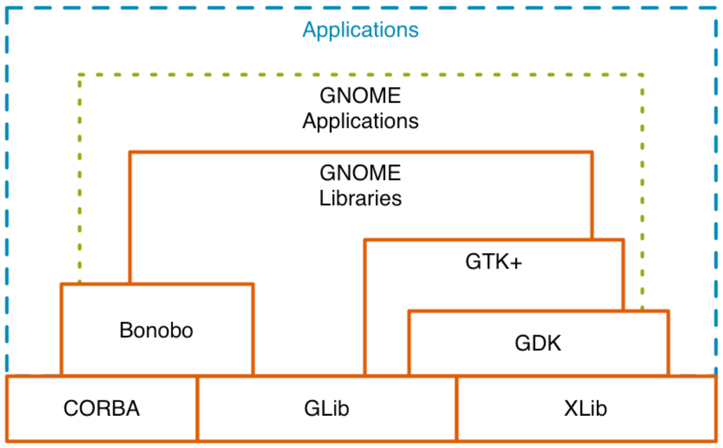


Figure B-VI. GNOME Architecture at a glance (Jang, 2006)

Two main libraries are briefly introduced: Glib and GTK+. Glib provides basic core application building blocks (e.g., common data structures, loop implementation) for libraries. GDK is a wrapper for the low-level library Xlib (X Window System) that serves as intermediate to GTK+. GTK+ (GIMP ToolKit) is the primary library used to construct user interfaces in GNOME. This library can be seen as an abstraction layer that enables developers to program user interfaces without dealing with the low-level details of drawing and device interaction. GTK+ has been developed over the last decade; it has a well-defined and well-documented object-oriented API.

Another important module is Bonobo, a framework for creating reusable components for use in applications. It is built on top of the industry-standard Common Object Request Broker Architecture (CORBA) and provides all common interfaces needed to create and use components in GNOME applications.

More generally, modules can be divided into platform modules, desktop modules, and language bindings. There are 21 platform modules and their APIs and ABIs (application binary interfaces) must stay stable until the major release number changes (e.g., GNOME 2.0 to 3.0)[59]. For desktop modules (16) a stability of their APIs and ABIs is not guaranteed. While platform modules are necessary to develop additional functionality for the GNOME project, desktop modules are

[59] We are grateful to Brian Cameron for pointing this out.

primarily necessary to ensure the *"basic"* functionality of the GNOME Desktop. Libraries are introduced in the desktop release to mature, and once they've stabilized and they are needed, they are moved into the platform.

The main advantage of platform modules that comprises the core can be seen in the development framework for desktop applications because it allows third parties to use existing functionality and to build their applications on top of it; the gnomefiles directory shows the extensive number (over 2,000) of available modules and applications (cf. http://www.gnomefiles.org) that ranges from commercial to open source licensed offerings.

The GNOME project actively supports development of new functions by providing for example the Integrated Development Environment (IDE) Ajunta for the GNOME desktop.

Moreover, GNOME is participating in freedesktop.org[60], which are open source/open discussion software projects aiming to enhance interoperability and shared technology for X Window System desktops by building a base platform for desktop software on Linux and UNIX that comprises both software and specifications.

B.3.2 Governance

In its first years, the GNOME community was loosely organized with a *"constitutional monarchy"* led by one of the founders, Miguel de Icaza (German, 2002). In 2000, because of the ongoing growth of the community and oftentimes less transparent decision-making processes, the GNOME Foundation, a non-profit organization (501(c)(3)), was founded. Even though the GNOME project has existed over 10 years, the GNOME Foundation has a flat organizational structure and a comparatively small size. In June 2010, there were 374 members, the Board of Directors (7 representatives elected by members), the Advisory Board (11 companies) and one Executive Director (besides the part-time administrative assistant and part time system admin, the only person paid by the Foundation).

Board of Directors. The Board of Directors is the decision-making body of the Foundation. It manages the release processes; is concerned with marketing, legal and financial issues of the whole project; determines which individual projects shall be part of the official GNOME release; and specifies standards, which define GNOME compliance - but it is not directly involved in technical decisions within the projects. The Foundation serves as spokesman for the community, its values and rules. Organic growth of the community is one major concern. Furthermore, the Foundation offers the technical infrastructure that can be used on a voluntarily basis to run existing projects and to integrate new ones.

Membership. Members of the Foundation are only individuals who make a non-trivial contribution to the GNOME project (meritocracy) and who are approved by the Membership Committee. People can participate in various defined projects such as the Accessibility Team, Bugsquad Team, Build Brigade, Documentation Project, Translation Project, Usability Project, System Administration Team, and Marketing Team. Each project addresses a specific issue in the development process or has a more general role like GNOME love, which helps new project members to get involved. A membership lasts for two years. Each member has one vote during the annual voting for the new Board members; candidates with the highest number of votes for the Board of Directors are elected. Furthermore, a member of the Foundation can run for the GNOME Foundation Board and can

[60] More information can be found at http://www.freedesktop.org.

suggest, vote or create a referendum (Project, 2010). Such a referendum goes only to vote if 10% of all members of the Foundation endorse it. The majority vote is finally necessary to get it passed.

Advisory Board. Companies and organizations (e.g., Google, IBM, Free Software Foundation) can participate directly in the Foundation by being members of the Advisory Board. They financially support GNOME by their membership fees at a minimum (fee depends on the size of the company, non-for profit organization are exempted) and in addition for example by the Accessibility Program, by Hackfests and by the technical infrastructure of the Foundation. Even though the Advisory Board has no decision-making authority, it acts as advisor to the Board of Directors on various topics such as: it provides input on financial, project management, and system administration issues, it identifies opportunities and supports collaboration outside GNOME, and it offers mentorship for community members. Employees of Advisory Board members are often participating in GNOME Foundation events and projects as well as supporting by specific tasks.

Project Management. GNOME's software development projects are organized within a federated system (Wagstrom, 2009). Core programmers manage their projects and oversee the whole software development, for example they produce releases and integrate changes (patches) from other people. Even though the projects are quite autonomous, they adjust their developmental roadmaps and goals to the overall GNOME community strategy.

Especially projects that are part of the release or that link to projects that are part of the release should follow specific guidelines (such as development guidelines), should meet the user interface guidelines and should be compliant with the accessibility guidelines.

The source code of the GNOME project is freely accessible by the GIT repository. But only as key contributors approved people are allowed to commit changes to the source tree. A contributor can apply for direct access rights to the code repository after submitting a reasonable number of patches (to a core developer) or Bugzilla reports to a project, or something entirely different. The criteria differ based on the person's role and contribution. An application for commit privileges has to be supported by the module maintainer and approved by the Accounts Committee.

In order to get a project hosted by the GNOME technical infrastructure, they have to meet specific requirements such as having a free source license, using GTK+/GNOME technologies, and must have had at least one public release. Furthermore, the accounts committee and people on the gnome-hackers mailing list have to agree to it.

Release Management. Projects, or so-called modules - separate libraries or applications, with one or more branches of development included, respectively - can be proposed for inclusion into the official GNOME release. The release team discusses such proposals based on specific judgment criteria (Project., 2010). Those criteria include for example:

- The improvement of overall desktop usability,
- The willingness to work with other teams, and
- The application must use GTK+ and other GNOME technologies.

An essential part of the criteria not only deals with technological requirements but with collaboration and community development in general. This might be a reason that most community members participate in various projects. Interestingly, companies that are involved in software development mainly contribute to core projects (Wagstrom, 2009).

In the release team, membership is normally by invitation and recommendation when another team member leaves. There is a defined release schedule that most frequently begins with the creation of a stable branch in each release project. New releases of GNOME are available as single files (source tarballs) in the source code repository. Therefore, for example operating system vendors create distributions in the form of easily installed, pre-compiled packages for their systems or provide them for end users.

Meanwhile, the GNOME has a 6-month release cycle for its *"official"* modules. In June 2010, 161 modules were included into the official release (consisting of 22 platform modules, 94 desktop modules, 2 admin modules, 6 dev tools modules and 12 C+, 5 Java, 8 Perl, 5 Python, 10 mobile and 2 misc bindings).

Intellectual Property. The platform libraries are licensed under the LGPL to allow proprietary software to link against them. All other modules in GNOME are using the GPL license or are using a license that is GPL compatible.

B.3.3 Collaborative infrastructure and culture

Even though GNOME is loosely organized, as most free software projects, there are various communication and collaboration tools available. The main website is for users and comprises release information, downloads, and documentation.

For developers of the core modules and for people using the development platform there is a developer portal available. The technical infrastructure consists of the usual open source software development tools such as bug tracker (Bugzilla), source code repository (git), a wiki, various mailing lists, instant messaging (esp., IRC), blogs, and blogs aggregators. Even though, this infrastructure can be used by every GNOME project, project maintainers are not forced to apply it.

Additionally, *GNOME News* contains information regarding the whole community whereas in *Planet GNOME* posts from personal weblogs of most of the developers are aggregated on one place.

There are several conferences each year, for example GUADEC, GNOME Boston Summit, GNOME Asia, and GUADEC-ES. GUADEC, GNOME Asia, and GUADEC-ES are annually conferences for GNOME users, developers, and vendors. The GNOME Boston Summit is an annual hacker get-together for developers.

Participants in GNOME are expected to conform with the GNOME Code of Conduct (GNOME). The principles are brief and few, but they describe expectations that participants will *"be respectful and considerate [...] patient and generous [...] and assume people mean well."* The high aspirations of the community are captured in the preamble of the code of conduct, *"GNOME creates software for a better world."* (GNOME).

B.3.4 Business opportunities

In order to provide a viable alternative to established commercial operating systems such as Microsoft and Apple, companies such as Novell, RedHat and Sun invested in the GNOME project in order to provide the end user with a free desktop. In 1999, RedHat included GNOME 1.0 in RedHat Linux and has been supporting the development since then. One reason for this support is that these companies wanted to sell support and services around free and open source software. In 2000, Sun founded the GNOME Accessibility Development Lab in order to ensure further quality improvement of the desktop application. Also Mozilla supports the accessibility efforts of the GNOME project.

In 2007, Canonical Ltd., the commercial sponsor of Ubuntu, joined the GNOME Foundation's advisory board in order to influence more actively the development (in form of feedback and support) of the GNOME Desktop, which has been used by Ubuntu.

Another business model is to employ open source software for existing hardware solutions. Software from the GNOME project is in use in a myriad of devices like the One Laptop Per Child and Nokia n800 series of Internet tablets; numerous start-up firms have created solid businesses around the project. In 2007, Collabora was one founding member of GNOME Mobile. The GNOME Mobile stack is used in mobile platforms such as Nokia's Maemo, Intel's Moblin and Linux Foundation's MeeGo platform. Companies like Collabora work with the GNOME community and platform companies to deliver free and open source solutions.

All mentioned companies are OSS integrators. Their main motivation is to support a viable alternative to existing commercial Desktops on different free and open platforms. Nevertheless, these companies are members of the Advisory Board; a more direct participation of companies within GNOME is only possible if an employee of a company gets involved in the community as an individual.

B.4 Mozilla

The Mozilla Foundation is self-described as *"a non-profit organization that promotes openness, innovation and participation on the Internet [...]. We provide core services to the Mozilla community and promote the values of an open Internet to the broader world."* (Mozilla.org).

These goals shall be achieved through community-developed software that enables people to easily and freely use Internet services. Their most widely known applications are the Firefox web browser and the Thunderbird e-mail client. But the community is also responsible for the development of the very successful software Bugzilla, a tool to support collaborative software development and defect tracking.

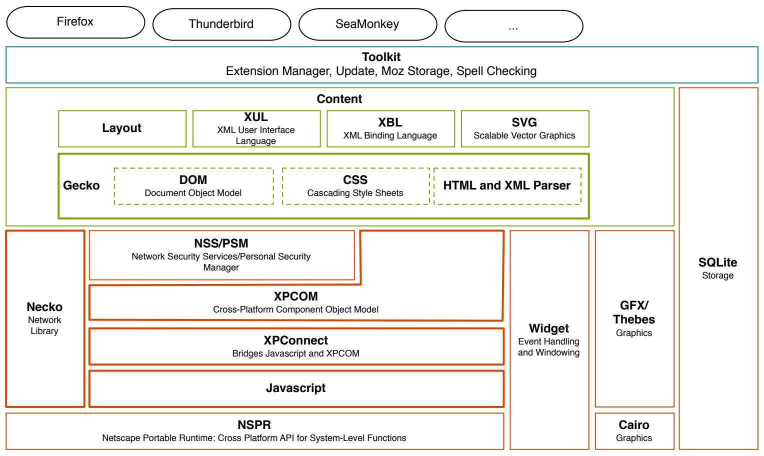
As of November 2009, Firefox's market share was around 25% worldwide: closer to 20% in the US and Asia; around 33% in Europe (Baker, 2009).

B.4.1 Technical architecture

Mozilla applications include seven tools such as browsers Firefox (for desktop or mobile) and Camino (for Mac OS X), e-mail client Thunderbird, calendaring software Lightning and Sunbird, software development tool Bugzilla , and SeaMonkey, an integrated Internet suite that includes source code from several other Mozilla applications. Additionally, there are applications which are based on Mozilla technologies, for example the Epic browser and Logitech harmony remote.

Mozilla applications are built on the Mozilla application framework which consists of several core technologies. These core technologies include a standard-based layout and rendering engine (Gecko), an extensible API for networking and communications functions (Necko), script execution engines, and the XPCOM, the Cross Platform Component Object Model (Mozilla.org). The latter is an object interface that allows interfacing between any programming language. The main objective of this generic framework is to provide cross-platform functionality for network (Web) applications. All technologies and their APIs are well-documented so that other developers can change them if needed for their purposes or can embed them in their applications, even proprietary ones. In order to support other organizations to use Mozilla technologies and to build their own application on top of them, Mozilla provides a development platform (called XUL) for desktop applications. Furthermore, the project provides various independent components such as SpiderMonkey

JavaScript engine and Gecko (third parties can use Gecko as a browser within their own application), and modules that are not even used by Firefox but available for the whole community such as Rhino.



Firefox
Thunderbird
SeaMonkey
...
Toolkit
Extension Manager, Update, Moz Storage, Spell Checking
Content
Layout
XUL
XML User Interface Language
XBL
XML Binding Language
SVG
Scalable Vector Graphics
Gecko
DOM
Document Object Model
CSS
Cascading Style Sheets
HTML and XML Parser
SQLite
Storage
Necko
Network Library
NSS/PSM
Network Security Services/Personal Security Manager
XPCOM
Cross-Platform Component Object Model
XPConnect
Bridges Javascript and XPCOM
Javascript
Widget
Event Handling and Windowing
GFX/ Thebes
Graphics
NSPR
Netscape Portable Runtime: Cross Platform API for System-Level Functions
Cairo
Graphics

Figure B-VII. Simplified Mozilla Application Framework (according to (Schroepfer, 2007)).

Mozilla emerged from the former Netscape browser. Netscape's source code was originally developed as commercial proprietary software, built for performance and quick shipment by a team of co-located developers working together for the same firm. At the time when it was released as Mozilla, it was not modularly designed for distributed development in an open source community. Some experts believe that the open source development *"really requires a software system which is as modular as possible, because otherwise you can't easily have people working in parallel."* (Torvalds, 1999). After an intense restructuring effort that improved the modularity of the code, contributions increased significantly. This effort may have been key to the project's success: *"The Mozilla experience suggests that proprietary products may not be well-suited to distributed development if they have tightly-coupled architectures. There is a need to create an "architecture for participation", one that promotes ease of understanding by limiting module size, and ease of contribution through minimizing the propagation of design changes [...] We speculate that without the architectural changes made in 1998, the Mozilla experiment may well have failed"* (MacCormack et al., 2006).

B.4.2 Governance

The Mozilla Project began in 1998 with the release of Netscape's browser code. This move was meant to tap the creative efforts of thousands of developers over the Internet, and to better compete with Microsoft's cost-free Internet Explorer. After a while, this code was given up in favor of designing a new rendering engine, called Gecko, still in use today. The first major open source version of this software (Mozilla 1.0) was released in 2002 to little fanfare (Mozilla.org). Firefox 1.0 was released in 2004 and quickly took off, with over 100 million downloads in the first year (Mozilla.org).

Mozilla Foundation. The Mozilla Foundation was established in July 2003 as a 501(c)(3) non-profit organization, headquartered in CA and guided by the principles of the Mozilla Manifesto. It replaced the former Mozilla.org organization which was founded by Netscape Communications Corporation in 1998. The initial US $2M investment came from American Online (AOL) which acquired Netscape Communications Corporation in 1999 (Decrem, & Corre, 2004). Although thousands of people are part of Mozilla, the Mozilla Foundation itself is a small team of people (Mozilla.org, 2005). It has a five-director Board, eight full-time staff, three part-time staff, and no members. The Board is self-elected with broadly defined powers.

The Foundation focuses on ecosystem and cultural management, maintaining the collaborative infrastructure and culture discussed below (including Drumbeat, cf. Section B.4.3). They are responsible for making decisions about these collaborative efforts including licensing decisions. The Foundation also oversees and coordinates software development for projects other than Firefox and Thunderbird.

There is presently no overlap between Board members and staff working directly for the Foundation. The Foundation coordinates projects in support of the Manifesto's principles. They also award grants for projects promoting openness and accessibility on the Web. The Mozilla Foundation has affiliate foundations in Europe, China, and Japan, independent organizations contracted by the Foundation for development and promotion of Mozilla products in their regions (Mozilla.org, 2005)

Mozilla Corporations. The Mozilla Corporation was established in August 2005 as a wholly owned subsidiary of the Foundation to coordinate the development and marketing of Mozilla technologies and products (Mozilla.org). The main motivation to form such a commercial subsidiary is the possibility to generate revenue and this revenue can be used to support the further development of Mozilla's open source technologies. The Mozilla Corporation has a staff of about 175 and there are subsidiaries in China and Denmark for work there, and a third subsidiary to operate branch offices in foreign locations. Despite the commercial background of the corporation, the community and the corporation should work closely together (Mozilla.org). For example, Mozilla Corporation employs 58% of individuals who have code review and check-in authority on multiple modules, 74% of developers with access to security bugs and 70% of super-reviewers.

The Mozilla Corporation provides the main of financial activity within the Mozilla Foundation, on the order of US $80M. Most of this comes from companies such as Google, Yahoo, Amazon, eBay, and others related to the search functionality in Firefox (Mozilla.org).

Mozilla Messaging, Inc. a second corporation also wholly owned by the Mozilla Foundation, was launched in 2008. This company develops and promotes the Thunderbird e-mail client. Mozilla Messaging has a staff of 10 who work with a much larger volunteer community.

Project Management. The Mozilla Foundation governs the source code repository and committer rights. Each module in the Mozilla project has zero or one module owner, who is responsible for setting the overall goals for that module, requiring a deep understanding of how it works and how it relates to the rest of the Mozilla codebase (Mozilla.org). Module owners are responsible for reviewing code submitted to that module before checking it in, although they may also designate one or more *"peers"* who can also review and check in code for their module. Any changes, which affect architecture, module interaction, or APIs, must also be reviewed by one of 27 senior developers (*"super-reviewers"*) after review by the module owner.

A module owner is usually a different person than the default *"component owner"* who receives notice of all component's bugs and is responsible for prioritizing and assigning them to the correct developers. Components (in their Bugzilla) do not map directly on to modules; components are related to how bugs are experienced and modules are related to how the code is structured. There is a meta-level *"module ownership"* module. Owners and peers of this module can install or remove owners of other modules.

Incubator repositories are available to module owners for challenging collaborations with contributors who do not yet have commit access to the Mozilla tree (Mozilla.org). They can be used for work that is planned in Mozilla's roadmap (not for "potentially related" work) that cannot be broken into smaller segments for the usual review and check-in process. Incubator repositories are temporary (< 6 months), until it is clear that the work has potential (or not) and at least one core contributor is ready for commit access to the Mozilla tree.

At the highest level, Mozilla has two *"benevolent dictators"*, who have the final say in any technical dispute (Brendan Eich) or non-technical dispute (Mitchell Baker). Eich is also a super-reviewer and driver.

Release Management. Within the Mozilla governance *"release drivers"* are defined in order to provide project management for milestone releases. The election of the release team is based on meritocracy. Close to a milestone release, a group of 17 release drivers (8 of whom are also super-reviewers) become very active, charged with finalizing a milestone branch of the code to make sure it is stable, secure, and ready for release. During these periods, drivers review all submitted patches to determine if they will become part of the next release.

Intellectual Property. Code contributed to Mozilla is tri-licensed under the Mozilla Public License (MPL), GNU General Public License (GPL), and Lesser General Public License (LGPL) (Mozilla.org). Add-on developers may choose whatever license they want for their add-ons (with certain minimum criteria to allow Mozilla to distribute it). The Mozilla Foundation owns the Mozilla trademarks and other intellectual property.

B.4.3 Collaborative Infrastructure

Mozilla's instance of Bugzilla plays a central role in the discussion, management, and development of changes to the software. Communication also takes place on IRC, a wiki, forums/mailing lists, and occasionally via direct instant messaging between two developers discussing a technical problem or review. The wiki is open to editing by anybody, but project-specific sections are socially limited to those who are working on that project (discussion pages open to anybody). Weekly conference calls are held to discuss updated in each major project.

"The common thread that runs throughout Mozilla is [participants'] belief that, as the most significant social and technological development of our time, the Internet is a public resource that must remain open and accessible to all [...] In the end, the Mozilla community, organization and technology is all focused on a single goal: making the Internet better for everyone" (Mozilla.org).

The organization also emphasizes its transparent and collaborative nature. The Mozdev site provides hosting resources for developers working on Mozilla related projects (e.g., add-ons) but Mozilla now provides those resources directly and MozDev is winding down as an organization and site.

Mozilla is also starting a *"Drumbeat"* platform which shares a common culture and collaborative infrastructure (but little else) with the software projects. Drumbeat is a way of involving more people with Mozilla's core culture and goals, at least partially through the spread of skills and knowledge. Mozilla's role in the Drumbeat Project is to provide the collaborative infrastructure and cultural management. Any individual or organization can easily become a member for free by registering on a simple web form, and easily start or join a project. It provides an online gathering and project hosting space for people, projects, and groups who actionably care about keeping the web open and free. Drumbeat will host online conferences and web pages, process financial details for donations, and provide a common branding and communication strategy for volunteer recruitment. The Mozilla foundation also picks some projects to *"sponsor"* with coaching and/or financial support (around US $25K, typically). Mozilla currently sponsors Peer2Peer University's School of Webcraft project, WebMadeMovies, and Universal Subtitles. To start a project, one needs to upload a project title, description, and funding goal. This person (or group) then becomes the project manager and is responsible for all aspects of the project: recruitment, road-mapping, release, media selection, etc. They can promote (recruit for) the project on the Drumbeat web page and at Drumbeat events. As an example of a project led by organizations, the Universal Subtitles project is nominally led by *"Help Mozilla, Miro and the Participatory Culture Foundation"*.

B.4.4 Business Opportunities

Netscape initially opened the source to their code in the hopes of getting free development work by open source volunteers around the world. They hoped that a strong, free, core browser would give them a ready market for their Networked Enterprise software and Netcenter services, and make them a driver of internet standards (Netscape). A basic but free (costless) client platform would make it easier for customers to adopt Netscape's primary revenue-producing products and services.

Today, the Mozilla project presents the same sort of business opportunities to anybody whose business involves Internet services or functionality. By building a free add-on for Mozilla products, a business can make it much easier for customers to interact with their main sites and consume their revenue-producing products and services. For example, the eBay search extension can provide fast access to an easy customer experience shopping on eBay from any starting point when the browser is open (Mozilla.org). Other add-ons are available to use online translating services, bookmark sites on Delicious, purchase products described on a page you're viewing, download music and video, use telephony services, find the weather from your favorite weather prediction source, or use a host of other online services (Mozilla.org).

Many developers of Mozilla products are themselves end-users, and implement (via add-ons or core code patching) the functionality they want or need from the software. Besides the added functionality, developers may receive a free technical education through hands-on interaction with experts, earn peer recognition, enjoy the benefits of social interaction, and feel like they are part of something much larger than themselves (Mozilla.org, 2008). Early developers were also motivated by how publicly visible Mozilla would be as a paradigm of whether or not open source could be a viable model (Phillips, 2008).

When a business uses the software (as an OSS Deployer), they may assign development resources to the Mozilla project (and thereby become OSS Participants as well). This person or team primarily ensures that their business needs are met by the software, but would also be expected to help the community in other general software development goals, so that help would be available from the larger community if needed. For example, a software development firm using Bugzilla has a strong incentive to build up the strength of that product; a university using Firefox has a strong incentive to ensure the security and functionality of that product; an office which coordinates via Mozilla's free e-mail client and calendaring clients has a strong incentive to ensure that those products are highly functional and bug-free.

Businesses looking to introduce new software functionality also generally have a financial incentive to extend open source technologies and applications rather than bear the added cost of developing software from scratch, because a good part of the work is already done. This approach also helps ensure later stability and maintainability of a business's software solutions even if there is turnover in software staff. In the terms of Section A.4, these firms are OSS Integrators.

Appendix C: Commonalities and variabilities of four open ecosystems

Among the four open source ecosystems we have studied, we observed both commonalities and variances. Commonalities suggest design principles that may be essential to successful operation of an open source ecosystem, while variances suggest options for different ways of implementing these design elements. The commonalities and variabilities analysis is used successfully in the design and understanding of software applications and languages (Coplien, Hoffman, & Weiss, 1998); we here extend this model to examine software ecosystems and communities. The approach is very similar to Yin's (Yin, 2009) pattern matching method for the analysis of case study data.

This chapter describes each of the four dimensions in terms of the commonalities and variabilities of the four studied ecosystems. A commonality is a design principle that is applied in all four ecosystems, and often, this design principle has the same value in each ecosystem. Variability means that each ecosystem might exhibit a specific design principle but the value differs in each system.

In Section 3, we used the identified commonalities of all ecosystems to help us think through the *"should do"* requirements and the identified variabilities to prime consideration of *"might do"* requirements for the VistA ecosystem design.

C.1 Technical architecture

Following we reveal existing commonalities and variabilities of the technical architecture of each ecosystem. We focus on the general technical structure of the project, the question how extensible and interoperable the technical structure is, and how the final software product is provided to users.

The qualitative comparison showed that each ecosystem distributed its development efforts into single modules, i.e. projects. Especially stable interfaces and open standards are used to ensure interoperability. Also, three of four ecosystems provide a bundled version of their software.

C.1.1 Commonalities

Each of the four analyzed ecosystems consists of multiple software projects to realize one software product or various ones. By decomposing a software product into single modules (a module often refers to one project in the ecosystem) allows developers to participate easier in the software development process because only a specific part of the functionality is unfolded and design changes affect only the changed module (MacCormack et al., 2006). Implementing open standards and providing stable APIs especially ensure interoperability between those software projects and at the same time, encourage third parties to implement additional functionality because a long-term usage is guaranteed.

C.1.2 Variabilities

The open source ecosystems examined here have significant variation in terms of their technical architecture. The type and the scope of the implemented software mainly cause these differences. GNOME and Eclipse provide more or less a single software product that can be adapted or extend based on the needs of their users. Apache and Mozilla follow more a general idea or philosophy, in the former ecosystem all projects are somehow related to the *"web serving problem"*, conversely the latter ecosystem whose projects are intended to *"make the Web a better place"*.

As expected, the final design of the technical architecture of each ecosystem differs. Three of four ecosystems provide a cross-platform (Mozilla and GNOME with GTK+) or Java platform (Eclipse) respectively and therefore, the software can be used on more than one platform. Also, three of four ecosystems provide specific development environments for their software to support third parties to implement additional functionality.

There is also some difference in how users can obtain the software from each ecosystem. Apache[61] and Eclipse[62] makes a list of mirrors available on its website, with one mirror chosen as "recommended" in a semi-random fashion for load balancing. Mozilla's download website[63] features a single download button which semi-randomly assigns a download mirror. By contrast, GNOME cannot be downloaded directly. It is free software, but in order to get it one must download an OS distribution[64] integrating the GNOME desktop.

C.2 Governance

In the following section, we focus on the general governance structure and governance processes, which consist of project and release management, as well as intellectual property.

As a result, each of the analyzed ecosystems is governed by a foundation. Another important commonality are well-defined governance processes, even though the characteristics of these processes differ which are mostly described in the variabilities section. For example, main differences are the types of members, how community members can participate in the decision-making process, and how developer can commit and create new projects.

C.2.1 Commonalities

Each of the investigated ecosystems is governed by a foundation, which has the code ownership. The Board of Directors is the main decision-making authority and one primary interest is community development. Besides the Board, each foundation has additional organizational entities, which are assigned to specific activities within the foundation such as conference organization and technical development.

Each ecosystem has well-defined development processes with specific roles and responsibilities. The source code is easily accessible from one central repository or over a central web page respectively. In each ecosystem specific guidelines are defined for when and how code can be changed by developers, which requirements exists to create new projects, and which steps are necessary within the release process.

There are also some commonalities in the particular leaders, donors, and developers who are active in various open source communities. For example, Brian Behlendorf is the co-founder of the Apache project and currently on the board of the Mozilla Foundation. Companies such as IBM and Google contribute significant resources to each of the four ecosystems studied here.

C.2.2 Variabilities

Even though each ecosystem is governed by a foundation, three ecosystems are non-profit organizations (501(c)(3)) and one ecosystem is a non-profit trade association (501(c)(6)). Whereas the latter can serve the business purposes of its members the former must serve public purposes.

[61] http://www.apache.org/dyn/closer.cgi
[62] http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/helios/R/eclipse-php-helios-win32.zip
[63] http://www.mozilla.com/en-US/products/download.html
[64] http://people.gnome.org/~daniellem/footware.shtml

The 501(c)(3) status requires that donations must be disclosed but in 501(c)(6) organizations this is not necessary. One of the 501(c)(3), the Mozilla Foundation has a very special organizational structure because the foundation owns two companies, which are wholly owned subsidiaries. These companies allow the Mozilla Foundation to generate revenue and to use this revenue for further investments in the development of its software.

Another main difference between the ecosystems is the design of the membership. The Mozilla foundation has no members, therefore community members have no influence into the decision-making process of the Foundation. GNOME allows only individuals to be member of the Foundation. The bylaws of Apache and Eclipse permit firms and individuals to be member of the Foundation. However, currently GNOME and Apache have only persons as members and they elect the Board annually. Eclipse has different membership types and two of them are automatically member of the Board without election. All other Board members are elected annually following a specific key per membership type. Also company participation is differently designed within the ecosystems. In GNOME and Apache the participation is limited to financial support and advisory function, in Mozilla companies can donate money to the Foundation and in Eclipse companies are members and they can determine the strategic development of the community. Even though Mozilla and Eclipse are sponsor-founded ecosystems, the design of company participation is very different.

The possibilities of participation differ for ecosystem members. Often, the majority of votes determines the final decision. In Apache, for specific decision the *"lazy consensus"* is used; positive votes without a negative vote are sufficient for the final decision. Only GNOME allows direct participation of its members by a so-called referendum.

Besides the general governance structure of the ecosystems, we had a look on the design of the development processes. The use of common collaborative resources is more optional in some ecosystems and mandatory in others. For example, Apache is quite strict about requiring projects to use the shared community infrastructure and do things *"the Apache way",* while Mozilla is less precise and encourages developers to be resourceful and creative in finding resources. Ecosystems vary with the way they handle new projects. Apache, Eclipse, and Mozilla have incubator procedures. GNOME has open hosting for new related projects. Mozilla's Drumbeat project hosts project organization resources for (especially) nontechnical projects.

Ecosystems also vary in how they grant different levels of access to contributors. Mozilla has multiple levels of commit access and uses social control within those bounds to determine who can commit to what parts of the codebase. Other projects have only *"committer"* and *"non-committer"* statuses and use pure social control to limit check-ins to the part of the code tree where a developer has expertise. The number of required reviews also varies, although patches with greater impact on any project generally require greater review. Some ecosystems follow a coordinated release schedule. Others, like Apache, allow each PMC-level project to set their own release schedule.

C.3 Collaborative infrastructure and culture

The collaborative infrastructure of an ecosystem is on one side determined by the used Internet tools to facilitate collaboration and on the other side by the culture of an ecosystem in terms of its openness. In our analysis, we specifically investigated the provided hosting services of each ecosystem and which requirements exist for projects to use these provided hosting services.

Our comparison unfolded that each ecosystem uses a wide range of tools to facilitate collaboration in the community. Main differences exist under which circumstances new projects can use these hosting services.

C.3.1 Commonalities
All investigated ecosystems have a collection of common community resources. First, they provide hosting for software development work; this includes file servers where the source code and distributions can be stored and downloaded, and space for people to upload project-related files. Also, all ecosystems use a software revision control system, which allows amongst other things to record who made what changes when, and allows easy reversion to an earlier version of the code. Another utilized tool is the bug tracking or modification request system. This allows developers or users to report on and discuss issues with the software. *"Bugs"* are a single *"place"* where a particular topic may be discussed, and the bug tracking system makes it easy to find and organize these discussions about changes to the software system. Besides these mainly development related tools, also communication tools and collaborative editing tools are applied. Communication tools are especially (E-)mailing lists, web forums, and IRC channel(s). These allow users and developers to communicate with one another, and allow each user and developer to manage his or her own preferences for communication and subscriptions. Wiki are used for collaborative editing purposes in order to document the current state and progress of the software development project or to collaboratively document the existing function of the software. Open source software projects tend to change more quickly than any central documentation authority could keep up with, so a Wiki allows each project member to participate in the documentation process and to change existing information more easily and quickly.

C.3.2 Variabilities
Though all of the open source communities we looked at provided these community resources, they varied in how much each of these tools were used, and who could (or had to) use them. Some ecosystems stipulate the usage of the community-provided resources. For example, the Apache project requires the utilization of their SourceForge hosting services for all software development work, but at the same time, allows every project to design their own web presence. Apache projects can often only be identified by their Apache logo. Mozilla extension developers, by contrast, are free to use any hosting services they like, and add-on developers must choose their own. (Mozilla only hosts add-ons for download during distribution, not development[65]). Ecosystems also vary in their choices of particular implementations for the above technologies and in how much they use various channels.

C.4 Business opportunities
Ecosystems allow (or do not allow) business opportunities based on decisions made when choosing the license governing the software, and the decision about the technical architecture and if and how the software can be extended.

C.4.1 Commonalities
A successful, sustainable open source ecosystem enables various business opportunities within and outside of the community.

For example, open source software is generally licensed so that anybody may copy or distribute the software. This means that hardware or software companies in related areas can package the open source software in their product. As an example, a hardware company selling server machines can pre-load Apache web server and other products, providing their customers with an easy and complete turnkey solution. GNOME relies completely on other software vendors to integrate the

[65] https://developer.mozilla.org/en/Submitting_an_add-on_to_AMO

GNOME desktop with their software for distribution. These other vendors are generally also free projects but do not have to be.

There are some business opportunities from the sale of complementary products. When Netscape opened the source code of its web browser (seeding the Mozilla project), it did so hoping to increase its sales in complementary products such as internet access (as an ISP), content linking (e.g., Netcenter, Netscape's web portal), and web-based enterprise services. Giving away a free, open source platform allows businesses to charge for other services that rely on customers having access to that basic platform functionality. This is a similar model to wireless phone companies that will *"give away"* mobile phones to sell carrier contracts, except that the marginal price of software does not require a contract lock-in while the price of a mobile phone does.

Businesses can also build add-ons that make it easier for customers to access their regular business; for example eBay offers add-ons to Mozilla that make eBay searches and purchases easier, boosting eBay sales.

In some open source communities, the ability to develop add-ons or extensions provides a fairly direct business opportunity. Software developers can enhance the core functionality and charge the end user for this (proprietary) added functionality, which is called the *"open core"* model (and is the subject of some debate in the open source community). Eclipse offers this as a business opportunity directly.

Even simple use of the open source software can create business opportunities by lowering the barriers to entry in a particular field. A software start-up, for example, can use Eclipse and immediately have a highly functional development environment, and get right to work on their value-added projects that support their business model. They do not have to invest as much time or resources in inventing or setting up this development environment as a software start-up needed before Eclipse. Open source software can lower entry barriers for many industries.

C.4.2 Variabilities

Some open source communities offer the opportunity for members to become distributors, for example Red Hat in the Linux community as well as Eclipse, Apache, and especially GNOME. Mozilla allows others to bundle the software, but the Corporation is a direct distributor of the software so there is not much room for another distributor.

Some open source communities allow business opportunities through service links. For example, the Mozilla foundation has a deal with Google to make Google the default search engine and to set the default homepage to a Firefox-themed Google Search page, for which Google pays Mozilla a considerable sum of money. Google recoups the money in advertisements displayed alongside search results. At least until recently, this has been Mozilla's primary source of revenue (>85%). Google also contributes more developers and technical resources to Mozilla than any other company, not counting the Mozilla Corporation (Soghoian, 2007).

Appendix D: Tabular ecosystem comparison

Comparison of architecture of participation and technical architecture of GNOME, Apache, Mozilla and Eclipse; (*) adapted from (West et al., 2008b), (**) adapted from (German, & Hassan, 2009). (Last update in September 2010)

CHARACTERISTICS	GNOME	APACHE	MOZILLA	ECLIPSE
General				
Founded/released	1997	1995	1998	2001
Type of core/ initial software	graphical user interface, developer framework	web server	web browser	integrated development environment (RCP)/ application platform
Recent version	2.30 (desktop)	2.2.15 (http server)	3.6 (web browser)	3.6 (RCP)
Motivation for initial project initiation	Create a great free desktop platform	Collaborative development to improve existing httpd program	Netscape released application suite as an open source project because of high market competition	IBM released Eclipse as open source project in order to improve competitive position
Type of community(*)	Community initiated	Community initiated	Sponsor founded	Sponsor founded
Type of users	End-users with varying skills, desktop programmers	Professionals/users with development skills	End-users with varying skills	Professionals/users with development skills
Self-concept	One community of volunteers	Several separate communities, each focused on a different side of the "web serving" problem; collaborative development across both nonprofit and commercial organizations	Hybrid organization, combining non-profit and market strategies	Vendor-neutral, open development platform supplying frameworks and exemplary, extensible tools
Main reason for ecosystem cohesion	Shared philosophy; product vision, use of GTK+	Community management; culture; brand	Vision (promotes openness; innovation and participation on the Internet)	Annual release train (shapes how projects collaborate)
Governance				
Foundation	Yes, GNOME Foundation	Yes, Apache Software Foundation (ASF)	Yes, Mozilla Foundation	Yes, Eclipse Foundation
Foundation type	Non-profit organization (501(c)(3))	Non-profit organization (501(c)(3))	Non-profit organization (501(c)(3))	Foundation Non-profit trade association (501(c)(6))
Date of formation	August 2000	June 1999	July 2003	January 2004
Membership type	Individual (people who have made a contribution)	Entity/individual (but only individuals are listed as members); new members are nominated by current members and elected due to merit the	No members	Firm/individual (5 membership classes: Strategic Developer, Strategic Consumer, Enterprise Member, Associate Members, Solutions Member,

CHARACTERISTICS	GNOME	APACHE	MOZILLA	ECLIPSE
		foundation		Committer)
Membership voting rights	One vote per member	One vote per member (one-third of the members entitled to vote)	–	One vote per member but collective vote for Committer Members who are employed by the same organization
Membership fee	No fee for individual members, but large companies pays a membership fee of US $20,000, small companies of US $10,000, and non-profit organizations are exempt to join the Advisory Board	No fee, but there is a ASF Sponsorship Program (platinum sponsors: Yahoo, Microsoft, Google; gold sponsors: Facebook, HP; 3xSilver and 6 Bronze Sponsors)	–	Yes (annual dues), depending on the membership type
Number of members	374	332 (e.g., 8 IBM, 2 Sun, 4xCollabNet, 2x RedHat, 3xGoogle)	–	158 (13 strategic developer, 1 strategic consumer, 3 enterprise members, 68 associates, 73 solutions members, number of committer member unknown)
Organizational entities	Board of Directors, Executive Director, Advisory Board (11 companies and 7 organizations), membership	Board of Directors (Board), Officers (95), Project Management Committees (PMC) (88)	Board of Directors	Board of Directors with three standing committees (membership, finance, compensation) and Executive Director and Secretary; under the direction of the Executive Director is the Eclipse Management Organization (EMO), Advisory Board
Main decision-making authority	Board of Directors (single entity is not permitted to represent more than 40% of the Board) but many decisions are delegated to the teams, like the release team	Board governs the foundation, PMCs govern projects, officers govern day-to-day affairs	Board of Directors governs the foundation	Board of Directors (Strategic Developers and Strategic Consumers hold seats, as well as representatives elected by Add-in Providers and Open Source committers)
Main tasks of decision-making authority	Coordinates each release; determines the set of modules that are part of GNOME release; specifies standards which defines GNOME compliance; communicates overall strategy; manages staff and finances of the Foundation	Board responsible for management and oversight of the business and affairs of the ASF; each PMC has technical decision-making authority for its project (the main role is not coding)	Introducing Mozilla Drumbeat; fundraising and engagement; strengthen Mozilla's position as open Internet charity; make Mozilla community more productive, cohesive and understandable	Business and technical affairs are managed by or under the direction of the Board; EMO is responsible for organizing and selecting the chair of Councils, enforcing policies, leading the Eclipse Platform development, marketing, and

CHARACTERISTICS	GNOME	APACHE	MOZILLA	ECLIPSE
				outreach
Size of decision-making authority	At least 7 but no more than 15 members; Board members are elected by email, at least one month before the annual meeting; usually during the annual meeting the new board starts working actively; current board members: 1xRed Hat, 1xOracle, 1xSun/Oracle, 1 Novell, 1xrPath Inc, 2xNon affiliation	Board has 9 directors, PMC (54) consists of at least one officer (vice president) of the ASF (= chairman)	Minimum of 5 and maximum of 15, at the moment 5 directors (2xMozilla Corporation)	No pre-defined size (depending on member); consists of Strategic Developer and Consumer and Add-In Provider and Committer Member; at the moment 14 Strategic Members, 3 elected Add-in Provider Representative, 3 elected Committer Representative
Decision-making process	Preferential method: single transferable vote (STV)	Lazy consensus approach: a few positive votes with no negative vote is enough to get going (negative vote includes an alternative proposal or a detailed explanation of the reasons); process of “consensus gathering” by negative vote	Majority of votes in Board	Different majority requirements (ranging from simple to super-majority consent) depending on particular decision
Nomination and election process	Self-nomination of members, annual election, candidates who receive the highest number of votes are elected	Board is annually elected by members of the foundation; chair of the PMC is appointed by the Board	Annual election by the Board itself	Strategic Developers and Consumers are automatically presented without election; Add-in Providers and Committers are annually elected
Company participation	Advisory Board that includes 11 companies (Canonical, Collabora, Google, IBM, Igalia, Intel, Motorola, Novell, Nokia, Oracle, Red Hat) and 7 organizations (Sugar Labs, OLPC, Debian Project, Free Software Foundation, Mozilla Foundation, Software Freedom Law Center, Limo Foundation)	Not directly, only by non-directed monetary contributions to the ASF Sponsorship program	--	Companies are member of the foundation
Further organizational entities	Several teams (Accessibility Team, Travel Committee,	Several ASF committees (Infrastructure, Labs,	Foundation owns the Mozilla Corporation (established in 2005	Project Management Committee (PMC) manage top-level

CHARACTERISTICS	GNOME	APACHE	MOZILLA	ECLIPSE
	Translation Team, Art Team, "GNOME Website Reorganization Team", Accounts Team, Quality Assurance team, Sysadmin Team, GNOME Marketing, GNOME Website Reorg Team, GnomeWomen, Membership & Elections Committee, ModeratorTeam, UsabilityProject, UserGroups)	Attic (finalized projects), Legal Affairs, Conference Planning, Security, Travel Assistance, Community Development); specific officer roles (Java Community Process, W3C Relations, Brand Management, Fundraising, Marketing and Publicity)	as wholly owned subsidiary) and Mozilla Messaging (established in 2008 as wholly owned subsidiary)	projects; project leaders manage sub-projects; PMC leads are approved by the Board and PMC members and project leads are approved by the EMO; Architecture Council, Planning Council, Requirements Council are comprised of Strategic Members and PMC representatives
Intellectual property				
License (**)	LGPL (platform libraries), everything else GPL or GPL compatible	AL$_2$ [The Apache Software Foundation, 2010]	Core project is licensed under MPL$_{1.1+}$, GPL$_{2+}$, LGPL$_{2.1+}$	EPL$_{1.0}$, as long as a plug-in for Eclipse uses the plug-in API the plug-in can be licensed under any terms [The Eclipse Foundation, 2010]
License Type	Weak/strong copyleft	No copyleft	Weak copyleft	Weak copyleft
Code ownership	Contributors	Foundation	Foundation	IP ownership is diffused by all contributions share the same license
Technical architecture				
Programming Language	C, Python (mainly)	Various languages such as C, C++, Java, Perl, Python	JavaScript (mainly), C++, XUL, Cascading Style Sheets, XBL	Java
# Modules/ projects	730 modules (desktop, infrastructure, development tools, bindings, administration tools, platform, productivity tools, others)	174 projects (6 build-management, 1 content, 10 database, 5 graphics, 8 http, 3 httpd-module, 3 javaee, 52 library, 2 mail, 12 network-client, 23 network-server, 2 regexp, 4 testing, 1 virtual-machine, 17 web-framework, 24 xml)	86 modules, 7 applications, 2 Mozilla-based applications, approximately 5000 add-ons for applications	148 projects (64 in incubation phase conforming branding, 13 non-conforming branding)
# Core modules/ projects	110 projects (desktop only)	88 projects (corresponds to the number of PMC)	3 core products (Firefox, Fennec and Thunderbird)	74 projects in mature phase
Platform	Generic cross-platform application framework (GTK+), platform core technologies (e.g., graphical interfaces, virtual file system)	No common platform, project directory website realized by a RDF based DOAP (Description Of A Project) file	Generic cross-platform application framework	Rich client platform

CHARACTERISTICS	GNOME	APACHE	MOZILLA	ECLIPSE
Extensibility and interoperability	Stable API	Projects are encouraged to use open standards	Extension system (add-ons)	Component model (i.e., plug-ins)
Software development process				
General	Specific guidelines for projects that are part of the release or link to projects that are part of the release, otherwise decentralized and managed by maintainer	Software development activities are carried out in separate autonomous projects that are managed by Project Management Committees (PMC)	Mozilla Corporation and Mozilla Messaging is responsible for all software development activities	Projects passes through a pre-defined Project Lifecycle consisting of six distinct phases, status reports at least quarterly to EMO (Eclipse Management Organization) necessary
Defined roles	User, developer, committer, maintainer	User, developer, committer, PMC member (developer or committer, nominated by current members and elected due to merit the foundation), PMC chair, ASF member	Module owner (86 code module owners), super-reviewer (27 individuals), release driver (17 individuals), Bugzilla component owner, benevolent dictator	Contributor or committer, project leader, PMC member, PMC lead, Council representative, user, adopter (plug-in developer)
Code access	Freely accessible (GIT)	Freely accessible (SVN)	Freely accessible (CVS)	Freely accessible (CVS/SVN)
Commit rights	Registered GNOME developers can commit changes	Only after signing the Individual Contributor License Agreement (CLA)	Committer's Agreement and at least one person who vouch for competence is needed (well-defined procedure); final decision is carried out by Mozilla representative	Contributors who have the trust of the project's committers can be promoted committer through election (least three (3) positive votes and no negative votes within the voting period of at least one week); confirmation by the relevant PMC necessary
Commit process	Every module one repository with its own history	Decentralized in each project	One tree, but three levels of code access (L1: incubator, L2: general access, L3: access to core products)	Different decentralized repositories but directly accessible on one web page
New project creation	Sub-projects can be created anytime, new modules can be proposed to be included in official release (pre-defined requirements)	Incubator period in which a new project is evaluated based on the likeliness that it becomes a successful meritocratic community (e.g., because of the diversity of committership); functional overlap of projects is allowed	Module ownership system (Mozilla source tree), but for new projects (not experienced contributors) a separate incubator repository is available	New projects must be proposed to the EMO; at least two mentors necessary (members of the Architecture Council); projects go through incubation phase; only after "graduation" member of the Eclipse Community; overlap between projects is

CHARACTERISTICS	GNOME	APACHE	MOZILLA	ECLIPSE
				not prohibited but is strongly discouraged
Release process	Defined release schedule; creation of a stable branch	Defined process; creation of a stable branch	Defined process; creation of a stable branch; super/review necessary for all core projects	Release review based on certain criteria (successful advisory vote and the approval of the Release Review by the EMO is necessary)
Release team	Membership is normally by invite and recommendation when one person leaves	Decentralized responsibilities	Nominated drivers	Decentralized responsibilities
Release set	New modules are discussed based on judgment criteria by the release team [Project., 2010], modules are introduced in the desktop release to mature, only libraries of wide use and with stable API/ABI are moved to platform	Defined by each project	Defined by each project	Defined by each project
Release cycles	6-monthly release (only "official" modules)	Defined individually by each project	Defined by each project	Defined by each project but there is an annual release train

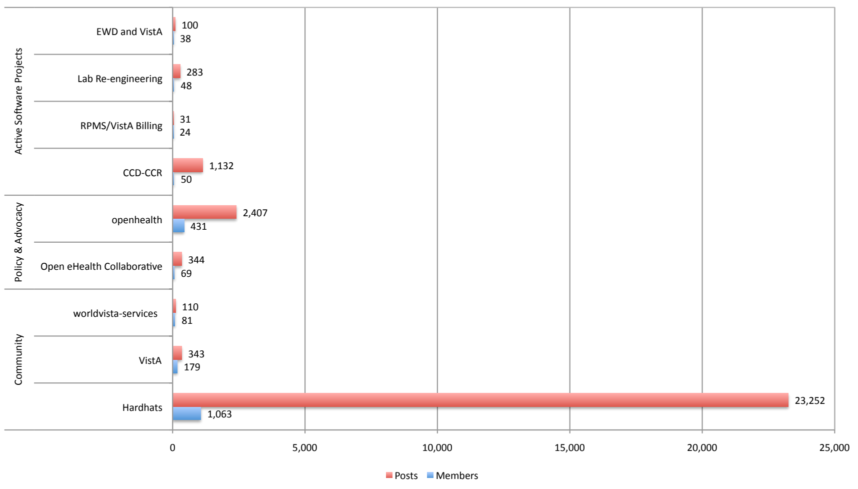
Appendix E: Brief overview about licenses

GNU General Public License, Version 3.0 (GPLv3)	
Brief description:	- Proprietary software linking is not allowed - Distribution of combination of software is allowed with software whose license is GNU GPL-licensed - Redistributing of the code with changes is only allowed if the derivative is GNU GPL
Source:	http: //www.gnu.org/ licenses/gpl.html
GNU Lesser General Public License (LGPL)	
Brief description:	- Proprietary software linking is allowed - Distribution of combination of software is allowed with some restrictions - Redistributing of the code with changes is only allowed if the derivative is GNU LGPL or GNU GPL-licensed - Compatible with GNU GPL
Source:	http: //www.gnu.org/ licenses/lgpl.html
GNU Affero General Public License (AGPL)	
Brief description:	- Proprietary software linking is not allowed - Distribution of combination of software is allowed with software whose license is AGPL or GNU GPL-licensed - Redistributing of the code with changes is only allowed if the derivative is GNU LGPL or GNU GPL-licensed - Compatible with GNU GPL
Source:	http://opensource.org/licenses/ agpl-v3.html
Apache License, Version 2.0	
Brief description:	- Proprietary software linking is allowed - Distribution of combination of software is allowed - Redistributing of the code with changes is allowed under another license - Compatible with GNU GPL
Source:	http://www.apache.org/licenses/LICENSE-2.0
Mozilla Public License (MPL), Version 1.1	
Brief description:	- Proprietary software linking is allowed - Distribution of combination of software is allowed - Redistributing of the code with changes is only allowed if the derivative is MPL-licensed - Compatible with GNU GPL
Source:	http://www.mozilla.org/MPL/MPL-1.1.html
Eclipse Public License (EPL), Version	
Brief description:	- Proprietary software linking is allowed - Distribution of combination of software is allowed - Redistributing of the code with changes is only allowed if the derivative is EPL-licensed - Not Compatible with GNU GPL
Source:	http://www.eclipse.org/legal/epl-v10.html

Appendix F: VistA discussion groups

At the moment, the most reliable source for communication within the community seems to be different Google discussion groups. There are the main group hardhats and various project related discussion groups, such as CCD/CCR project.

The following figure shows selected discussion lists (Google Groups) with their number of members (blue) and number of posts (red) (from 11-09-2009):



The number of members and posts of selected discussion groups are shown in the following table (from 11-09-2009):

Scope	Name	# Posts	# Members	Founded
Community				
	Hardhats	1,063	23,252	Jul-06
	VistA	179	343	May-04
	worldvista-services	81	110	Jul-07
	worldvista-adoption	N.A.	71	Jun-05
Policy & Advocacy				
	Open eHealth Collaborative	69	344	Jan-09
	openhealth	431	2407	Apr-05
	FOSS Health	N.A.	407	May-07
Active Software Projects				
	CCD-CCR	50	1132	Jan-08
	RPMS/VistA Billing	24	31	Sep-09
	Lab Re-engineering	48	283	Jan-08
	EWD and VistA	38	100	Jun-08

Appendix G: Red Hat and Fedora

(This appendix was written in collaboration with Michael Tiemann of Red Hat.)

Red Hat was perhaps the first commercial endeavor to create a successful business model based primarily on open source software. From the earliest days of the company, Red Hat promoted Linux, helped to make it available to the public for free, thereby increasing their potential market share. The value added by Red Hat was software to simplify installation and management, as well as customization and support. The greater the market penetration of Linux, the greater the potential market for Red Hat's proprietary software and services. Red Hat's success has been impressive, with about 1,000 employees in 15 countries, with annual revenue around US $200 M, and net income of about US $45 M for fiscal 2005.

Red Hat's success has not come without some painful lessons along the way, and here is where we begin to see the unique role of open source communities in software value chains. In 2002, Red Hat identified an attractive business opportunity in the enterprise server market. But because of the enormous commitments of time and money required to qualify a server in an enterprise environment, an enterprise release could include only features that were of proven stability, and the rapid pace of change typical of Linux and open source more generally was not well-matched to enterprise customers. So for good business reasons, based on a sound understanding of the market, Red Hat began to manage its open source Linux in a very conservative way, drastically limiting the changes and features that went into Red Hat releases.

The reaction of the open source community that Red Hat had cultivated was rapid, highly vocal, and overwhelmingly negative. No longer could the community get their code into a release, and they were not happy. As Eric Raymond might have said, there was now a community of developers who had no way of "scratching their own itch," and they felt betrayed. Red Hat had managed its assets in a way designed to serve its customers, but exerting this unilateral control over the open source asset enraged the community, and left them asking, "Who the hell do these guys think they are?"

After some internal discussion, Red Hat realized that it had made a serious mistake. Red Hat Enterprise was cut off from the contributions of the open source community. Keeping new functionality out produced the stability that enterprise customers needed, but at a high price. In response, there was a movement inside Red Hat to create a community release with no revenue responsibility, hosting and maintaining the community resources. The move was fairly controversial, since it was not clear, from a traditional point of view, that this was a responsible use of company resources. After a few false starts, Fedora was launched, and the benefits to Red Hat have been enormous.

Red Hat had at least two critical value relationships with the Red Hat Linux open source community, which were restored when Fedora was released: Red Hat benefits enormously from the innovation that the larger community produces, and benefits from the extensive testing and error reporting the community provides. The Linux security module (LSM), a framework developed by a number of contributors at the behest of the National Security Administration, provides an excellent example of both. Security features are often annoying to non-security-conscious users, since they tend to impact performance and convenience. So the decision was made to add LSM to Fedora, rather than RH Enterprise. The initial reception was highly unfavorable, since kernel modules in LSM typically had a default to disallow connections, which was too invasive. Rather than this simple default, Red Hat wrote scripts to deny things that touched the internet, to create a targeted policy.

Fedora core 3 shipped with LSM, and was much more secure than previous versions, and most surprisingly – the change was hardly noticed. After running long enough in Fedora to be confident of its stability and usability, Red Hat migrated the module and scripts into Enterprise 4. Fedora provided a crucial testbed in which the stability and acceptability of features can be tested in a wide variety of real environments.