

Software Strategic Research Agenda (SW-SRA)

Finnish Strategic Centre for Science, Technology and Innovation:
For Information and Communications (ICT) services, businesses and
technologies

Version 1.0
20th May 2009



Executive Summary

Software forms one of the key elements in global competitiveness for a broad range of industries. This has led to increasing demand for software related competencies and formation of new business value networks. These software-driven ecosystems and value networks have changed radically over the past years. A significant majority of software intensive businesses in Finland continue to sell their effort-based value and fall short in increasing value to their software assets and making most of them. At the same time, software and system complexity continues to increase, while cost pressures and availability of required competencies bring additional challenges to stay competitive.

Software Strategic Research Agenda (SW-SRA) direct research efforts to areas of identified significant value creation. It presumes that the foundation for successful businesses is built on capability to build solutions and services that exceed customer expectations, differentiate from the competition and/or build cost leadership. All these factors are based on building sustainable competitive advantage by managing the software assets systematically, building new core competencies, ensuring operational efficiency and quality in a broad perspective.

The ICT SHOK Software strives to create the foundation for the success of Finnish high tech and other software intensive businesses. By solving the challenges identified and focusing on the themes suggested, we progress towards the vision:

By 2015 the Finnish software industry will substantially increase the value of its software assets due to it's world-class capability and know-how to efficiently and competitively develop, deliver and use software competencies with a focus on defining, building and exploiting software assets and new ecosystems that have the largest sustainable value add for the global business.

By following the road we have defined, Finnish companies will become prime drivers of a global software intensive business ecosystem whose products have the highest added value in the world. Reaching the vision and to ensuring the future competitiveness of Finnish software-intensive businesses depends on taking the best use of the opportunities provided by software as part of products and services, networking with partners both locally and globally and being able to offer competitive products and services in selected global market place.

Contributors & acknowledgments

The editors of the present version are Pirkka Palomäki, Pekka Abrahamsson and Tua Huomo. The other authoring team members are Jussi Autere, Keijo Heljanko, Eija Kaasinen, Jussi Katajala, Timo Lukumaa, Tommi Mikkonen, Tomi Männistö, Markku Oivo,

Juha Röning, Reijo Savola, Pasi Tyrväinen, Andras Vajda, Raimo Vuopionperä and Kaisa Väänänen-Vainio-Mattila.

We would like to thank several people for their ideas and contributions. These experts include Janne Järvinen, Timo Kaltio, Risto Kero, Niley Oza, Jari Partanen, Veikko Seppänen, Matti Sihto and Kari Systä. We would like to thank also the Tivit and its board for the support and guidance. The SW-SRA has been prepared in close collaboration with broad participation from Finnish industry and academia. In the three workshops held in the spring 2009, we had more than 200 experts participating and identifying the future needs for Finnish software-intensive industry. We would like to express our sincerest acknowledgement for their valuable contribution and Panu Liira for facilitating the workshops.

TABLE OF CONTENTS

Executive Summary	2
1 Introduction	6
2 Background and Rationale	7
2.1 Global trends	7
2.1.1 Changes in user demands and user role	7
2.1.2 The Internet	7
2.1.3 Openness as a driving force for software	9
2.1.4 Diversity of software business models	10
2.1.5 Increasing security threats	11
2.1.6 Environmental awareness	12
2.2 Strategic Challenges & Opportunities	12
2.2.1 Benefiting from the “10 bucks an hour”	12
2.2.2 Need to differentiate with user experience	13
2.2.3 Exploiting effectively the web space	13
2.2.4 Making money from free software	14
2.2.5 Creating a security engineering culture	15
2.2.6 Achieving sustainable development	16
3 Vision for 2015	17
4 Research Strategy	18
4.1. Research methods	18
4.2. Projects & international collaboration	18
5 Themes	19
5.1 Strategic Theme: Operational efficiency	19
5.1.1 Description	19
5.1.2 Focus areas	19
5.1.3 Goals	21
5.1.4 Results	21
5.2 Theme 1: User experience	21
5.2.1 Description	21
5.2.2 Focus areas	22
5.2.3 Goals	23
5.2.4 Results	23
5.3 Theme 2: Web software	23
5.3.1 Description	23
5.3.2 Focus areas	24
5.3.3 Goals	25
5.3.4 Results	26
5.4 Theme 3: Open systems and software	27
5.4.1 Description	27
5.4.2 Focus areas	27
5.4.3 Goals	28
5.4.4 Results	28
5.5 Theme 4: Security development lifecycle	29
5.5.1 Description	29

5.5.2 Focus areas	29
5.5.3 Goals	30
5.5.4 Results	30
5.6 Theme 5: Green software development	30
5.6.1 Description	30
5.6.2 Focus areas	30
5.6.3 Goals	31
5.6.4 Results	32
6 Summary	33

1 Introduction

Over the past 20 years, impact of software in different products, systems and services has grown rapidly. This has led to increasing demand for software related competencies and formation of new business value networks. Many businesses are dependent on software assets and competencies and such companies face many challenges growing their business in today's global, fast-moving and competitive industry environment. At the same time, software and system complexity continues to increase. Cost pressures and availability of required competencies bring additional challenges to stay competitive.

Software is a key element in global competitiveness for a broad range of industries. It provides the means to differentiate the end customer offering by enabling competitive user experience and to produce systems that are more adaptable to different customer segments. Software is a driver for reducing manufacturing or service provisioning costs and it also enables new business models where services are available globally at the place of customer choice.

Software ecosystems and value networks have changed radically over the years. Open interfaces, open source and communities bring new opportunities and challenges in building new business ecosystems and benefiting from the potential of communities in product and service development as well as end customer involvement.

Environmental awareness has been an important global theme in the past few years and software plays an important role in reaching the goals for reduction of the energy consumption, optimizing the usage of the raw materials and providing environmentally friendly ways to offer new services.

Foundation for successful businesses is built on capability to build solutions and services that exceed customer expectations, differentiate from the competition or build cost leadership. All these factors are based on building sustainable competitive advantage by building new core competencies, ensuring operational efficiency and quality in broad sense. In many businesses, it is critical to build a sustainable base of software assets, while protecting the relevant IPRs or establishing a clear role in a given value network.

The future competitiveness of many Finnish businesses depends on taking the best use from the opportunities provided by software as part of products and services, networking with partners both locally and globally and being able to offer competitive products and services in selected global market places.

Software Strategic Research Agenda (SW-SRA) is formed to address the global trends and requirements from the business to combine the individual efforts in companies, universities and research institutes to form a new ecosystem to drive the competitiveness of Finnish software intensive businesses. Its goal is to form a new Finnish software ecosystem and to foster the creation of new start-ups which support new value networks.

2 Background and Rationale

There are several global trends that shape the future of software. These trends pose clear challenges to Finnish software-intensive industry but also offer a range of opportunities that can be exploited by 2015. The following subsections describe the main trends and the implications of these trends to Finland from the viewpoint of software industry.

2.1 Global trends

2.1.1 Changes in user demands and user role

Information and communication technology is increasingly embedded to everyday live of people both at work and during leisure time. The diversity of users is growing, and at the same time especially work-related applications are getting more complex. These trends set higher demands for usability and user experience of software.

The term User Experience (UX) was launched in early 2000's to describe user's holistic interaction with a product, its additional services, and even with the company producing the product. UX takes product design a leap beyond usability. In UX perspective, user's task performance and satisfaction is extended towards broader scope of supporting human motivations, values and needs – both pragmatic (goal-oriented needs) and hedonic needs. Products need to provide users with positive emotional experiences.

Today people use routinely web services to access information but increasingly also to publish their own content. User role is changing from passive user to active content creator as users get used to ICT. This trend will continue to grow and user role will be extending from content creation to service design. User role as a designer may show as software products that are not ready made but are designed to be tailored by the users who create their own usage practises for the software. Another trend is that companies are increasingly providing services rather than traditional software-intensive products. This trend creates a need to improve the connection to customers and end-users and creates the possibility of utilising the experience and innovativeness of users in the service design and development.

2.1.2 The Internet

The World Wide Web – or, in the more general sense, the Internet – has undergone a number of evolutionary phases. At the high level, the evolution of web pages has advanced from simple, “classic” web pages with text and static images to animated multimedia pages with plug-ins to *Rich Internet Applications* (RIA), as shown in Figure 1.

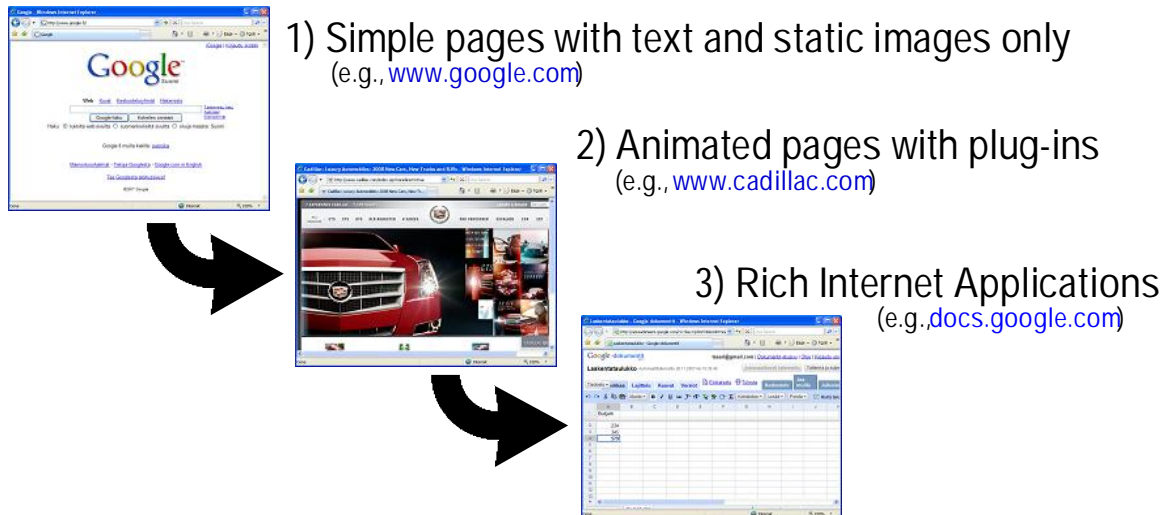


Figure 1. Evolution of the Web¹

Note that the three phases discussed above are not mutually exclusive. Rather, web pages representing all three phases coexist on the Web today and even classic web pages can provide good user experience as Google has proven. The majority of commercial web pages today represent the second phase. However, the trend towards web applications is becoming increasingly common, with new web application development technologies and systems introduced frequently.

Web software is in several ways different from conventional software. Some of the main differences are presented in Table 1.

Web software	Conventional software
1. Documents	1. Applications
2. Page / form oriented interaction	2. Direct manipulation of software
3. Managed graphics	3. Directly drawn graphics
4. Instant worldwide deployment	4. Conventional deployment (installation needed)
5. Source code and text favoured	5. Binary representations favoured
6. More frequent releases possible	6. Less frequent, larger releases
7. Target environment not designed for applications	7. Target environment specifically designed for applications

Table 1. How the web software differs from the conventional software

There are some common themes that have started to emerge².

¹ Taivalsaari, A., Mikkonen, T., Ingalls, D., and Palacz, K. Web Browser as an Application Platform: The Lively Kernel Experience. Sun Microsystems Tech report TR-2008-175, Sun Microsystems, January 2008.

² Taivalsaari, A., Mikkonen, T., Ingalls, D., and Palacz, K. Web Browser as an Application Platform: The Lively Kernel Experience. Sun Microsystems Tech report TR-2008-175, Sun Microsystems, January 2008.

- *Trend toward dynamic languages* [³]. Most of the systems above rely on dynamic, interpreted languages such as JavaScript.
- *Technology mashups*. Many current web systems are hybrid solutions in the sense that they combine various existing, sometimes previously unrelated technologies. For instance, Ajax [4] is a combination of a number of existing technologies – HTML, CSS, DOM, JavaScript, asynchronous HTTP networking and XML protocols – rather than a uniform, coherent application platform. In this regard, these systems resemble the content mashups that are common on the Web today.
- *Dependence on tools*. Most of the web systems are dependent on tools and integrated development environments. For example, Ruby on Rails introduces a set of naming conventions that are automatically applied by the development tools.
- *Non-conventional and evolving software engineering principles*. For example, the JavaScript language has very limited support for modularity or information hiding.

2.1.3 Openness as a driving force for software

When the software systems were first taken into use in numerous contexts, it was common that a number of individual, closed systems that were incapable of working together were used. However, it has become a necessity to take compatibility with existing systems into account when developing new applications, to the extent that integrated systems that provide an access to almost all facilities are needed. This in turn requires defining open interfaces that can be used for integration between different systems.

Software systems are also innovated and developed in increasingly open environments. That is, existing systems are combined with open source components, new ones are created by composing open source components, and systems are more increasingly required to interact with open source systems.

Increased openness is an ongoing trend that drastically changes the landscape of software engineering and development of software-intensive systems and services. The change radically affects existing business ecosystems. The business dimension has multiple facets. One facet includes free components available for anyone to use. However, this also threatens the businesses based on proprietary, closed solutions by a potential devaluation of the software assets by providing the customers with free alternatives.

Another facet of the business dimension is that open source provides communities of (at least potentially free) developers as a workforce to the development of software they find interesting and meaningful to build. The important characteristic of these communities is their high independence of any traditional organisational commitments. Thus, although developing software for free, the communities may be hard to persuade to work for anything particular if that is not in the interests of the key members. Moreover, the role of communities has been rapidly changing from free, even ideological participation to a direction where community behaves more like a company, and open source is only a licensing model.

³ Paulson, L.D., Developers shift to dynamic programming languages, *IEEE Computer*, Vol 40, no 2, February 2007, pp. 12-15.

Finally, open source has become a natural way for companies to collaborate and share development effort. This non-predatory way to work on a common software system is especially promising in cases where a number of companies work in the same large project.

2.1.4 Diversity of software business models

Over the last 20 years, software's impact on different products, systems and services has grown rapidly. This has led to an enormous increase in software complexity as well as increasing demand for changes in software business models. Software intensive companies face many challenges in growing their business in today's global, fast-moving and competitive industry environment.

The contemporary software business environment shows major changes due to several factors reducing the threshold of adopting new solutions. These include:

- *Software products.* Most common enterprise systems (e.g. industry-specific ERP systems, CRM etc.) have been packaged into standard low-cost products.
- *Processing capacity.* Moore's law demonstrates itself through availability of multi-core hardware and operating systems making it available for common software.
- *Communication capacity.* Gilder's law demonstrates itself through availability of sufficient bandwidth for executing systems remotely in a computing cloud enabling low-cost hosting and ASP.
- *Web-based software development.* Use of web as the software platform enables to provide a single solution for a high number of potential users.
- *Software-as-a-Service (SaaS) and software-on-demand.* Using standardized software remotely gives relief from maintaining IT infrastructure and personnel in small enterprises and paying for use avoids high entry costs of software products.
- *Open source.* Most common platform functionalities (e.g., databases, web programming) and products are available as low-cost and society-maintained OSS packages.

Through these changes benefits of software are available for much higher number of customers than a few years ago. Also the number of software providers is increased through availability of vendors through the net. In this context the customer-oriented software development and understanding of customers' business and customers' customers needs became key factors. This stresses the importance of:

- *Agile enterprise using agile SW methods.* Fast re-creation and evolution of solutions for capturing customer feedback and continuously improving the solution (or failing fast). Valid both for customer-specific systems and products / OSS used for SaaS.
- *Community-based operations.* Close interaction with user community driving the direction of the market and a community / network of business partners.

Collaborative Models & User Experience

The importance of different collaborative models and user's participation in the sw development cycle is gaining momentum. Companies are increasingly communicating with different user communities to understand and even foresee user requirements and to provide good user experience which will be a key differentiator on the market. Designers and developers need to work in multidisciplinary groups and must apply information from different sources more than ever before. The business models, delivery models, plans and

assets protection must be taken into account and the companies must optimise for an effective collaboration within and outside the company. This requires collaboration between highly skilled internal, external and virtual teams and user communities and these new cooperation models will require new approaches to the business models and value networks. Open innovation models will be even more popular, and again affect business models.

Software and knowledge as an Asset

Software, tools, platforms, processes and knowledge can be regarded as valuable, but not yet fully utilised, assets for many companies. Managing, protecting and exploiting these assets through legal or other means (selection, maintenance, validation, delivery and applying the assets of the other companies or e.g. open source communities) will play a key role in determining the operational performance and profitability of software intensive companies. The growing importance and value placed on the software assets, information and ideas in the global information economy, coupled with the fact that these assets are now generally stored electronically, creates an increasing business opportunity to the companies and therefore a need to understand the new business models, methods and legal issues around it. For software companies digital knowledge management opens new significant business opportunities that are not yet fully understood.

Software as a Service (SaaS)

SaaS is a new software delivery model in which customers get access to applications on an on-demand basis, without having those applications deployed in their networks or without paying any up-front capital expenses. From software industry point of view the SaaS providers maintain control and protect against illegal use. Although the SaaS business model has entered to the software business by storm, for many companies it is still unclear. It is also unclear what systems should be implemented as SaaS, how to combine SaaS with the open source approaches and what other strategic and legal changes and decisions companies need to make in order to fully utilise SaaS with the other business and delivery models. Especially for telecommunications industry the potential of SaaS is not yet fully understood though it may provide significant competitive advantage for Finnish companies.

2.1.5 Increasing security threats

Security and related issues have become a core issue of high practical relevance in software development. The lack of appropriate security solutions might have serious consequences to the business.

Software complexity and number of interfaces in devices is increasing, resulting in an increased need for the management of the systems, services and applications. New security threats are introduced when software intensive systems become more versatile and more complex. At the same time, the user has more responsibility for security and privacy issues. The user-managed and system-controlled security elements will have to work hand in hand. In addition, one can learn from the history of technological development that software (or at least parts of it) are often re-used for purposes for which they were not originally developed, causing major security problems in many cases.

Software quality problems, wide impact vulnerabilities, phishing, botnets and criminal enterprises have proven that software and system security is **not just an add-on** despite the past focus of the security industry.

2.1.6 Environmental awareness

Focus on sustainable development is an emerging strong trend. Consumers are becoming more aware of environmental issues such as global warming and pollution. This reflects directly into their consumption habits. Consumers not only want products that satisfy their needs, they also want these products to be environmentally friendly. This change in attitudes reflects also to legislation, because politicians are under pressure to create legislation that supports sustainable development. Another aspect of this trend is the increasing prices of energy and raw materials. Consumers want products that use as little power as possible and therefore also industries must consider ways to minimize their power consumption. Sustainable development requires environmentally friendly products and solutions that are developed and manufactured by using minimal amount of energy and natural resources.

2.2 Strategic Challenges & Opportunities

2.2.1 Benefiting from the “10 bucks an hour”

The challenges and opportunities of globalization are forcing companies to become more nimble, using an increasingly geographically-dispersed and virtual workforce to remain competitive. For Finnish industry the globalization has many meanings depending on the context. We should not fight against the trend, but seek ways to take the best use of it, while growing our own core competencies, focusing on the high value creation and retaining the core assets in Finland.

Cost of software development is often measured in hourly cost (or rate) not factoring in the productivity, cost efficiency or quality of the outcome. The challenge is evident. *Why should we keep software development in Finland, if we can buy it for 10 dollars an hour outside from our borders?* We should see this as an opportunity to decrease costs in certain activities. However, we should focus more on retaining the relevant core competencies and value creation in Finland.

While software development takes place in many countries, innovative software is not created everywhere⁴. Domain knowledge and proximity of potential lead users create conditions for innovation. However, cheap hourly rate does not guarantee a low total cost of software development. A great mass of development resources does not guarantee high quality software. Research has established that the single greatest challenge in global software development is communication⁵ and lack of critical domain understanding.

Software generates competitive advantage to many products and services. For example, in the manufacturing sector in Finland, about 30% of the total R&D costs are from software

⁴ A. Arora, M. Drev & C. Forman (2009). The extent of globalization of software innovation. Communications of ACM, Vol 52 (2): 20-22.

⁵ S. Komi-Sirviö & M. Tihinen (2005). Lessons learned by participants of distributed software development. Knowledge and Process Management, Vol 12 (2): 108-122.

related research and development⁶. This implies that software is already the key ingredient to the future competitiveness of manufacturing industry in Finland. While some parts of the software components are likely to be acquired outside in the future, the highest value added software related, domain specific and value chain management competences should remain in Finland.

The opportunity for Finnish industries is to transform from project based software engineering to leading the value creation chain. This consists of managing the full lifecycle, leading in the industry specific expertise as well as understanding deeply customer requirements. Furthermore, we need to be able to use effectively Finnish core technologies, low cost development, communities, open source and automation to focus on the highest value of the software asset and customer value creation.

2.2.2 Need to differentiate with user experience

As the selection of different ICT services is growing, and people have choices, services that do not provide good user experience will be discarded. Thus user experience is no more an additional attribute but a necessity in any software product.

In current distributed, global software development environments the requirements for work efficiency are increasing. In this kind of challenging development environment, creation of high-quality user experience is quite demanding task. However, users and customers require good user experience (UX) and UX design should be integrated efficiently to the development process.

Integrating UX design and evaluation in new SW design approaches such as agile, lean and OSS approaches provides a good opportunity to improve the quality of software products, still responding to the requirements of efficient and fast development.

New software paradigms such as mash-up services and cloud computing are changing the ways how people find and access services. User experience should be considered when developing these paradigms to ensure user acceptance of the future computing environments. To ensure both user acceptance and commercial success, business and user experience perspectives should be combined.

High design potential can be seen in utilising the expertise of ordinary users who are the best experts of their own life and work practises. Better products and better user experiences can be achieved as user role is changed from design object to an active design partner by utilising and developing design methods with active user participation such as user-driven design.

2.2.3 Exploiting effectively the web space

The web has fundamentally altered several businesses. Perhaps surprisingly, the software industry has been one of the industries where the effect of the web has been the least. However, the software industry is currently experiencing a paradigm shift towards web-based software, evidenced by the introduction of numerous web development systems,

⁶ Rönkkö et al. (2009), to appear.

such as Google Web Toolkit [⁷], Adobe Air [⁸] and Microsoft Silverlight [⁹] for client development and the numerous platforms introduced for cloud computing. In the future, applications that were previously written for specific operating systems or CPUs will be written for the Web, to be used from a web browser from anywhere on the planet. In web terminology, such applications are referred to as Rich Internet Applications or simply as web applications.

In service development that reuses code and content from other web sites, the Web essentially becomes a distribution media for software components consisting of objects and code located anywhere in the world. Components can be modified on the fly without static compilation, linking, or “stopping the world”. In general, we intend to create an environment in which the mechanisms needed for services as mashups are an integrated, built-in feature, rather than an afterthought or a combination of hybrid technologies as in today’s systems.

The transition towards web-based software will dramatically change the way people develop, deploy and use software, making software distribution as easy as the distribution of documents and other services on the Web already is. In essence, the Internet becomes an omnipresent media for distribution, and it is up to the developers to figure out suitable component models to be used in application development. In essence, service orientation that combines data, code and content from several sites is becoming a new, emerging software development paradigm whose limits should be studied more carefully.

2.2.4 Making money from free software

Three fundamental business models for open source software have traditionally been support/service provider, multi-licensing, and packaging and distribution. They are addressed in the following.

Support/service provider. A number of large companies – such as IBM and Sun – have built an extensive portfolio of open source systems that form a baseline for consulting, which in turn is a major business for both of the companies and justifies the participation of company personnel in open source communities.

Multi-licensing. One of the prominent ways to combine open source and commercial development has been multi-licensing. Then, there are several sets of rules that are applied depending on the licence. For instance, when using the open source license certain obligations may result which are not applicable when using a commercial development license that must be acquired with money.

Packaging and distribution. One of the first business models used in open source has been packaging and distribution of already existing systems, which saves the user the trouble of collecting all the required components and verifying that they are compatible. This model, used for instance by Red Hat in their Linux distributions, has also resulted in stable starting points for the development of applications on top of Red Hat Linux distributions.

⁷ <http://code.google.com/webtoolkit/>

⁸ <http://www.adobe.com/products/air/>

In addition to business models applicable by companies, one could also claim that there is also other models that can be applied by owners of a company. For instance, establishing a lively developer and user community for a system backed by a company, and then selling the company to a large client, which wishes to ensure that the best developers are always available, has been common. Examples of such acquisitions include for instance by Trolltech, which was later acquired by Nokia, and MySQL which in turn was acquired by Sun Microsystems.

The above business models have emerged from practice and needs of the industry, and represent the very first steps of commercializing open source software. However, there is a lot of freedom to create new innovative business models created by combining the different elements of already existing business models as well as totally new elements, such as user and developer communities. Furthermore, to a large extent open source can be applied best by one-man companies which concentrate on serving one client and participating one community, and international giants that can afford adequate support stuff, including lawyers. However, small and medium enterprises face difficulties in benefiting from open source due to gray areas in the interpretation of licence terms, lack of resources in active participation in communities, and lack of knowledge regarding the forms of participation in open source development.

2.2.5 Creating a security engineering culture

There is strong need to achieve a systematic and proactive security engineering culture in software intensive systems industry, integrated to all relevant software development and business management and development activities. Nowadays, security engineering includes still diverse ad hoc activities such as risk management, cryptographic algorithm design, information security management, computer security, network security, security oriented testing and monitoring, and so on.

The envisioned new security culture will include appropriate tools and methods to support the culture and to raise security awareness. The security engineering practices will be an integral and built-in part of the software lifecycle management. The culture will include also means to bridge the gaps between different stakeholders of security – security analysts, product developers, business developers and risk management personnel. Without this collaboration, the security solutions suffer from too narrow thinking and do not offer appropriate countermeasures for relevant security threats, taking into account the exposure and impact dimensions of them.

The new culture will promote true proactive security engineering, where security IS actively managed during the whole life-cycle. Active proactive security engineering will affect architectures and technological choices as early as possible. Appropriate means to offer evidence of the security level of products, services and systems and to manage the vulnerabilities will be deployed, together with security assurance tools and methods.

⁹ <http://www.microsoft.com/SILVERLIGHT/>

The most important security countermeasure is to increase security awareness. By creating a proactive security engineering culture with practical tools and methods, security awareness among the software developers and other stakeholders will be increased as a side product. Furthermore, specific awareness raising actions (education and training) will help to bring the situation ever better.

Finland as a small yet technologically skilful country with a lot of security competence has the potential to be the cradle of this new culture.

2.2.6 Achieving sustainable development

The ICT industry can provide solutions that will help other industries and human activities reduce their environmental footprint - prime examples already include reducing travel by using communication technologies, optimizing transport costs, or controlling energy consumption in industrial processes. There is significant business potential for the ICT industry in such innovative solutions. As software is all the time more important part of ICT products, the way it is developed and how it performs will have more and more effect on how sustainable ICT products are.

ICT industry itself will be facing the same sustainability challenges as other businesses. Main challenges will come from energy costs and resource usage. ICT products require large amounts of energy for their operation and also for cooling. As the price of energy is increasing, ICT industry must focus on reducing the energy consumption in order to keep development and operational costs down. From resource point of view the prices of raw materials are increasing and the materials used in ICT equipment generate hazardous and non-hazardous waste. Waste management becomes an issue at the product end of life, when the product is disposed of. Although software does not have direct impact to the equipment, portable and platform independent solutions offer a possibility for switching to more environmentally friendly infrastructure. Another solution is to develop software that can utilize the existing infrastructure more efficiently, thus prolonging the equipment lifetime.

Other challenges are non-technical ones. Legislation sets targets and penalties regarding power consumption and waste management. Meeting legal requirements is usually mandatory in order to enter a specific market. As the end users become more aware of environmental issues, they will favor suppliers, who are dedicated to sustainable development. Suppliers in turn start to demand sustainable solutions from the manufacturers. New software solutions can be developed for finding and offering sustainable solutions. Yet another challenge is the business challenge: how ICT products can be developed in a sustainable way so that the products are still competitive and profitable?

3 Vision for 2015

SW-SRA bears an ambition to create the foundation for the success of Finnish high tech and other software intensive businesses.

The vision is formulated as follows:

By 2015 the Finnish software industry will substantially increase the value of its software assets due to it's world-class capability and know-how to efficiently and competitively develop, deliver and use software competencies with a focus on defining, building and exploiting software assets and new ecosystems that have the largest sustainable value add for the global business.

4 Research Strategy

SW-SRA provides the framework for projects enabling methods, tools and business models for Finnish industry and society to be able to exploit software in order to gain a global competitive edge when we go towards 2015. The focus of innovative application development itself is mainly included in other ICT-SHOKs. The focus of SW-SRA is in the infrastructural elements that enable the effective development and deployment of innovative software based products and services. SW-SRA driven projects create and develop a critical mass of software assets, competencies and ecosystem in Finland that provides a competitive edge at a global level.

The research will take into account the needs of other than pure software applications of software companies. These include, but are not limited to, the needs of embedded systems, machinery, health care, and public services to mentions a few. Software is a vital element in all activities in the society and this is a driving force of the SW-SRA.

4.1. Research methods

Metaphorically, SW-SRA provides the highways and supports effective building of various types of vehicles on the roads. The research in SW-SRA is thus driven by industrial needs, characterized by an extensive industry involvement, striving for a solid empirical research approach. The researchers should retain from only proposing new methods and tools expecting the industry take them into use without a clear deployment strategy.

The research is an enabler also for other SHOKs (FIMECC, etc.). In this way SW-SRA is by nature different from other ICT-SHOK SRAs. It can be seen as a horizontal SRA offering an opportunity for extensive cross-domain collaboration. SW-SRA projects seek to foster close cross-disciplinary collaboration in all possible dimensions in industry and research.

It is preferable of the research methods and strategy to support and foster open innovation. Research can also develop new methods to support innovation as part of software development. The research will provide a mechanism for collecting, disseminating and exploiting the developed assets. As an example, cross-industry best practices development and deployment is one such an asset. The research creates software ecosystems across different software intensive organizations and sectors. The exploitation strategy requires a clear path from ideas to practical use.

4.2. Projects & international collaboration

The SW-SRA welcomes projects and initiative for themes as long as they bear the ambition to reach the vision and goals defined in the SW-SRA. We do not seek to create a single large monolithic project for the whole SW-SRA, but rather to establish a dynamic project portfolio with clear breakthrough targets and ambition.

An essential element of the research is strong international collaboration (joint research, researchers' mobility, etc.). Potential collaboration initiatives and programs include the pan-European platforms (ITEA2, Artemis JTI, EU FP7), other high level research networks, standardization bodies, and innovation hubs globally.

5 Themes

This section describes the breakthrough themes needed to increase the profitability and productivity in Finnish software-intensive business context. The section is broken down in two principal elements: Strategic theme and theme. The strategic theme of SW-SRA is operational efficiency, which underlies all other themes but also acts as an area of study of its own. All other teams bear equally significant relevance. The themes are interlinked and all of them strive to offer and identify focus areas and ambitions that help to reach the vision described for 2015. Some of the focus areas are focused on shorter term benefits while others support the longer term goals.

All the themes are described in a uniform fashion including a description of the theme, critical focus areas for Finnish software intensive industry, goals for the theme and a high-level outline of possible results.

This section should not be seen as a project plan but rather as roadmap towards the vision. The SW-SRA projects shall execute the defined roadmap.

5.1 Strategic Theme: Operational efficiency

5.1.1 Description

Still, in 2009, the great majority of software is being built by humans. Human capital is the most important source for the analysis, designing, implementing, testing and maintaining of software services. As such, humans continue to introduce misinterpretation, logical defects, errors and other unwanted features into the software systems and services. While the development of new software continues to be effort-consuming, by 2015 significant advancements can be made in several areas. These include automation of routine tasks, supporting software development workflows with automated tools, validation of business requirements, and enterprise-wide agile/lean processes & methods.

Operational efficiency deals with means to improve productivity in software development. The means include software development technologies and tools as well as supporting processes, methods, techniques and practices enabling an efficient, market-driven and durable software development.

5.1.2 Focus areas

Enterprise level agile/lean processes, methods and tools. Agile methods and thinking strive to meet the changing market needs and customer expectations. While agile methods were introduced a decade ago including a concern whether they could be applied in larger development contexts, the Finnish lead European research (e.g., Flexible Global Product Development and Integration, FLEXI – www.flexi-itea2.org) has firmly established that significant benefits can be gained from their use. It has also been discovered that agile methods and processes do not alone suffice. The impact of agile transformation when scaling up the ideas is far beyond the R&D settings. Lean thinking together with lean software solutions is likely to offer the needed step to develop enterprise wide agile models. There is a significant lack of tools in the market place that would support this new form of development. Valuation of requirements as the means to prioritize the development and an

early validation of them, require an extensive attention as well. Enterprise level agile/lean processes, methods and tools should also extend to new application domains such as the development of safety critical systems.

Partnering & collaboration models and tools. Software continues to be developed in the global space. New forms of networking require novel views on facilitating collaboration across stakeholders within the organization and outside. This will have a large impact on the way contracting takes place but also how to position oneself in the value chain in order to sustain the competitive edge in the future. This area of focus will have an impact on the subcontracting practices and bears an opportunity to offer Finland the needed competitive edge.

Automation of routine tasks. Rather than outsourcing the work to low-cost resources globally, the companies should strive for significantly higher levels of automation in all developmental phases. Since testing continues to be the most effort-consuming software development phase summing up to 40-70% of the total development costs, the automation of testing should be particularly striven for and considered at several levels including also the higher-abstraction level testing such as model-based testing. Automation tools may require the inclusion of machine-processable metadata. This aims at enabling automation beyond routine tasks such as automated discovery and integration of Web and open software components. This line extends also to tools supporting automatic collecting various contextual information related to software components under development as well as providing efficient ways for the components' developers to record other relevant metadata.

Leaner products & management of large legacy assets. Software systems continue to evolve and starting from a scratch is rarely a viable option in a turbulent business environment. Management of legacy systems bears significant relevance in the near future as well. While history has shown that software systems grow in size and complexity, adding new functionality has often become exceedingly expensive. A viable option is to develop technologies, which enable the controlled reduction of problematic or unused parts the system while still retaining the required non-functional qualities. Methods should be developed to enable early detection of areas, which grow problematic during system evolution.

Social media in software context. Finally, software development shifts towards ways we have not seen before. Knowledge transfer from business stakeholders and end users to software developers has always been a time-consuming and error-prone activity leading to a need to raise the abstraction level in software development to enable effective involvement of non-programmers to the software process. By 2015, the penetration of social media is likely to have extended to software development and use context more profoundly than we have thought of. Ideas like Facebook software changes the way the software development is perceived and should not be overlooked in coming few years time. This may work as a way to raise the abstraction level in a novel way.

5.1.3 Goals

Operational efficiency strives to significantly improve Finland's capability to develop software in a more productive way making the software businesses lean and agile, increasing the level of automation and developing the capability to do more with less.

There are several trends that support the reaching of these goals. These include the transition from strong process- driven development to higher levels of autonomy in organizations, teams and individuals. Detailed processes & guidelines are shifting towards frameworks, principles and mental models. Management and control thinking will be replaced by leadership & responsibility thinking. Software and service design shifts from attempt to have a complete understanding to trade-offs with partial, emerged, opportunistic business understanding. The current ad-hoc automation shifts towards a strategy & context driven systematic automation. Finally, end-user driven development is likely to take place in the form of involving social media to software development context.

5.1.4 Results

The work in the strategic area of operational efficiency yields new methods, processes and tools that enable a significant reduction of feedback time at all organizational levels enabling individuals, teams, organizations and networks to better respond to changes in the market place and customer needs. While the whole production becomes leaner, also as the result, the products and services have the capability to be leaner as well. This bears significant opportunities to Finland and abroad as it leverages new opportunities to more effectively deploy new functionality without the risk of breaking the existing system. A higher level of automation can be defined and implemented, which enables the strategic decision making at least in the area of software testing.

5.2 Theme 1: User experience

5.2.1 Description

In 2015, people will be surrounded by a growing amount of interactive products and services. In our vision, People can utilise contextually relevant services that are easily available and provide efficient and satisfying UX. Products will increasingly have elements and characteristics that support both pragmatic and hedonic aspects of product/system usage. In addition to fluent and flexible task performance, users will be motivated by the high-quality product aesthetics and features that support users' basic needs such as skills development, social relatedness and self expression.

In consumer products, the aim of UX design is to arouse positive emotions in users that go beyond satisfaction – thrill, fantasy, joy and fun. Social media is an example of product category that is growing and will enable many emotional experiences within users' social networks. Personal products will be increasing their value as tools of self-expression and development of users' skills and knowledge. Embedded systems will hide intelligence in various products, and interaction with these systems will need to be an ambient, yet pleasing part of people's everyday lives.

In the work-related product categories, efficient task completion will still remain an important goal. However, to achieve optimal UX also in the information system side, the broader scope of human motivations and performance will be accounted for. For example, the workers' ambitions in improving their job performance can be supported by social interaction with their fellow workers. It is also important to let people plan their own work, for instance letting users personalise their work-related systems to better suit their own needs. Even efficiency-oriented fields such as machinery automation will benefit from the experiential aspects of interaction design.

User experience research theme is focused on developing and applying human-centred and human-driven design and evaluation methods that target high-quality user experience. UX design and evaluation methods will be integrated to new software design approaches such as agile and lean methods. The theme will study how new software paradigms such as mash-up services or cloud computing will affect user experience. The methodological development will be carried out in close cooperation with the other themes.

5.2.2 Focus areas

User experience theme will be focused on the following focus areas:

Development of UX design and evaluation methods that support new SW design approaches. This focus area aims to develop UX methods that support Agile, Lean and Open software development approaches. Agile software development will be supported with fast and light ways to evaluate SW with end users. Open systems and software will be supported with user-driven design methods that facilitate user role as a part of the design team. Methods to define and maintain design vision are also important in the development of open systems and software. In Lean software development UX methods support the identification of the most important software features that should be focused on in the design. This focus area is also developing and applying methods that combine the viewpoints of productivity, user experience and quality. A combined methodology is needed to be able to study parallel these complementary aspects.

User interaction paradigms for new SW paradigms. This focus area studies how user interaction should be facilitated for best possible user experience in connection to new software paradigms such as service orientation in design, mash-up services and cloud computing. The users will need interaction tools that support getting an overview of available services and easy ways to access relevant services. Interaction tools are also needed to facilitate services that the users themselves can set up and tailor to their own needs. Systematic user interaction design will be supported by defining user interaction design patterns that facilitate good user experience. User interaction paradigms will be analysed to identify their implications on not only the user interface design but also to other levels of the software, including the level of software architecture design.

UX design needs in other thematic areas of SW SRA. It is also necessary to link the UX theme to the other SW SRA themes by studying user experience in specific SW areas. Possible research fields could be User friendly security solutions, Web'd SW from user point of view and Green end user applications. The implementation of these studies would

be most efficient if UX studies were integrated to the individual research projects in each theme area.

5.2.3 Goals

The main target of UX theme is to ensure the success of software services, so that people can utilise contextually and personally relevant software services that are easily available and provide efficient and satisfying user experience. This is achieved by making user participation in software design "de facto" and developing methods to ensure high-quality UX in agile and lean software development processes. UX themes will establish best practices in which developer communities include active user representatives. The best practises will be applied within the SW SRA in close cooperation with the other focus areas of the SRA.

In more details the UX development will be made systematic, and UX will be studied not only in connection to the design of individual user interfaces but on all levels of software design, as all the design decisions may affect user experience. UX will be considered in all stages of SW development and UX measurements will be made a part of SW life cycle.

In addition business goals, quality and UX targets are recognised as inherent part of SW development and these three aspects are studied and developed in parallel.

5.2.4 Results

The outcome of UX theme will be that Finnish software design is well-known for excellent user experience. This is achieved by established methods and practises for understanding and foreseeing user and customer needs and to innovate and design together with users. More concretely, the key results of the UX theme will be: ::

- User experience design and evaluation methods that support agile, lean and open SW development approaches.
- User interaction paradigms and design patterns for new SW development paradigms such as cloud computing and mash-up services, and best practices to integrate them to service and software design.
- Methods that integrate user experience, business and quality studies during different phases of SW development.

5.3 Theme 2: Web software

5.3.1 Description

Compared to how dramatically web usage has increased since the early 1990s, it is remarkable the web browser has remained merely a document-viewing tool. In general, web browsers are already so widely established that it may seem rather difficult to try to make any significant changes in the design or the behaviour of the browser. However, given how quickly the use of web applications is increasing, it is quite possible that web browsers will have to adapt to accommodate a more application-oriented approach, in addition to the document-oriented approach that dominates the Web today.

In addition to browsers, an essential component of web-based software is the development of back-end systems, or computing “clouds”. The development of cloud computing systems however can be subjected to similar concerns as the web browser, since the two are commonly used together. However, the development of cloud computing systems is more flexible, because their maintenance is usually a responsibility of trained and dedicated personnel. Moreover, there is an opportunity to use more sophisticated platforms, which in turn provide more extensive development opportunities. Despite the added flexibility, there has been little reported work on establishing software engineering principles for either cloud computing systems or developing applications that benefit from cloud computing.

5.3.2 Focus areas

There is a fundamental contradiction regarding the development of web systems and conventional applications. In particular the following issues associated with the development methodologies and web software arise.

Management of the collection of components and data coming from different sources and hosted in different places. At the moment nobody alone is – or even can be – in control of the total functionality of a complex web application combining data, code, and content from different sources. There is a need to define new types of interfaces and ways to manage and update them.

Modular development and upgradeability. We cannot stop web, and we cannot even be in control of all the components we use, and we cannot assume synchronous upgrades. Thus, we need to develop dynamic modularity and means to upgrade in dynamic fashion.

Design methodologies benefiting from extreme dynamicity. By extreme dynamicity we mean systems where all bindings are postponed to run-time and the functionality, structure and behaviour of the system can change in run-time. Development of small and throw-away software benefits from dynamic environments, but problems appear when software grows and is developed and maintained over longer period of time. We need architecture models and design methods to bridge the dynamic worlds of software living in the Web and reliability requirements of traditional software and software processes.

Development and testing issues. The transition from conventional applications to web applications will result in a shift away from static programming languages such as C, C++ or C# towards dynamic programming languages such as JavaScript, PHP or Python. Mainstream software developers are often unaware of the fundamental development style differences between static and dynamic programming languages. Developers need more knowledge about the evolutionary, exploratory programming style associated with dynamic languages, as well as agile development methods and techniques that are available for facilitating such development. In the testing area, there is an increased need for code coverage testing to ensure that all the parts of the applications are tested appropriately in the absence of static checking. Some of the problems can also be solved by tool support. For instance, static verification tools, such as jslint [10], can be valuable in checking the

¹⁰ <http://www.jshint.com/>

integrity of an application before its actual execution. This are also includes methods and tools that able to, e.g. through web-exposed metadata, discover the components that match an application developer's needs, and able to automatically generate code needed for accessing those components functionality.

Interoperability and compatibility. In order to improve compatibility, an independently developed compatibility test suite, similar to the test suites available for the Java platform, would be very valuable. For each new browser or cloud computing feature, a reference implementation should also be made available. Having an independent third-party compatibility test organization might also help. If such an organization were available, new versions of web software could be subjected to third-party compatibility testing before the new versions of the browser (or other web related facilities) will be released to the public.

Deployment issues. One of the main benefits of the Web is instant worldwide deployment: Any artefact that is posted on the Web is immediately accessible to anybody in the world who has a web browser. This "instant gratification" dimension will revolutionize the deployment and distribution of software applications, and will imply various changes in the business model of almost everyone in the software industry. One of the main challenges in the deployment area is to define a design framework and architecture model that address the fundamental changes in the nature of applications that we discussed above: applications that are always on, the ever-shortening release cycles, and the perpetual beta syndrome.

Diversity in access regarding both connection and devices. It is assumed that share of mobile users is increasing. This adds diversity to input/output devices, computing capacity and connectivity. We need to create solutions that allow use from pocket-size device, slow and varying connection, and consumes as little computing capacity and battery as possible. Furthermore, it is possible to share resources available in individual devices.

Internet of things. The items we mash-up in future web software are not just files and content but also items and objects around us. Furthermore, intelligent and autonomous behaviour is appearing around us, which can further be incorporated in applications. In the long term, the concept of web sites will be extended to include also resources available in individual devices, resulting in exponentially increasing number of web sites.

Intellectual property rights. In today's web applications, the value is typically in service, content and business model, and at least at the moment it is seldom in actual code. Will this ever change, and if it does how to protect such IPR remain future research questions.

5.3.3 Goals

The transition towards web-based software will not be easy, and nor will it happen overnight. Companies will have to adapt to new practices also in the development, and the range of potential solution architectures will increase tremendously. When doing so, the balance between parameters such as technical artefacts, development time, and maintenance costs will be increasingly important, and must be taken into account in the management, development, offering, and use of web-based software.

The most important goals regarding this topic is the definition of engineering principles for web software, as well as the documentation of readily available alternatives, including in particular the development using the facilities of so-called Open Web [¹¹] and W3C's standards and recommendations, plugin technologies, and different flavours of cloud computing [¹²]. Moreover, the development practices should be scalable and well-suited for interaction with both conventional software development and the development of web applications.

5.3.4 Results

The long-term outcome of the result is the establishment of Finland as a leading country in the development of web software. This includes at least the following dimensions:

Leaner development, deployment, and maintenance processes. The emergence of the web as the omnipresent platform enables leaner and more optimized processes. Systematic definition of different alternatives will result in readily available information that can be used as basis for optimization in the development, deployment, and maintenance in existing software businesses.

Mashware development. An important realization about web applications is that they do not have to live by the same constraints that characterized the evolution of conventional desktop software. The ability to instantly publish software worldwide, and the ability to dynamically combine data, code and other content from numerous sources all over the world will open up entirely new possibilities for software development. While present mashups combine content such as text, images or videos, there is no reason why content combination could not be applied to software development as well. In the long run, web technologies will evolve to a direction that would allow the developers to easily create services as "mashware" – mashup software that leverages source code and software components downloaded dynamically from all over the world. Then, the Internet essentially becomes a uniform distribution media on top of which different component models for applications can be built. Such applications could dramatically improve the productivity of software development, allowing massive reuse of software components across the planet. Again, engineering and security principles for such software systems are needed.

Datacenter hosting and cloud computing. In service development that reuses code and content from other web sites, the Web essentially becomes a distribution media for software components consisting of objects and code located anywhere in the world. Components can be modified on the fly without static compilation, linking, or "stopping the world". In general, we intend to create an environment in which the mechanisms needed for services as mashups are an integrated, built-in feature, rather than an afterthought or a combination of hybrid technologies as in today's systems. Since the Internet computing is virtually independent of the locations and important prerequisites are availability of energy, trained workforce, and stable society, Finland is an attractive country for establishing datacenters and cloud computing premises. However, special knowledge on the establishment of such

¹¹ <http://openwebfoundation.org/>

¹² <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>

facilities as well as on processes that are needed for keeping such systems constantly active are needed.

5.4 Theme 3: Open systems and software

5.4.1 Description

The openness comes with the new rules and attitudes. Therefore, it is of utmost importance for anyone aiming to utilise the potential of open software and systems, to understand the fundamental constraints and mechanisms behind the open development. One central issue is the management of these open development communities. One can also gain from the practices and attitudes of the open development by establishing and nourishing open communities inside a company.

5.4.2 Focus areas

Business perspective. From the business perspective, clear challenges arise, as the development of economically feasible solutions based on free software are fundamentally different from traditional closed source. For example, it is challenging to develop solutions based on free components that can be clearly differentiated from something that anyone can put together.

IPR management. Although the open source software is free in some sense of the word, the rights to the code belong to someone and the code typically comes with a licensing policy that controls its usage. In the light of commercial utilisation of the open source code, the licenses provide a major obstacle to any company. There is a myriad of licenses and their versions, their conditions are complex and many of them have not been tested in the courts of law.

Mixing proprietary and open systems. Issues related to how much proprietary code should be opened for anyone to develop further, integrate with or build new application on top of and how to understand what are assets that constitute the competitive edge by being kept closed or potentially patented. This also includes development and personnel issues. For instance, a company may copy best practices of open source to its internal use (inner source) or a group of companies can form a community (closed community). Moreover, a company may hire developers to participate in open source communities in order to foster its own interest, to the extent that it is not always clear whether a developer works for a company or a community.

Initiating new open source communities. Since the establishment of a successful open source community is hard [¹³], a recipe for creating a lively community of developers and developing companies an interesting research topic.

¹³ Järvensivu, J. and Mikkonen, T. Forging a community - Not: Experiences of establishing and open source project. Open Source Development, Communities and Quality. 15-27, IFIP International Federation for Information Processing, Springer, 2008.

5.4.3 Goals

“Openness becomes a natural and effective part in engineering of software intensive systems and services.”

The goal is to reach the next level of open innovation and business that effectively utilise the open source code and open interfaces. The easiest way to utilise open source components is the usage of open tools in development. Moving from there to use of open source components and opening own code bring in the above-mentioned challenges. The challenges are even more complicated when opening of interfaces is considered, as there are alternative levels at which the opening can be done and considering their pros and cons is not trivial.

The incorporation of an open component into the solutions provided by companies already brings in the issues with licenses and properties of the open components. New aspects include issues such as threats to security they may impose, life cycle credibility, suitability within safety critical domains, means to build a consistent user experience, and so on.

5.4.4 Results

Business models. The business models for the effective business ecosystem based on openness. Three commonly identified business models for open source include support/service provider, multi-licensing, and packaging and distribution. However, there is an emerging need for new, innovative business models that combine the above and aspects such as user and developer community.

Open source utilization framework in Finnish scale. A framework will be defined to address the challenges identified in order to promote effective utilisation of open source in software intensive products, services and systems. The practices have been tested in real environment, experiences systematically collected, exchanged between companies and refined. These provide a basis for gaining and maintaining a justified confidence in open source and systems and for protecting core assets, architecting for suitable IPR management, means for product and service differentiation with open and shared code base. The results will be disseminated in a fashion that is applicable also in small and medium-sized enterprises.

Supporting methods and tools. The practices are supported with methods, tools and knowledge bases, for example, for understanding and reducing the treats from licensing policies of open components, for architecting and designing to cope with overall qualities, such as security, safety or usability.

Documented concrete cases, case products, or services. These will act as a reference for others to copy best practices, as well as provide a concrete testbed for proof-of-concept case studies.

5.5 Theme 4: Security development lifecycle

5.5.1 Description

The security development lifecycle theme deals with offering seamless solutions for security and related issues.

Security related issues have to be considered and managed during the whole life-cycle of the system and software development. Achieving this does not only make us safer and more secure but improves overall system quality and development efficiency. Security and related issues should be analyzed and built in to the system, product or service starting from the very early stages of its development in order to be able to develop proactive architectural, technology selection and system-level solutions.

5.5.2 Focus areas

In the area of security development lifecycle, there are four key areas of focus. These are briefly outlined below.

Development and integration of relevant Security Development Lifecycle (SDL) approaches into other current SW development models. Many current SW development models lack the built-in approaches of taking security issues into account during the course of the entire development lifecycle. SDL approaches, together with tools and methods answer to the . Suitable methods to be addressed are, e.g., agile, lean, embedded. Many of recent security initiatives for software development, such as Microsoft SDL and BSIMM, have been relatively open can be leveraged to support the Finnish industry and to initiate new business.

Management of increasing complexity and vulnerabilities. One of the major threats to security is nowadays the challenges of managing the architectures and implementations. Security engineering work cannot be carried well if there is not enough control of the implementation basis.

Security metrics, security requirements and collection of evidence of the security performance. You cannot manage well a topic if you cannot get evidence of that. Security metrics can be applied to the measurement of security engineering practices, for on-line monitoring activities and for comparison and iteration during the development time. Metrics help potentially to optimize the effort spent on security. Security requirements engineering is an important closely related topics to this. Moreover, measuring security at run-time requires specific architectural mechanisms, i.e. measuring and adaptation techniques, which is possible after managing security measurement at design time.

Tools and methods that support security engineering, including automation. Practically oriented security assurance (analysis, testing, monitoring) tools are needed. The current available tools often support either too abstract or too detailed issues. New bridges should be built between different specialists with the help of usable, scalable, practical and “engineer-friendly” tools. New automated tools and environments are needed to security

and robustness testing, penetration testing and regression testing. Interoperability of tools is also important. In addition, automation of security auditing was mentioned as a topic.

5.5.3 Goals

From ad-hoc security practices to systematic security engineering culture. Security engineering culture involves the analyses, testing, auditing, certification and monitoring of security related activities in the lifecycle of software development. Seamless integration of security into the software development life-cycle forms the basis for a systematic security engineering culture.

From separation to integration. The work undertaken in security development lifecycle aims at bridging the gap between security analysts, security engineers, software engineers and business developers. This requires the development of methods, tools, practices to seamlessly integrate security into the whole life-cycle of software and service development.

From abstract concepts to concrete tools. Security issues have traditionally been handled via high level analysis and abstractions. The lack of concrete tools has prevented them to be effectively employed in practice. The goal is to develop tools and methods for enabling true security engineering. The shift, therefore, is from reactive security engineering to proactive security engineering.

5.5.4 Results

The concrete results of the theme are new globally relevant validated methods, tools and practices in the area of security development life-cycle engineering, capable of increasing the security level of products, services and systems. The results will enable the creation of a new systematic and proactive security engineering culture.

5.6 Theme 5: Green software development

5.6.1 Description

ICT industry can have a major positive environmental impact and it can help other fields of industry to achieve sustainable development. Green SW development is about seeking new ways to reduce the environmental impact of ICT product development throughout the full product lifecycle, innovating new solutions to reduce the consumption of natural resources in ICT products and enabling other industries to minimize their environmental impact.

5.6.2 Focus areas

Business aspects shall consider how to gain savings and competitive advantage with green SW development. Since the price of energy is increasing, power efficient green SW has direct economical impacts. Green SW also offers possibilities for new businesses which significant potential for the Finnish ICT industries. New business can be created by helping other fields of industry to save energy with help of ICT, e.g. by using SW for optimizing manufacturing processes or for traffic control. This requires co-operation between several different fields of science.

Measurements are about measuring the environmental impacts of an ICT product. Measurements contain also ways to visualize the results for the stakeholders. Examples are making SW carbon footprint visible to end-users and SW power usage rating calculated at build time.

Energy efficiency means methods through which ICT products use less natural resources. We aim to focus on system level, total energy consumption, rather than on individual, device level. For example from user point of view the lifetime of a mobile phone battery is important (less need for battery charging) while from network point of view the network nodes could give most power savings. Power saving also includes things like efficient use of processor capacity via parallelization, adaptive systems that can learn and create power use models based on the environment and usage patterns and pushing data processing to the network instead of user device. There is also a need for tools that could be used for finding energy inefficient areas in code. Parallel architectures with large number of relatively small processing elements facilitate energy-efficient solutions by allowing finer-grained ability to perform dynamic voltage scaling and power down. Power down operation is required for minimizing static power consumption. Realizing energy efficiency in heterogeneous networked systems means also a new kinds of architectures with reasoning, prediction, planning and dynamic action mechanisms.

Information and knowledge means spreading information about green SW development and methods for achieving it. One solution for communicating the aspects and goals of green SW development could be a "Green manifesto" similar to Agile manifesto. "Green manifesto" would contain a holistic view of the green software development lifecycle. Besides power saving issues, it could contain also things like recycling, production, office locations and travel between development sites. Information and knowledge aspect includes also green development guidelines that would give concrete advices about green SW development, i.e. guidelines, tools and SW libraries that designers could use in development.

Ways of working, methods and tools are needed for successful green SW development. This includes e.g. ways of working that do not require on-site visits by maintenance personnel, compiler support for optimization for energy saving at compilation time and tools for parallelization of legacy SW with dynamic analysis SW. The last one is important, because multi-core SW is one solution for making greener SW, but legacy code, debugging tools and optimization tools are main obstacles.

New applications can be developed in the green SW area. There is a market for end-user applications that support environmentally sustainable lifestyle. An example of these is SW to follow electricity usage in real-time.

5.6.3 Goals

Finnish ICT sector becomes the world leader in green SW. This creates new businesses and is a competitive advantage, generating an added value that will secure the growth of the ICT industry.

5.6.4 Results

Finnish ICT companies contribute, through their ways of working, products and services to radically reduce the consumption of natural resources and waste of human activities; new businesses are identified and developed, generating new growth sources for Finnish companies.

6 Summary

SW-SRA is formed to address the global trends and requirements from the business to combine the individual efforts in companies, universities and research institutes to form a new ecosystem to drive the competitiveness of Finnish software intensive businesses. Its goal is to form a new Finnish software ecosystem and to foster the creation of new start-ups which support new value networks.

SW-SRA formed a roadmap to reach for the ambitious vision in 2015:"

By 2015 the Finnish software industry will substantially increase the value of its software assets due to it's world-class capability and know-how to efficiently and competitively develop, deliver and use software competencies with a focus on defining, building and exploiting software assets and new ecosystems that have the largest sustainable value add for the global business.

To meet these needs, we identified several global trends, challenges and opportunities that offer the rationale and background for the roadmap. The strategic underlying theme for SW-SRA is to increase the profitability and productivity in Finnish software industry by means of mastering the operational efficiency in the whole value chain. We identified six thematic areas, which offer the alternate and complementary avenues to reach the vision. These areas were user experience, web software, open software and systems, security development life-cycle and green software development.

Projects that will execute the roadmap will be launched in 2010. There will be broad participation from the leading Finnish software companies and research to meet the ambitious targets proposed.