softwarefactory
learn.share.grow.
MAGAZINE

THE VERY FIRST IN 2010 / March
ISSN-L 1798-8845 / ISSN 1798-8845

INTRODUCING
Software Factory
page 3

GREEN
SOFTWARE
page 9

IN THE SOFTWARE FACTORY PIPELINE
pages 13 - 17

UNIVERSITY OF HELSINKI
DEPARTMENT OF COMPUTER SCIENCE

EDITORIAL

In this first issue of Software Factory Magazine, we are happy to present a diverse range of perspectives on the Software Factory concept. It is easy to think that Software Factory is only a single thing: an educational facility, a research experiment, or a platform for seeding entrepreneurship. But it is all of these things, and from the interplay between the angles emerges a constant stream of activity that was not anticipated from the start , only envisioned.

This issue has two goals. First, it provides some answers to the question "What is Software Factory?". The response ranges from strategic outlines to specific accounts of what it means to be part of the project. Second, this issue encourages the reader to set aside any preconceptions about the first question and instead ask "What could Software Factory be?". Software Factory is about learning, sharing, and growing, its form is not set in stone. What will be learned, shared, and grown, is up to the participants.

This is precisely the mode of operation that led to the Web revolution, as highlighted in Esko Kilpi's article. The strategy of Software Factory is open-ended, there is no single end target, but rather a multitude of options, some of which can be harvested immediately, and others which require nurturing to grow. Software Factory Magazine invites the reader to participate!

Fabian Fagerholm
Editor-in-chief

"Whatever we deside to do, we do it exceptionally WELL!"

Prof. Pekka Abrahamsson from Software Factory.

UNIQUE INFRASTRUCTURE INVESTMENT: INTRODUCING THE SOFTWARE FACTORY

CONCEPT

Software Factory is Department of Computer Science's (University of Helsinki) strategic investment to a new infrastructure supporting software engineering research, education and entrepreneurship.

KUVA: FABIAN FAGERHOLM

Having software for breakfast

UNLIKE other mature disciplines, the field of software engineering continues to lack a research and development infrastructure that supports systematic testing of novel software engineering techniques and tools, the seamless integration of current to-date practices and methods to the educational curriculum as well as the means to support ways to grow entrepreneurs from those students with ideas and mindset supportive of the needs of software business. Software Factory is designed to serve all these needs. It is a unique infrastructure platform where innovative software is being developed. It is in its essence an experimental software R&D lab.

If only everything in life was as reliable as software

Software Factory's education strives to integrate its operations in the teaching of several universities offering a wide range of real-life data and material for university teachers' use. The global development space makes the learning experience for students something unreachable in typical university settings. In addition to this, the real-life development environment and requirements give students an opportunity for significant learning gains as opposed to traditional software development projects.

Software Factory's research effectively trains PhD students, performs basic and applied research in its operating context and performs tests for evaluating different research methods. Software Factory provides a context for PhD students as well as Master's students to pursue their thinking further and challenge the common wisdom. Software Factory is designed for allowing a multitude of different ways to collect data unseen before. Cross-disciplinary research is part of the field of software engineering. Therefore the factory has an open call for research proposals to investigate team dynamics, programming psychology and beyond to fully make use of the opportunity at hand.

Finland, Europe and the world needs software entrepreneurs to make business out of software applications developed. Each of the Software Factory's development initiative strives to develop a business-prototype for an alpha-test phase when released. The business development cycle lasts for a single Software Factory operational cycle, which is 7 weeks. After the business prototype cycle is finished, the team goes into a process supported by our collaborators in which the aim is to launch a high-expectation entrepreneurship. This includes the procedures for setting up a company, seeking for a funding as well as other means for business development support.

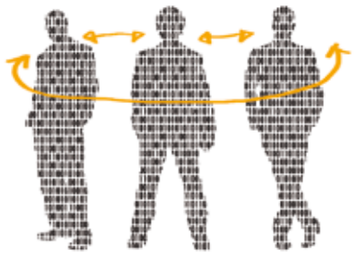
Stop. This software is not ready yet.

The Software Factory operating in University of Helsinki's Kumpula Campus is designed to be a reference laboratory and aims include building up similar infrastructures in the global space. If successful, we are launching a fully operating satellite-hub in Universidad Politécnica de Madrid in Spain in the near future. We continue our future build-up endeavor in India, Australia and Canada. We keep looking for other opportunities around the globe in order to set up a fully operational global software factory by the end of 2010.

We believe that our unique approach to this software engineering infrastructure will attract companies and other researchers to share data and cumulatively build an understanding of the complexities of software and services development.

Pekka Abrahamsson *is a full professor a the Department of Computer Science, University of Helsinki. Dr. Abrahamsson is an active member of the scientific software community and he is currently the academic coordinator of the Cloud Software research programme. He is a member of ISERN, IEEE and ACM. Dr. Abrahamsson can be reached at pekka.abrahamsson@cs.helsinki.fi*

INSIDE VIEW: WHAT BUSINESS CAN LEARN FROM THE WEB



EUGENE Garfield founded the Institute for Scientific Information in 1960. His pioneering work was in citation indexing. Garfield's studies demonstrated that the number of citable items, i.e. the number of papers, together with the frequency of their citation, is a good measure of scientific success. Nobel laureates write more papers than other scientists and these papers are more linked to than other papers. The system effectively measures quantity and quality at the same time.

Links on the Web are also citations, or votes, as the founders of Google realized. The whole Web is a densely interconnected network of references. The observation of Larry Page and Sergey Brin that links are citations seems commonplace today, but it was a breakthrough at the time Google started on September 7, 1998.

What Google has proved to managers is that people's individual actions, if those actions are performed in a transparent way, and if those actions can be linked, are capable of managing unmanageable tasks.

Collaboration and collective work are best expressed through transparency and emergent, responsive linking. The mainstream business approach to value creation is still a predictive process designed and controlled by the expert/manager. This is based on the presupposition that (1) we know beforehand all the needed linkages, and (2) what is the right sequential order in linking and acting. Neither of these beliefs is correct any more. The variables of creative work have increased beyond systemic models of process design.

It is time to learn from the Web.

By relying on the uncoordinated actions of millions of people instead of experts/managers to classify the content on the net, Google democratized scientific citation indexing. To be able to manage the increasingly complex organizations of today, the same kind of democratization needs to take place in the corporate world.

Companies are transforming from industrial mass production to creating value in wide area networks of mass communication. The transparency of tasks is the corporate equivalent of publishing academic articles. Responsive linking, rather than predictive linking, acts as a measure of relevance and is the guarantee of quality. This has served the academic community well. It made Sergey Brin and Larry Page billionaires. Now is the time to do the same in the corporate world. Complex, creative, knowledge based work requires new approaches.

The Google lesson for management is, that the more work is based on responsive processes of relating and the more organizing is an ongoing process in time, the more value we create!

Esko Kilpi, *executive adviser and author, takes part in academic research and lectures on the topics of organizational learning, knowledge-based view of the firm and interaction technologies. He has been a member of the advisory board of the World Bank on Knowledge Management, and a member of the expert think tank on Knowledge Management for the European Union. http://eskokilpi.blogging.fi/*

STRIVING FOR MULTI-DISCIPLINARY RESEARCH

THE field of software engineering has not agreed within itself to which scientific foundation it best belongs. For years, strong argumentation has been put forward for the benefit of engineering discipline where software development has been seen as a field where the engineering takes place according to a certain set of pre-defined rules, steps and procedures. On the other hand, the role of people in developing software has been raised as one of the most important elements to consider since the early days of software engineering. Software Factory attempts to make no claims to one way or another. It is a platform where both seemingly opposing views can be brought together and developed in harmony.

Software Factory research staff belongs to all of the above named disciplines and represent a wide range of different research philosophies and practices. We seek to involve researchers from the fields of social and work psychology, organizational behavior and management sciences. We naturally welcome researchers from computer science, software engineering, as well as information systems sciences to join our quest for high quality basic and applied research.

Openness is the key

Open platforms, interfaces, open source software all represent important streams of development within the software field. Research in the field, on the other hand, has been closed to the extent that the findings made in one place cannot be easily replicated in another setting. Evidence based software engineering [1] has been proposed as a way to build up the maturity within the field. Within its own context Software Factory aims at contributing to the development of evidence in a systematic way. Access to data is another important dimension. Software Factory research attempts to build up an open research platform, which is driven by curiosity and high ambitions. Openness refers to the ability to truly share the scientific data for scrutiny by several competing or complementing research teams. Our ability to collect data is quite unique but shall be treated with the highest ethical guidelines and procedures as it belongs to the class of data called "sensitive biometric data". We should however not fear to use this ability to develop highly ambitious research plans and execute them to perfection. We explicitly call for theoretical triangulation and framing as a way to build up an understanding of the phenomena called innovative software development.

Improving PhD students' research capability

Research in Software Factory benefits industry in several ways but also builds up the capability of a young scientist to perform his/her field studies. Entering the field is a difficult task for a PhD student and it is very likely that several mistakes are made. To better equip the scientist, the Software Factory offers a PhD student an opportunity to develop his/her capabilities to conduct observation and participant observation, perform interventions and conduct interviews in a scientific manner. The PhD student can perform initial tests to their research design and see where it can be improved before conducting field studies on a large scale.

In the near future, when Software Factory grows to its international capacity with several global sites joined together, it opens up new opportunities in terms of research questions and study settings. Researching global software development from tens of different compelling study angles has not been done before.

References

[1] Barbara A. Kitchenham, Tore Dybå, Magne Jørgensen: Evidence-Based Software Engineering. ICSE 2004: 273-281

Pekka Abrahamsson / *Full professor at the Department of Computer Science, University of Helsinki.*

STRATEGIC OUTLOOK:

SOFTWARE FACTORY FOSTERS NEXT-GENERATION SOFTWARE PRODUCTION AND BUSINESS

UNLIKE the early software factories of 1980s based on traditional manufacturing principles, the new Software Factory approach takes a wider perspective of modern software product development. Following the recent global trends of moving from product-focused development towards more service-oriented solutions, it aims to build not only new software products but also developmental knowledge, business and – ultimately – influences people therein to grow in the new thinking.

Drivers

Recent software technology advances – such as cloud computing – and more generally Internet-based networked software services are emerging in many different business fields and public sector areas. On the other hand, modern globalized software product development requires new skills, competences, and even mindsets.

The Software Factory is a development base for building such future capabilities. It offers advances for both academic as well as industrial interest groups in terms of software business, research, and training/education. This aligns with the overall strategy of University of Helsinki for fostering new linkages towards business life in 2010 - 2020.

Strategy

The strategic intent of the Software Factory is to promote:

- Growth: Develop innovative software product ideas and prospective software business ventures.
- Learning: Train skilled and competent software professionals for the software-intensive industry.
- Sharing: Facilitate intensive industrial-academic collaboration and relevant research knowledge creation.

Those traits strive for putting forward operational models and templates of future high-performance, agile/lean software enterprises. Fundamentally, it attempts to rediscover the seminal "knowledge-creating company" elements in modern settings (Takeuchi and Nonaka 1986).

Connections

The establishing focus area of the Software Factory is web-based Internet service software development in particular for cloud computing. Following the current globalization trends, the software

Future Prospects

This line of strategic thinking opens up many potential avenues for leveraging the Software Factory, such as:

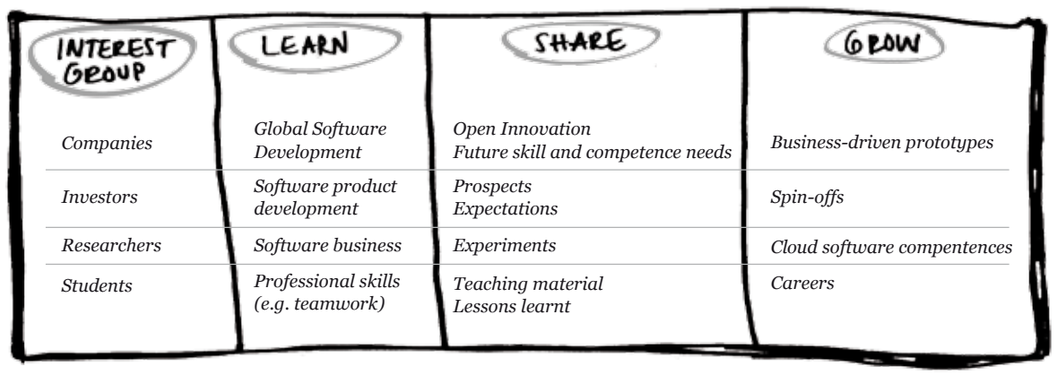
- Innovation Hub (growing)
- Living Lab (learning)
- Research Center (sharing)

Ideally there can be shared interests and win-win-win set-ups here for multiple parties over time. For instance, industrial companies could share their research needs as well as educate their (future) employees. Furthermore, networking with related software research, consulting, and training/education organizations (e.g., EIT ICT Labs) could provide further synergies and additional opportunities.

References:

Takeuchi, H., Nonaka, I., 1986. The new new product development game. Harvard Business Review 64(1), 137-146.

Petri Kettunen*, D.Sc. (Tech.) from Helsinki University of Technology (since January 1, 2010 Aalto University), has had an extensive industrial career mainly in telecom business. Now with Prof. Pekka Abrahamsson at University of Helsinki, department of Computer Science, Dr. Kettunen's current interests include software engineering management (in particular Agile/Lean), software-intensive new product and process development, and organizational development.*



EDUCATIONAL PERSPECTIVES IN SOFTWARE FACTORY

LEARNING is about personal change. Formal higher education in computer science gives an opportunity to invest time and effort into learning – in other words, to make the personal change possible. In order to accumulate enough expertise to really be an expert in any given, a multitude of educational methods are used. There is a place for lectures, lab exercises and open-ended assignments; bits and pieces of information. But there is also a need for authentic learning situations that glue everything learned into a meaningful whole. Many times in formal education, these authentic learning possibilities are too rare. Such learning possibilities within the relatively safe formal context are indeed craved by the learners themselves. It is easy to see how inherently motivating and rewarding genuine learning possibilities are, when everything just "clicks" – real people, real demands, real facilities, and real need to showcase your talent, knowledge, creativity and perseverance.

At the Department of Computer Science, University of Helsinki, we now have a place for such a deep learning experience: Software Factory and a Software Factory project. It is a hands-on, outcome-driven personal growth experience for the students. It is not just a physical place to work but also a challenge and an opportunity. It allows the learners to put every piece of their accumulated expertise into practice; a challenge to exceed oneself. Moreover, it is an attitude: "Show me what you really can do!"

Software Factory has started in January 2010. It has been an instant success, even though the activity is still in the pioneering stage. What are the reasons behind the success?

Software factory project is intense

It is clear that any project in the Software Factory requires devotion of time and effort. Everyone needs to declare their commitment to the project by signing into the four- or five-days-a-week work cycle for seven weeks. The Factory is an office space, and it is equipped with real tools and facilities, precisely what you could expect to find in an up-to-date software business. Such an authentic setting does not undermine the academic side of the project work but is beneficial to the learners' self-confidence, since it provides them with real experiences and exposure to the same facilities as the businesses in the field.

When the activity in itself is the motivation, there is no need for external, artificial assessment of learning. Projects are inherently and intrinsically motivating as every software project in the factory has a potential path to commercialization, spin-offs and startups.

Software factory is about interpersonal interaction

Interaction between people and other social skills are required for every expert in a highly complex field such as software business. Every project is a group effort. On the lines of best examples of cooperative learning (e.g. Johnson & Johnson 1989), software factory emphasizes positive interdependence between the team members while requiring individual accountability. In addition, there is a cultural context as the factory has been multi-cultural from the beginning, and aims to be global in the very near future. Team members work side-by-side in close cooperation, but relying also on dispersed resources.

Software factory empowers the learner

In formal education, often a rigid structure is imposed to the learning process, leaving little control to the learners themselves. Structure equates to teacher control, dialogue to negotiated control, and autonomy to learner control (Dron 2007). There is a place for structure, dialogue and autonomy, so none of them is undesirable in learning. But there is an added benefit for the learners if the locus of control can be shifted dynamically. As Dron puts it, "the ideal would be to allow the learner to choose whether and when to delegate control at any point in a learning transaction". This is precisely the educational structure of the software factory – true learner-driven learning with always-available non-invasive support for technical, managerial and leadership challenges.

Software Factory's global extension takes the learning to a new level

While software development today is by definition globally distributed, also Software Factory has plans to set up satellite sites around the world. This is likely to take the learning even further as the global space will increase the level of authenticity significantly. Challenges discovered in industry include communication and coordination issues as well as cultural themes, which are likely to shape the outcome. Later this year, our sister-site at Madrid will be established and the Born Global thinking will begin to take shape. We believe that this is also increasing students' motivation toward the maximized learning experience. We also believe that having students passing the Software Factory experience are better equipped in facing the volatility and inherent dynamics of the industrial setting once graduated.

References

Dron, J. (2007). Designing the Undesignable: Social Software and Control. Educational Technology & Society, 10 (3), 60-71.

Johnson, D.W. and Johnson, R. 1989. Cooperation and competition. Theory and research. Edina: Interaction Book Company.

Jaakko Kurhila, *PhD, is the Head of Studies at the Department of Computer Science, University of Helsinki. His earlier career work included research on various topics of educational technology, especially collaborative and social software. Nowadays, he eagerly promotes greatness in education.*

SOFTWARE FACTORY RESEARCH HELPS IN FACTORING high-performance software enterprises – EVEN AT LARGE SCALE

MOST software-intensive product development organizations face new challenges with such factors as globalization, service-oriented business models, and web-based technologies – such as cloud computing. Agile software methods and Lean principles are attracting both large companies as well as SMEs for coping with those challenges, but their deployment and moreover large-scale transformation cause often further complications. The Software Factory can help in piloting and also developing such approaches in a realistic, state-of-the-art test environment.

Drives in the large-scale

The current software-related megatrends impose in particular the following competitive environment drivers for many industrial product development organizations (Tekes 2009):

- market share "battles"
- new product time-to-market "windows"
- customer-driven responsiveness
- global-scale cost pressures
- sustainability expectations

Moreover, in many competitive environments those dynamics and other, even disruptive factors cause considerable turbulence. Nevertheless, industrial software development organizations still need to achieve their business targets for instance in terms of profitability.

New software development models such agile software methods address many of those issues yet their effective adoption tend to introduce additional organizational considerations (Kettunen 2009). It is often not clear how the software development functions can really contribute to the bottom-line business targets in particular under changing conditions. There is a need to build and demonstrate the linkages in order to be able to manage software engineering projects strategically.

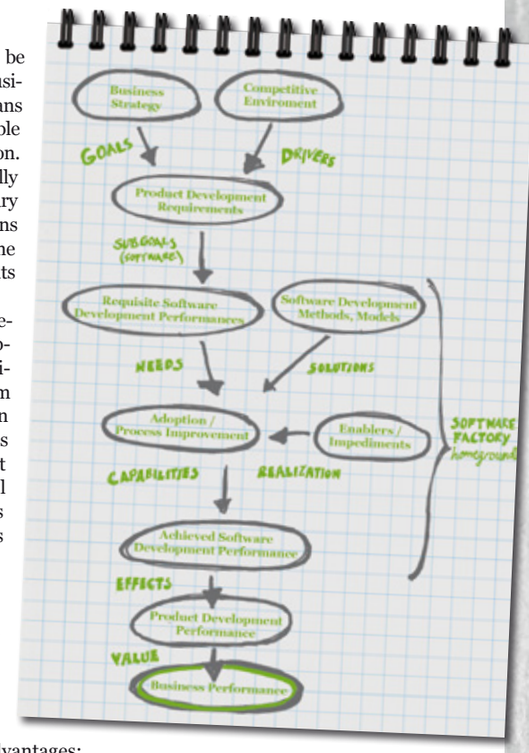
Leanness and agility are not goals

In particular with the Agile/Lean software development models, it is important to distinguish between the desired effects (goals) and the different ways (means) of achieving them:

Goals: strategically targeted level of process performance (e.g., reduced cost)
Means: tactical and operational realization of the targeted goals (e.g., agility, agile software practices)

Naturally the Goals must be aligned with the company business strategy, and the Means should be based on the available resources of the organization. In addition, there are typically many enabling factors necessary for realizing the selected means efficiently (e.g., skills). On the other hand, some impediments may hinder them.

All in all, those different elements can be seen as contributing elements in the organizational performance system of the firm like illustrated in the sidebar figure. Systems thinking is then important in understanding the overall value-creation mechanisms of the enterprise (e.g., flows and wastes).



New Opportunities

The Software Factory brings new opportunities for product development organizations to improve their software functions. It provides in particular the following advantages:

- "fail-safe" yet realistic R&D surroundings
- an advanced instrumentation environment

The former point is valuable for large-scale industrial organizations in which it is often not feasible to distract the ongoing development projects with additional experiments. The latter element serves well smaller organizations, which do not necessarily have the resources for high-end instrumentation facilities.

The Software Factory supports in particular the middle stages of the performance improvement model depicted in the *sidebar figure*. It is possible to develop and analyze the software development operations (even global) for example in respect of the following goals:

- cycle-time acceleration
- waste reduction
- teamwork boosting (also multisite)

Altogether this creates an extensive test and training post for various software product development organizations to improve their current operations as well to experiment with new practices. They can first test and benchmark them in the instrumented Software Factory environment, and then mirror them to the actual development sites.

References:

Kettunen, P., 2009. Agile Software Development in Large-Scale New Product Development Organization: Team-Level Perspective. Dissertation, Helsinki University of Technology, Finland.

Tekes, 2009. The Role, Importance, and Future Challenges of Software Competences in Finland – Drive for Future Software Leverage. Report 262.

Petri Kettunen / *D.Sc. (Tech.), postdoc researcher, Department of Computer Science, University of Helsinki*

DISCOVERING SOURCES OF WASTE IN SOFTWARE DEVELOPMENT

WASTE, in terms of lean software development, is everything that has no value for the customer [1]. Waste appears when a particular action is taken, which does not add value in fulfilling customer expectations. Potential sources waste include waiting, programming unnecessary features, doing extra processes, extra motion, partially done work, and defects, as well as switching between tasks is considered waste regarding software development [1]. Management activities may not be valuable for value adding either. However, they do have an impact on wastes in the organization. Recognizing the time spent on adding value versus the time spent on waiting is essential when developing a value stream map of a particular product or service development endeavor.

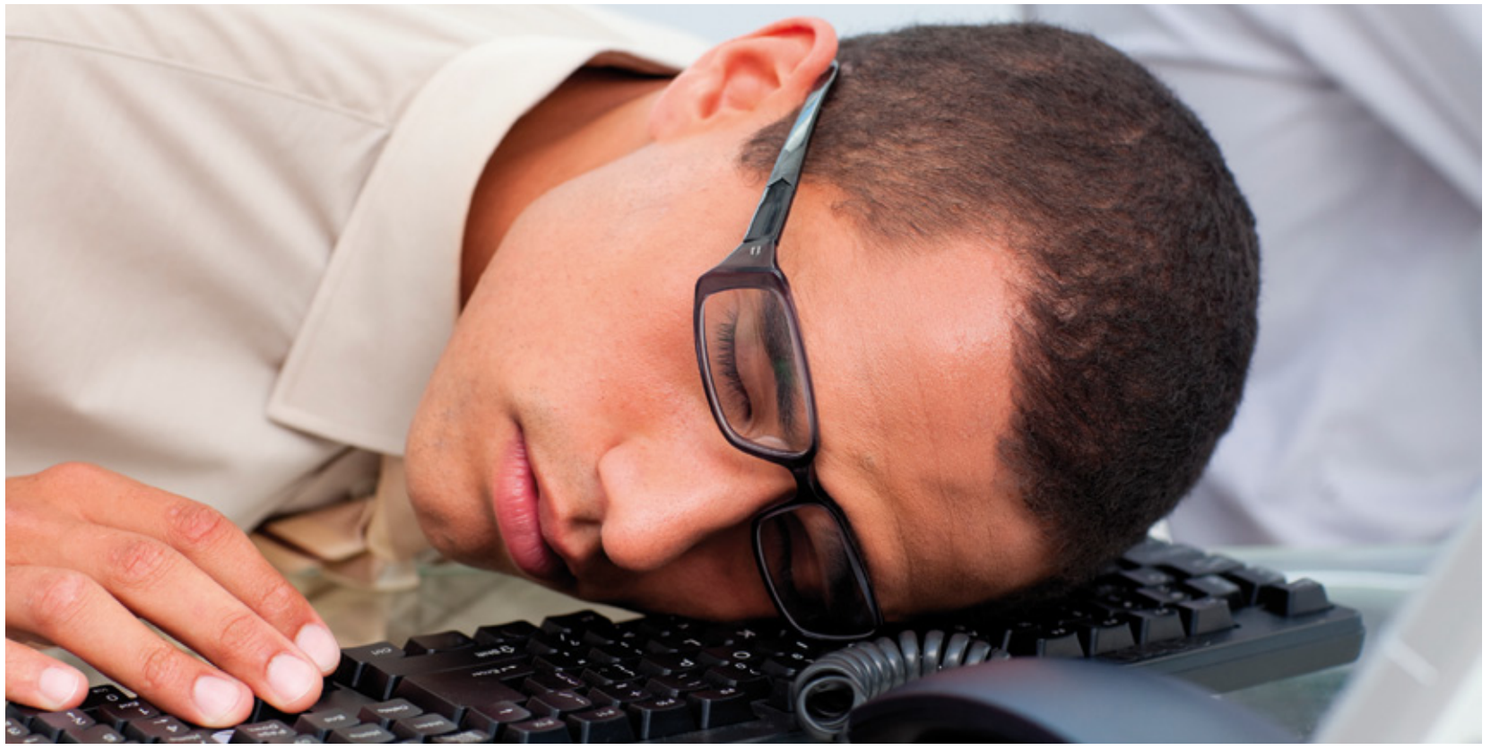
These principles form the underlying baseline for which the present author will study how to find wastes in software engineering projects. It is expected that new sources of waste are to be discovered as well. In addition, waste removal strategies will be developed. We expect these to be effective in increasing the value creation capabilities in software organizations. The study includes empirical evaluation of software engineering projects in both industrial and Software Factory settings.

We have developed a framework and a questionnaire [2] to aid the evaluation of determining the potential sources of waste as well as its relation to software engineering project success. The framework has been designed to include the majority of success relationships drawn from different reference disciplines. Such a multidisciplinary perspective provides a comprehensive approach for practitioners who are searching more cost-efficient ways to implement their software projects.

The questionnaire includes both team-level and individual issues regarding effort, competency, coordination, communication, and performance, among others. It is designed to be applicable to a wide range of different project types including the agile mode of development.

Marko Ikonen
M.Sc., is a researcher at the Department of Computer Science, University of Helsinki. His particular area of interest is the elimination of waste in software engineering projects. In this area, Ikonen applies his knowledge of leadership, management, organizational behavior, and lean software development.

References:
[1] Poppendieck, M. and Poppendieck, T. Lean software development: an agile toolkit. Addison Wesley, Boston, Massachusetts, USA, 2003.
[2] Ikonen, M. Working Toward Success Factors in Software Development Projects. Number 2009 19 in Series of Publications C. Department of Computer Science, University of Helsinki, Finland, 2009.



GOOD SOFTWARE IS ALSO GREEN!

Calculating the Software Factory's carbon footprint

WE live in a changing world. According to some climate experts we should aim for a carbon-free society in the next few decades. In practice, this implies that every field, including software engineering, must cut their carbon emissions.

Carbon emissions are measured as a carbon footprint (CF). It defines how much carbon dioxide emissions a product or service produces. Every entity that has a life cycle also has a CF.

The University of Helsinki Software Factory (SF) is a new educational, research and entrepreneurship concept where a group of students work full time for seven weeks in an environment designed to create flow in software development. The result of the project is a business-prototype service ready for alpha-tests.

The basic way to calculate the CF of the Software Factory project is to concentrate on the working environment and its CF. Since the life cycle of the project takes place entirely in a single facilty, we can count the CF of the room usage and propose that it is the carbon footprint of a Software Factory project.

When considering the room usage, it is likely to include at least the following elements impacting the "size" of the carbon footprint: heating, lights, computers, monitors, video projectors, printers, papers, and binders. All of the listed elements have a life cycle and hence a carbon footprint.

While the basic assumption shown above is correct per se, it gives a limited idea of the carbon footprint of a project executed in the Software Factory settings. It excludes important aspects: participants and the delivered software or service. In a more advanced analysis we need to include the missing components. We consider this briefly here.

Each participant of the Software Factory has a life cycle and hence a personal carbon footprint. The intuitive solution is to include the CF that is part of the SF work hours. This includes at least the following during the working hours: food, drinks, snacks, and other physical used resources. Anything beyond the intuitive solution is a matter of debate. For example, participants' travelling, accommodation, phone usage, clothing usage etc. may or may not be part of the CF. However, it is not clear how much of the resources are directly required for the SF.

Each participant of the Software Factory has a life cycle and hence a personal carbon footprint.

Finally, the result of the SF project is the delivered software product. Also the software has a life cycle so it has a CF. Is the software CF included in the total CF? Perhaps not since it is included in the computer CF. However, if the SF delivers low quality software that requires too much CPU time to use, the total CF of its users will also increase. While the direct CF of the SF is not affected, the indirect CF definitely depends on the delivered software. Good software is green software!

Juha Taina, *PhD, University Lecturer, University of Helsinki, Department of Computer Science. Research interests: Software systems.*

DEFINING USER EXPERIENCE REQUIREMENTS - HOW SUPERIOR IS GOOD ENOUGH

GOOD user experience (UX) is a key word in today's development endeavors. It is a fundamental competitive factor and critical to product or service success. Products should have surprising features which create a "wow" effect on the customers. When users use a system empowered by high user experience, they are likely to become loyal to the service. Such systems can even tolerate having some key features missing.

When designing anything, one should first define the requirements: what do we want to achieve? The focus of our research is this aspect of UX work: how to determine the "right" user experience requirements – how superior UX is good enough?

As with any requirements, UX requirements should be valid and verifiable. Valid UX requirements mean that they depict a product that truly provides a good user experience; the achievement of verifiable requirements can be objectively measured.

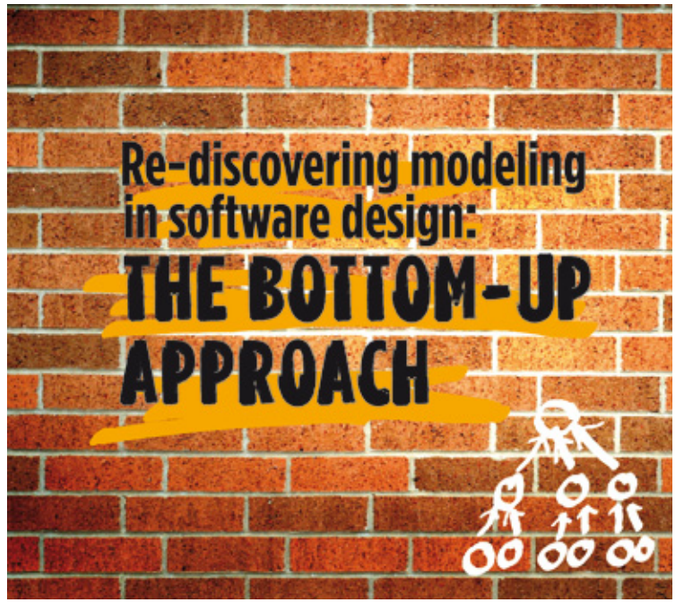
When defining UX requirements, one should define UX measures, measuring instruments and set appropriate target levels. For example, if we regard user effectiveness as an appropriate attribute of UX, then we might select summative usability testing as a measuring instrument and define a target "95% confidence that at least 75% of users achieve the correct outcome". All of those issues are related to definition of "superior UX". Requirements may be challenging to discover: appropriate UX measures, measuring instruments and target levels.

How to determine the "right" user experience requirements?

The hypothesis of our research is that this starting point of UX requirements definition is the business needs, and the aim of our research is to develop understanding and methods for how to derive the UX requirements from the business context of the product to be developed.

In the future, we have plans to extend our research from the business needs perspective also to the workings of the human brain. We seek to explain what happens in the brain when a user discovers something that provides him or her with the feeling of superior user experience.

Timo Jokela, *Ph.D., is a usability researcher and practitioner. His particular interest is deep diving into users' world. Customers say that Timo is capable of understanding the essentials in an amazingly short time.*



MODEL-DRIVEN architecture in its various forms appears into the software engineering research radar every now and then. In principle, the idea is beautiful: we could increase software development productivity by raising the level of abstraction, thus allowing our developers to concentrate on the actual domain instead of fighting against the compiler on the programming language level. However, traditional model-driven approaches are rather heavy-weight, as they require intensive support from various tools before meta and meta-meta -level constructs are defined precisely enough to be used. The end result appeals to architectural astronauts, but is seldom agile enough for real software development.

The main message with bottom-up modeling is to constantly look for sub-domains that are amenable to lightweight formalization.

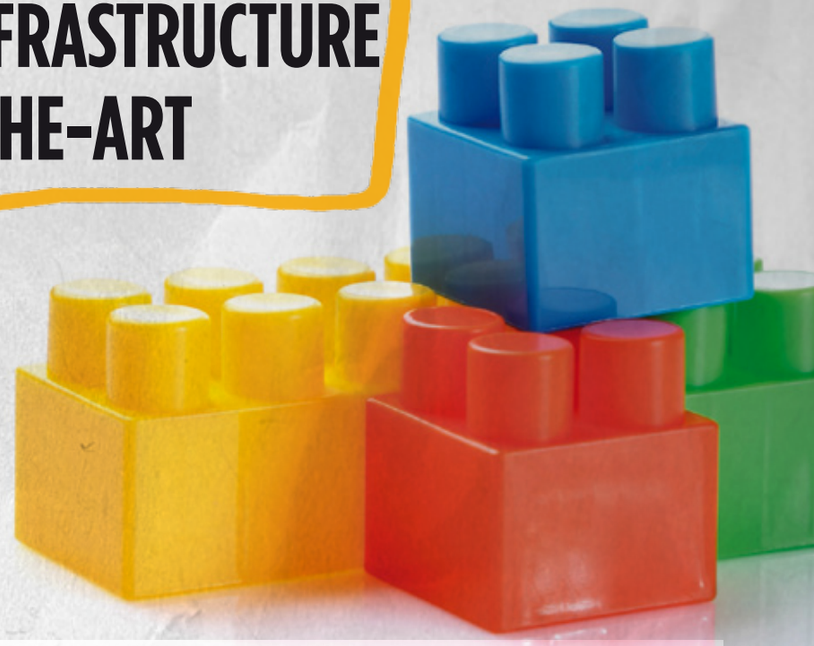
Science often finds two approaches to any given problem. Traditional model-driven architecture is essentially a top-down method, which recursively decomposes the problem to smaller pieces, until any given piece is small enough to be solved with available tools. The alternative to top-down thinking is to use a bottom-up approach. This approach identifies smaller sub problems and develops solutions to these. When this cycle is repeated, gradually the solution for the whole emerges. Given a program domain, a bottom-up approach identifies sub-domains that are amenable to modeling.

Agile bottom-up modeling constrains the identification process to such tools and techniques that can be applied in an agile process model. Its application is thereby limited to a small number of tools, which can be evaluated and applied within a tightly time-boxed iteration. Yet, although this search-and-discover approach theoretically produces non-optimal solutions, it guarantees that progress is not stalled while searching for the optimal solution. This way, a bottom-up approach to modeling dodges the heavy up-front planning phase associated with traditional model-driven architecture. There is no need to build complex modeling languages with associated tool support. Instead, existing languages can be piggy-backed and reused or a domain-specific modeling tool can be used. The obvious downside of the approach is that repetitive application of ad hoc modeling constructs might gradually erode the software's overall architecture.

The main message with bottom-up modeling is to constantly look for sub-domains that are amenable to lightweight formalization. Crafting meta-models and tools for such a small domain is definitely easier than building an all-inclusive supermodel of the whole world.

Pietu Pohjalainen, *M.Sc., is currently pursuing a Ph.D. at the Department of Computer Science, University of Helsinki. His special interests include finding ways of improving software development productivity.*

SOFTWARE FACTORY´S TECHNICAL INFRASTRUCTURE IS STATE-OF-THE-ART



THE BEST working and least demanding solutions for IT infrastructure are usually quite simple. This principle has been kept clearly in mind when building the needed infrastructure for Software Factory. Possibility to clone the configuration to various locations around the world has also been a guiding factor in design and implementation.

Overview

Software Factory currently relies completely on modern standardized IT infrastructure and is thus easily implementable in any location equipped with certain basic key components: storage area network (SAN), disk array system, tape library, blade servers, centralized authentication, high-speed local area network (LAN) and a few off-the-shelf workstations. With these components implementation has been kept simple, working and easy to manage without sacrificing functionality or performance - the key factors on this project.

Camera data is stored in the network video recorder, which makes nightly archives to the file server.

Storage

We currently provide a total of 10 terabytes of very fast disk space from our disk array system for user and camera data. An additional 10 terabytes will be allocated for analysis space as soon as data analysis from e.g. camera data starts. File service is handled by a modern blade server connected to a storage area network for connectivity to the disk array system. Protocol for disk sharing to workstations is SMB with POSIX extensions to allow us to run GNU/Linux in both servers and workstations. File server backups are taken nightly over SAN to increase performance and reduce network load on LAN.

Video and audio

Cameras are powered using power over Ethernet (PoE), allowing the same cabling to be used for data transfer to the network video recorder. Data is compressed in cameras using H.264, MPEG-4 or Motion JPEG and thus doesn't need much bandwith. Only uplinks from switches and links between servers require faster (1Gbit/s) speeds. Camera data is stored in the network video recorder, which makes nightly archives to the file server.

Connectivity to other sites

Connectivity to future remote installations of Software Factory is planned to be handled with VPN tunnels, thus removing the need for separate per-port tunneling. This also allows ease of administration and more flexible communication between locations and makes adding locations easy. Each location mainly utilizes local networks; only traffic routed to other locations passes through VPN tunnels.

Conclusions

Building and maintaining IT infrastructure for a complex project like Software Factory requires reliable, simple components. These components can be combined to provide advanced functionality, but at the same time keep the system easily manageable. Due to simplicity, the setup can be replicated elsewhere to the extent needed.

Pekka Tonteri *works as the IT manager at Helsinki Institute for Information Technology HIIT. His core competence is in IT infrastructure design and building. He is also involved in research carried out in ICT SHOK Future Internet programme.*

PSYCHOMETRIC measurements in software development

SOFTWARE development is an inherently social activity. The psychology of programming is the field of research that deals with the psychological aspects of developing software [4]. Despite its importance, quite little emphasis has been placed on gaining a deeper understanding of the psychological aspects of software development. We know that professional software developers, designers, testers, and architects are often well versed in software engineering techniques and methods. They are able to apply these in their individual work tasks, and they are able to perform consistently on their own, each with their individual style and strengths. Software development requires the advanced cognitive abilities that these professionals possess [3].

However, reality mandates that software development be performed in teams, on a schedule, and most often with evolving and or vague requirements. Teams that were previously co-located may be spread globally. Members may work in several projects at once. The amount of communication required to work out the boundaries of a single task is growing, perhaps beyond human capacity. Organizational structure impacts software development. This effect has been observed and reported in software engineering literature for decades, and its impact has continued to increase [1,2].

What is the basis for coping with the demands of a modern-day software organization? Are differences so great among individuals that only some are suitable for the work despite their technical expertise? If so, what are the characteristics that set these individuals apart? On the other hand, if only a handful of individuals are capable of performing in such an organization, can that organization be considered a well-working, professionally managed environment with trustworthy leadership - a good place to work? Will employees stay in such an organization, or will they seek an environment that allows sustainability and personal growth? Furthermore, what are the characteristics of a truly excellent work environment where expertise can be put to the best possible use? Are they purely ones that enhance cognitive ability or are there other psychological and social dimensions as well?

What is the basis for coping with the demands of a modern-day software organization?

Finding answers to such questions requires treatment of software development as a social activity, and treatment of the individual as a human with a psyche. Psychometric measurement can approach individual traits, behavior, and experience scientifically, but application of measurement instruments can be too disruptive in a work environment. This is one of the reasons the Software Factory's data collection instrumentation has been carefully designed to minimize unnecessary disruptions. Software Factory provides a realistic environment where all the elements of teamwork are present, along with external pressures such as deadlines, changing requirements and expectations from real customers. At the same time, it allows the small compromise of periodically disrupted work required for individuals to make self-reports.

Currently ongoing psychometric research in Software Factory is collaborating with researchers at the Faculty of Behavioural Sciences, University of Helsinki. We will be offering compelling scientific evidence and deeper understanding about the teamwork dynamics, leadership, performance and the fun-factor of software development.

References

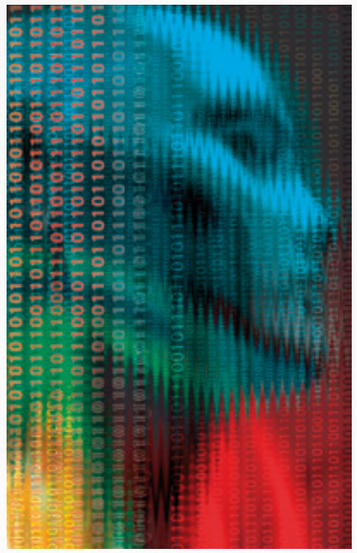
[1] Frederick P. Brooks, Jr., The Mythical Man-Month: After 20 Years. IEEE Software, vol. 12, no. 5, pp. 57-60, Sept. 1995.

[2] Nagappan, N., Murphy, B., Basili, V., The influence of organizational structure on software quality: an empirical case study. ICSE '08: Proceedings of the 30th international conference on Software engineering, pp. 521-530, 2008.

[3] Robillard, Pierre N., The role of knowledge in software development, Communications of the ACM, vol. 42, no. 1, pp. 87-92, 1999.

[4] Weinberg, Gerald M. The Psychology of Computer Programming (Silver ed.). New York: Dorset House, ISBN 0-932633-42-0, 1998.

Fabian Fagerholm, *M.Sc., has a background in information technology management consulting, in mass media, and communications, and in software and service architecture and development. He is currently pursuing a Ph.D. at the Department of Computer Science, University of Helsinki, is the Software Factory lead, and the editor-in-chief of Software Factory Magazine.*



IN THE FACTORY PIPELINE: MOBILIZING CHINA

LEAD-USER driven innovation and open innovation paradigms together with Web 2.0 technologies seek to involve consumers and common people to innovative product development projects. It is believed that people potentially possess bright ideas about future products and services. Literature and companies have presented several success stories about involving people in their core product development. It is unclear, however, how well common people's ideas reflect actual technological development and their potential use in the future or even near future.

A few years ago we undertook a massive collaborative research effort and collected 40000 ideas from 2000 common people about future mobile services that they would like to use [1]. We inspired people not to think about the technology but to focus on their concrete needs. The age variance for the target group varied between ten and eighty. The results show that consumers' technological foresight is within 1-2 years to the future. We demonstrated that people are excellent in the capability of coming up with novel service ideas that are implementable with today's technology. We estimated that from the massive number of 40000 about 80 - 90% are such ideas that would bear business benefit already today if developed.

The facts are clear. Considering the future of mobile services, it is very clear and evident that China has raised its mobile profile unlike many other market areas. Motivated by the excellent results gathered in the trial in Finland, the Software Factory undertakes an ambitious initiative to help in mobilizing China in the coming fall. We plan to develop a service with a compelling revenue model for the users. The service is designed to meet the special demand stemming from Chinese markets and will be made available only in the focal language. Stay tuned for further developments!

References

[1] Alahuhta, P., Abrahamsson, P., & Nummiaho, A. (2008) On Exploring Consumers' Technology Foresight Capabilities - An analysis of 4000 mobile service ideas, ICE-B 2008 - International Conference on e-Business, 2008, Setubal, Portugal, INSTICC

Pekka Abrahamsson and Petteri Alahuhta

Petteri Alahuhta *is currently working for VTT Technical Research Centre of Finland. He has more than 10 years of experience in successfully leading R&D activites in mobile and ICT-business. His research interests focus on changes in mobile technologies and -business.*

Software Factory in Action: EXPERIENCES FROM THE TRENCHES

WORKING with a great team in a very realistic project, learning a lot, and having the freedom to decide what to do and how. These were the main things that my team members mentioned when I asked them what the coolest things have been in the very first Software Factory project. It is very easy to agree with them.

In Software Factory the team can decide how things are done. Basically the only external limitations we had in the beginning were the facilities, Kanban as a process model and Ruby on Rails as the main development technology. And actually all of these were very positive limitations. Nevertheless, we have been able to take all the necessary actions in order to satisfy customer needs. We have used different agile methods like pair programming, test- and behaviour-driven development, retrospectives, and continuous integration, to name a few. All of these have emerged from the team's own willingness and decision to use them.

I believe that another reason for the team to be highly motivated is that we have worked in an environment that is very close to real working life. A Software Factory project just isn't like a typical university project, it is so much more. It is truly agile, it has the finest possible facilities and it makes you forget that you are actually having a university course there.

I also asked about the negative things but it seemed like the team had problems in finding them. Someone mentioned that it requires quite much work and another said that in the beginning the learning curve is quite steep. But another aspect is that in the end you have really learned a lot and the effort you have put in hasn't been a waste of time.

Stay tuned for upcoming projects. I can sincerely recommend Software Factory for all computer science students.

Henri Karhatsu, *M.Sc. (economics) and B.Sc. (computer science), private IT consultant and currently a research assistant in the Department of Computer Science. His particular interests include agile and lean software development.*

IN THE FACTORY PIPELINE:

MASHING UP THE NEWS YOUR WAY!

Introduction

The electronic media, internet in particular, has changed the way we read the news by providing access to hundreds of news websites and information sources. There are already many news websites, for instance Google News and Yahoo! News, which gather news from different sources, and summarize them according to categories. While the user has unlimited access to news, it has however become difficult to find what is relevant or interesting to the user. There are some news recommendation systems which generates news items based on user's interest. These recommendation systems, as far as we know, are restricted within one language. Related news items in different language may provide access to other important aspects of news, hence giving more useful information to the user. Software Factory will undertake an ambitious effort where we aim at creating a personalized cross-lingual news retrieval system based on user's interest by learning from both explicit and implicit feedback. Our highly efficient retrieval system will be flexible by allowing user to specify the news sources of interest, and also to add other information sources, for instance wikipedia, twitter, blogs.

Possible recommendation strategies

News recommendation systems are typically based on either content-based mechanism or collaborative filtering mechanism. In content-based recommendation system[3,4], news items are generated based on user's profile and its history, for instance click behavior or content of already accessed news items. The collaborative filtering recommendations systems[1] are based on the opinion of peer users and do not take into account the actual content. Recommendations based on collaborative filtering reflect the trend of news reading based on ratings by peer users, and would not be able to recommend news items which are not rated so far but might be interesting to the user. The content based recommendation systems take user's preference or interest into account and are able to recommend unrated news items too. A recommendation system using a combination of both mechanisms would be preferable. Earlier version of Google News [1] was based on collaborative filtering, and recently a recommendation system based on click behavior has been presented in [2], which is a hybrid of content-based and collaborative filtering mechanisms. PIN [4], a content-based news recommendation system, allows the user to explicitly define a list of keywords to reflect his/her interest, and the system then updates the profile of user's interest based on explicit feedback given by user. Unlike PIN [4], the system in [2] does not require explicit user feedback instead it learns the user interest automatically using implicit feedback.

Software Factory approach

Our news retrieval system may be categorized as a hybrid of the approaches in [2,4] since it would allow the user to define a list of keywords, but it would not only depend on explicit feedbacks from user to infer what is interesting or relevant but would also use the implicit feedback, for instance click-behavior, and topics detected based on accessed items, to automatically infer interesting news. Unlike [2], we do not want to use opinion from peer users as it might generate news that may be less interesting to user, we rather use keywords and explicit feedback given by user to find what user really want to read. Current recommendation systems put more emphasis on learning user's preference and interest to generate all possible related news items.

Our idea is to put more emphasis on finding news items that reflect different aspects for a given a topic of interest provided by the user, we do however infer profile of user's interest based on user's history of news reading. A related work to our line of thinking is presented in [5], which tries to augment news information given a story of user's interest by finding similar items that contains something new about given story and not just any similar news item which might not provide any new information. It is however limited to find news items within one language, on the other hand it is reasonable to expect that news items in different language may also contain more useful information for a given topic of interest. We focus on developing a cross-language news retrieval system.

Personalized user experience

Another important feature of our news retrieval system is that it does not believe that all what a user want to know can be generated only from news websites, there may be instances where a topic of user's interest is not a news in general. For instance, if I want to keep track of research activities and developments in my area of interest, the usual news recommender systems would not help. In an abstract level, our news retrieval system does not recommend the user based on what is published in news websites, instead it would recommend the user all possible information that can be generated from internet based on user's interest. For this reason, we also allow user to specify the sources of news and information that the user may be interested to follow; for instance, apart from news website the user would also want to get information from different blogs and twitter. In another example, a researcher would want to include a specific set of scientific journals to keep track of research advancements related to his/her research interest,or a technical expert in an antivirus company would want to follow news about competitors, reviews and feedback about particular security product by following only specific websites dedicated to such technical information. This kind of flexibility makes the news retrieval system more focussed on what user really want to know and provides more personalized experience to the user.

Summary

The goal of our cross-lingual news retrieval system is to generate news items and information based on user-specified keywords and implicit feedback from user such as click behavior. It will allow users to search information about topic of interest beyond only the news websites in a more focused and personalized way. The system not only provides different aspects of given topic of user's interest, it also infers user's preference based on user's implicit feedback to recommend what else could be interesting to the user. Furthermore, it would allow user to access the news and information from multilingual source websites, which has been lacking in earlier news recommendation system.

References:

[1] Das A. S. , Datar M. ,Garg A. ,Rajaram S., Google news personalization: scalable online collaborative filtering, Proceedings of the 16th international conference on World Wide Web, May 08-12, 2007

[2] Liu, J., Dolan, P., and Pedersen, E. R., Personalized news recommendation based on click behavior. In Proceeding of the 14th international Conference on intelligent User interface.IUI 2010.

[3] Billsus, D. and Pazzani, M. J., A hybrid user model for news story classification. In Proceedings of the Seventh international Conference on User Modeling, Pages,99-108, 1999.

[4] Tan A., Teo C., and Keng, H. M., Learning user profiles for personalized information dissemination. IEEE Int Joint Conf on Neural Networks. 1998.

[5] Iacobelli, F., Birnbaum, L., and Hammond, K. J., Tell me more, not just "more of the same". In Proceeding of the 14th international Conference on intelligent User interfaces. Pages 81-90.IUI 2010.

Abhishek Tripathi *is pursuing a PhD at Department of Computer Science, University of Helsinki. His research interests include machine learning methods for multi-view learning, data fusion, feature extraction and matching algorithms. He holds a master's degree in Mathematics from Indian Institute of Technology Kanpur, India.*

In the Factory pipeline: SOCIAL MEDIA IN AN OPEN PLATFORM – SCIENCEBOOK

SOCIAL media is currently the hottest topic in Internet evolution. In this evolution we have witnessed a change from technology-driven showcases to today's rich social applications that are available to every member of society. These networking sites drain their primary energy from people's need to be in contact with their friends, relatives, and colleagues.

A number of different architectures of social interaction have been invented. Twitter emphasizes the flow of interesting pieces of information about current events ("tweets") while Facebook provides a chat and presence application. On top of the main offering, third parties can build their own applications on top of the corresponding platform. A common feature with many of these social applications is that they are primarily targeted for entertainment. A trade-off with the chosen target is minimal support for professional communications.

For this reason, we are building a networking site for science communications, called Sciencebook. Shamelessly we're re-using the good sides of current social media's practices, but we are also aiming to fix their shortcomings. Our social site is a decentralized network of active nodes, with enhanced focus to security and privacy between the nodes. It will be an interesting research question to find out how such a social network can still be developed into an active community.

Shamelessly we're re-using the good sides of current social media's practices, but we are also aiming to fix their shortcomings.

Pietu Pohjalainen */ M.Sc., Department of Computer Science, University of Helsinki.*

READY FOR BUSINESS: GNOBLES MAKES EVERYONE A BUSINESS ANGEL

FINLAND needs entrepreneurs who seek to build high-expectation companies. Software Factory acts as an incubator for young entrepreneurial minds. Gnobles is the result of the very first Software Factory undertaking. After just seven weeks of development, the service is ready for alpha testing as a business prototype.

Finland suffers from the lack of seed money coming from business angels. Still, many studies have demonstrated that it is the very early phase seed money that helps in getting the ideas fleshed out and companies into the next phase. Gnobles is designed to meet this need. It will make everyone a business angel in Finland and in Europe. It will demystify the role that a few business angels have taken.

Connecting great minds with common people

Today, if you want to support a young entrepreneur in getting his or her idea forward, there are no real channels available. Gnobles will be this link for everyone to use. If you have a great idea and you need a little money to get forward, you are likely to find these people in Gnobles. As a micro-investor, if you think you can afford a small investment ranging from 50 – 5000 euros to help a young entrepreneur in his or her journey, this is your place.

It is all about community feeling

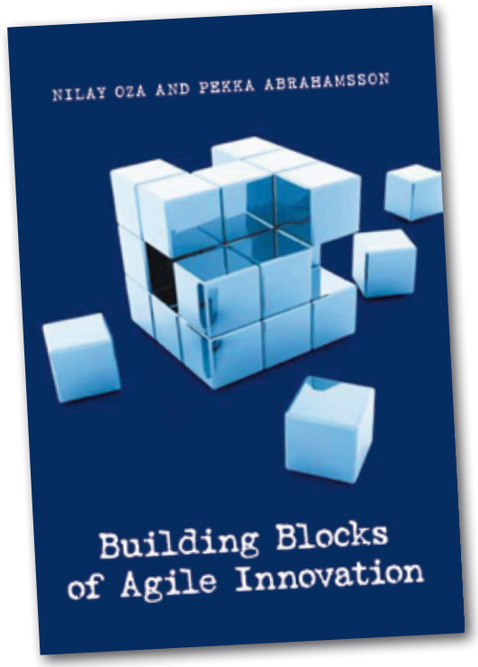
Gnobles is a community initiative. In Finland alone individual Gnobles chapters will be built in each of the major cities to help community growth. Individual investments made are very small but as a community action is brought together, they make a significant impact on the ability to help companies in making their steps towards high-expectation growth. Gnobles is designed to help young high-tech, mobile or web companies to find people believing in them to succeed. Gnobles believes that if your idea is backed up by hundreds of micro-investors, it will make a significant mental leap in driving further with the idea.

It is all about making Europe succeed

Gnobles will be launched in all European member states as a localized service in the coming years. This means the build-up of hundreds of local chapters in each country. Europe will succeed due to its great cultural diversity and unique history. Gnobles is built on this mental model.

Stay tuned for Gnobles launch at www.gnobles.com in the near future

Pekka Abrahamsson and Henri Karhatsu



SOFTWARE FACTORY PEOPLE BRIDGE AGILITY AND INNOVATION TOGETHER

AN ENTREPRENEUR has to be agile and innovative to succeed. Yet, agility and innovation are two distinct terms. Agility is generally associated with something lightweight and easy to move while innovation refers to introducing something new. Innovation is generally viewed as an idea that bears commercial value. By combining these two concepts in the context of software, we refer to agile innovation as the way to be innovative in every step of the software and service development.

Challenge the conventional wisdom

Learning the agile way and fostering continuous innovation is no longer just an option; it is the strategic direction that software companies need today. Many companies have already started initiatives and programs to foster organization agility and innovation. Ability to learn means the ability to challenge the expected.

In Software Factory, we need both top-down vision and bottom-up learning to foster agility and innovation. It has become more difficult to predict future needs as markets take unexpected turns. Therefore, it is not enough to merely follow markets and their development: instead, we should be able to shape how the future markets develop.

Stand in the crossroads

In our recent book [1], we collected a large base of experiences and research from the crossroads of agile and innovation. Software Factory is designed to be the crossroad where students, educators, entrepreneurs and researchers meet. This combination will yield unique results that we have already witnessed. It is our mission now to build on this base and continue exploring ways to build our innovative capabilities.

[1] Oza, N., & Abrahamsson, P. (eds.) (2009) The building blocks of agile innovation, http://www.agileinnovationbook.com/, BookSurge publishing

Pekka Abrahamsson and Nilay Oza

Nilay Oza *Dr. Nilay Oza is a senior research scientist at VTT Technical research centre of Finland. He conducts research, develops and manages R&D projects and offers consultation to companies as a member of VTT. His current areas of research interest include agile adoption and transformation, lean thinking, Green IT business models, and global software business.*

In the Factory pipeline: COLLAB – a Platform for Service Brokering

RECENTLY we have experienced the internationalization of business as a trend that not only is driven by outsourcing to multinational companies but that also affects business practices of small and medium sized enterprises. Current automation approaches include business process management paradigms have been extended to support cross-organizational processes. This can enable automating the interoperation of business activities between the collaborating organizations through service-oriented computing.

In the Software Factory project we plan to realize tendering and brokering functionalities or Collab registered services.

In such a setting, the question arises how to find and establish a business collaboration that is trustworthy in an anonymized electronic environment. Collab addresses this question as a platform, on which business service-offers and requests are registered together with information about contact persons and service-issuing organizations.

Furthermore, mashed up information is displayed on demand to allow site users an investigation of related business services, persons and companies together with additional business-critical information from the web-cloud.

In the Software Factory project we plan to realize tendering and brokering functionalities for Collab-registered services. On the one hand, the tendering/brokering involves traditional financial compensation for a service and on the other hand, we aim to integrate non-financial bartering techniques.

Alex Norta PhD., *is a post-doc researcher at the Department of Computer Science, University of Helsinki. His special research interest is cross-organizational business-process automation with service-oriented computing.*

SOFTWARE FACTORY BREWS GROWTH ORIENTED ENTREPRENEURS

WIDELY defined, entrepreneurs are persons who are willing to takes risks to develop new ideas to real businesses. Unlike ordinary managers, they prefer to try and fail rather than analyze systematically everything before action. Software factory aims at making the try and fail faster and less costly. Thus its customers focus is in entrepreneurs.

I have had the pleasure to follow personally the development paths of over 200 start-up companies. This experience has shown me that there exist two types of small companies.

First, there are the companies that are smaller versions of large corporations. The managers of these companies speak a lot about growth possibilities and follow their peers that have been growing. But they are also good at finding reasons why they cannot exploit possibilities available to them. Thus, every morning they end up doing the same things that they have been all the days before today. These companies achieve a modest growth at best.

Then, there is the other company type. These companies are most of the time in crisis or even on the verge of bankruptcy, as they spend all the resources available to them to pursue growth opportunities. Often they end up in a situation where they have no feasible way to continue operations and they go belly up. Many times the management can save the company from crisis, but has burnt its fingers during the process so badly that the company transforms to a risk adverse stable small company.

These lucky companies are then the ones that create millionaires and billionaires and the economic growth in modern economies.

Sometimes the company has luck before it runs out of money and succeeds in creating a growth business. As they have learned to grow in this process, they continue growing. These lucky companies are then the ones that create millionaires and billionaires and the economic growth in modern economies.

I can see a clear role for Software Factory in increasing the likelihood of the second type of a company to be lucky. As the factory helps entrepreneurs to try new ideas and reduces the amount of resources needed to test them, the potential growth engine company has a possibility to have more attempts to build business. Even though the probability of one attempt would remain the same as earlier, the company would have more chips in the venturing roulette.

Jussi Autere, *D.Sc. (Tech.) is founder and chairman of Gearshift Group Oy, where he also leads one of the strategy teams. Previously, Jussi Autere served as a professor of software business in Helsinki University of Technology. His special expertise is in corporate and technology strategy and he also has experience from M&A and venture capital fund raising.*

MAKING SOFTWARE FACTORY TRULY GLOBAL: THE SMART SOFTWARE FACTORY PROJECT

INDRA and Technical University of Madrid (UPM) have joined forces to launch a leading initiative to set up a global and distributed software factory in Spain in close cooperation with the top industrial and research collaborators in Europe.

Indra already has a distributed network of factories all over the world that support a large customer base with an increasing need for an ever reduced time to market. In this context, the project "Smart Sofware Factory" is borne, to fulfil the needs of software development in the next coming years.

UPM has been conducting research for a number of years in helping industry adopting the methodologies for making their processes more agile by introducing new paradigms in software development.

This project lays the foundation for the development of a global software factory, where new methodologies and software development methods can be experimented and demonstrated.

Both private companies and universities participate in this project. Indra Software Labs leads this initiative at the corporate level in Spain, with the close cooperation of Madrid, Technical University (UPM). In the international sphere, the main collaborators sharing the objectives of this project will be the University of Helsinki, Nokia, Nokia Siemens Network and ABB. Thus, deep university-industry cooperation is a key feature of the project.

The focus of the project research will be set on investigating lean / agile methodologies and techniques for collaborative development. It seeks innovation and continuous improvement in productivity and efficiency aspects (including energy efficiency) of the software development process.

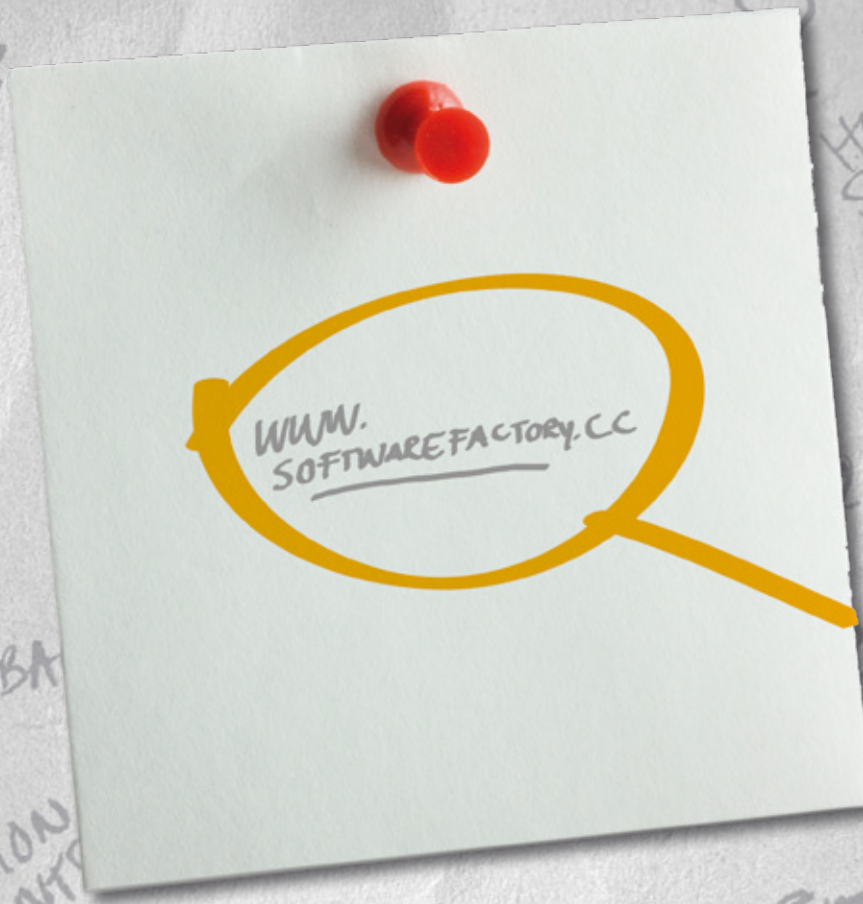
The project will also help create models and tools that will contribute both towards the implementation of the new processes and methodologies generated within the scope of the project, and the monitoring and tracking of the results. This last issue will be fundamental towards measuring the impact of applying the project results in the production process of a software company.

Juan Luis Martin Ruiz, *BSc in Computer Science from the Technical University of Madrid (UPM), works at Indra Sistemas as Project Manager for Energy and Operators Markets. He has 10 years experience working on R&D&i projects. For six years he has been working for international Telecom Operators as an expert on Operational Support Systems for the definition and coordination of proposals for Innovation projects focused on the Management of Networks and End-to-End Services. Those proposals were based on the prospection of new initiatives, new technologies (web services, semantic web, intelligent agents, etc.) and standards for development and management.*

Agustín Yagüe, *Associate Professor at E.U. Informatica (Informatics Engineering), Universidad Politécnica de Madrid, Spain. His research interests include software processes, software processes improvement, project management, agile development approaches, system and software testing processes and tools. He is one of the founding members of the Agile-Spain Association. He also cooperates with AENOR, the Spanish association for standardization, in the technical committee about Information Technology AEN/CNT71. Subcommittee SC-07: Software Engineering and Information Systems.*

Eloy Gonzalez Ortega, *BSc (Bachelor of Science) and MSc (Master of Science) in Aerospace Engineering by the University of Texas at Austin, works at Indra Software Labs (ISL) as Senior Manager and Project Director for Indra Sistemas and has been focused on Utilities markets, Energy Technologies and Internal Systems. More than 15 years experience working at international environments on the development and deployment of distribution network management products on Spain, Africa and Latin America, following CMMI methodology. Recently he has managed the ISL distribution products area serving numerous clients in several countries and has coordinated an important number of R&D proposals for domestic and international calls.*

Juan Garbajosa, *Professor at the Informatics School of the Technical University of Madrid (UPM), Spain, is the leader of the Systems and software technology research group (SYST-UPM). Among the research fields in which he is currently involved is the introduction of innovation into the software process, and the adoption of agile processes and practices in software intensive systems industry. He is active at standardization activities, being the convenor of ISO/IEC JTC1 SC7 WG20 Software and Systems Bodies of Knowledge and Professionalization. Before joining the University he spent more than fifteen years serving in aerospace industry, national and local government.*



BEHIND THE MAGAZINE ?

ISSN-L 1798-8845
ISSN 1798-8845

softwarefactory
learn.share.grow.

Editor-in-chief: Fabian Fagerholm, fabian.fagerholm@helsinki.fi
Department of Computer Science, University of Helsinki

Postal address: P.O. Box 68, FI-00014 University of Helsinki , FINLAND

Typesetting and layout: Pakkahuone Oy, www.pakkahuone.com

Printing: Helsinki Metropolia University of Applied Sciences

Next Software Factory Magazine will be out during the fall 2010.

BE PREPARED!