Swedsoft

# A Strategic Research Agenda for the Swedish Software Intensive Industry

*January 2010*
*Presented by Swedsoft*

***"There is a strong sense of urgency in the Swedish software industry.***

***We must become at least ten times more efficient to remain globally competetive.***

***The stakes are huge."***

**CONTENT:**




Editor: Adam Edström
Production: HeadQuarters Annonsbyrå, Stockholm
Print: Löwex, Växjö, 2010

# 1 Executive manifesto: A strong sense of urgency

**There is a strong sense of urgency in the Swedish software industry. The old approaches to software development are simply not sufficient for maintaining global leadership.**

**To remain a strong competitor, the Swedish software intensive industry must increase its efficiency and productivity, enhance the software quality and reduce lead times.**

**Several changes are required in the approach to processes, system architecture design and implementation, testing and maintenance, pared with improved management of software projects.**


**THE STAKES ARE HUGE.** About half of Sweden's exported products are critically dependant of software. A majority of the costs associated with R&D are already related to software development in many companies.

The importance of software development, including embedded software, for the Swedish industry has been highlighted by Swedsoft, in a report supported by VINNOVA, called *"Programvaruintensiva företag i Sverige. Konkurrenskraft och synlighet"* (Software Intensive Companies in Sweden. Competitiveness and Visibility, Swedsoft 2008. Available in Swedish at www.swedsoft.se and www.vinnova.se).

As software has become the prime industrial differentiator and basis for innovation, an increasing number of Swedish companies have more or less by accident found themselves software centric.

Volvo, the truck company, estimates that 90 percent of new innovations are in the field of electronics, and 80 percent thereof is software.

Some 25-35 percent of the cost of a car today is software, not to mention the much higher number for Ericsson's products. Similar assessments can be made at Saab, ABB and other major Swedish industries, operating within sectors as diverse as telecom, defence, automotive, medical technology, industrial automation or finance.

There is a relatively recent realization within these companies that they – and many others – share very similar challenges.

Thus, a new level of cooperation is necessary to bridge the efforts on all levels within the software domain – between corporations, between industries, academia and research institutes as well as between industry and academia.

Through the Swedsoft initiative, there is now an arena where representatives from all these bodies regularly meet to exchange experiences, ideas and visions about software development.

Swedsoft aspires to be a platform for future cooperation and interactivity, aiming to increase the competitiveness of Swedish software development.

**BUT FURTHER CHANGES ARE REQUIRED.** In particular research funding institutes, corporations and academia need to regard software development as a discipline in its own right, and organize their efforts to move away from today's situation of fragmented development in more or less isolated vertical silos.

Funding should be dedicated to pure software development research, not only to domains where software is a mean to an end.

Our conclusions and recommendations are further outlined in Section 3.

More incentives are also needed to encourage cross-fertilization. Today there is very little incentive for cooperation between universities, between corporations and between industry and academia. New mechanisms are necessary to leverage the true potential of the scattered competences.

**THE CHALLENGES ARE NUMEROUS.** Much of today's software runs 24 hours a day, 7 days a week. Complexity increases dramatically for each new project. Integration and test of legacy software, open source and other third party software are issues in dire need of new levels of understanding.

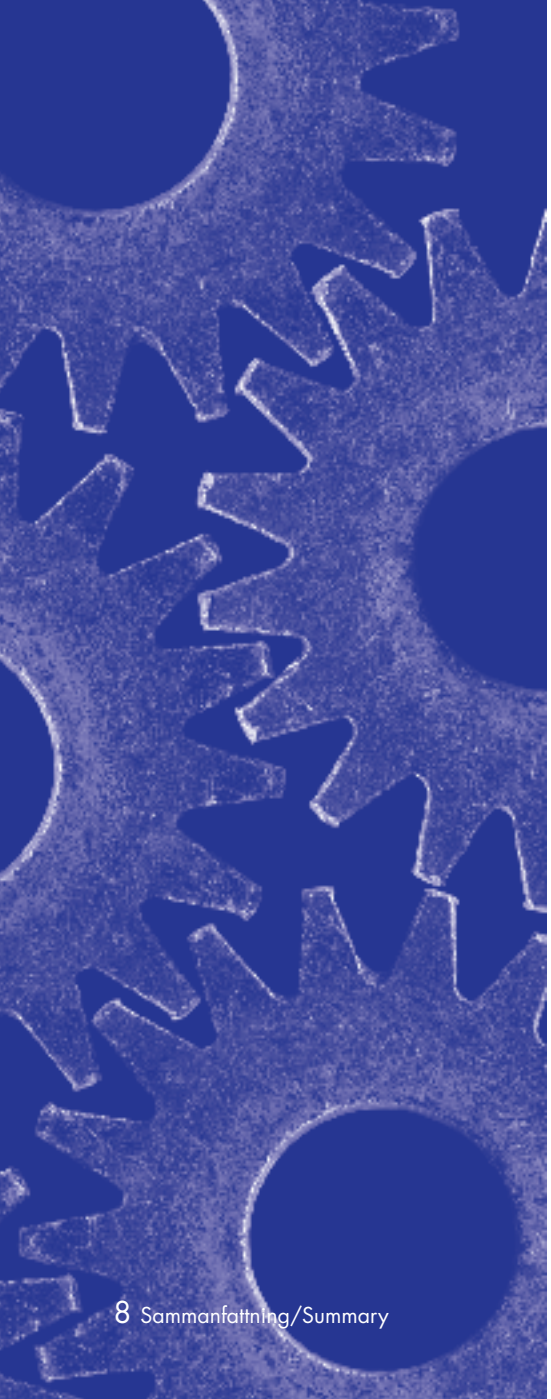
In spite of existing standards it is still difficult to predict the behavior of systems with many different components, especially real-time systems. Regulations, security, safety and environmental issues put further strain on the development.

One thing that is not increasing, at least not very rapidly, is the number of competent software developers in Sweden. Young people – especially young women – tend to prefer careers other than software engineering. Skilled software management is equally scarce, resulting in yet another industrial bottleneck.

**SO, WHY BOTHER?** With so many challenges and so few resources, what's the point?

Within Swedsoft, representing a substantial part of the Swedish software community, we believe it is timely for the Swedish industry to take the next steps in software development to maintain and increase our competitiveness. A national effort is of essence to maintain the Swedish industry's current position and become a world champion of software development within ten years. That goal is ambitious, but realistic.

*We hope to share our goals with as many influential people as possible. Please feel free to distribute this document to any and all concerned parties.*



# 2 Sammanfattning/ Summary in Swedish

**Den svenska mjukvaruintensiva industrin har ett gyllene tillfälle att nå en världsledande position inom tio år. Men samtidigt finns en stor oro inför framtiden. Dagens metoder för mjukvaruutveckling är inte tillfyllest för att bibehålla och utveckla en ledande position. Det finns ett stort behov, och en stark konsensus, om att nya grepp måste tas för att öka effektiviteten och produktiviteten i utvecklingsarbetet. Inte minst måste samarbetet öka mellan de idag många gånger isolerade öarna där mjukvaruutveckling bedrivs. Detta gäller industrin såväl som akademin.**

**ATT FRÅGAN ÄR AV NATIONELL VIKT RÅDER INGEN TVIVEL OM.** Hälften av all Sveriges export är kritiskt beroende av mjukvara. Mjukvara har blivit det viktigaste medlet för företag att differentiera sina produkter, och en majoritet av industrins FoU-projekt domineras av mjukvara. Ett exempel är Volvo Lastvagnar, som räknar med att 70 procent av alla innovationer är mjukvara. Ungefär 25-35 procent av värdet av ett fordon utgörs idag av mjukvara. Den siffran är ännu högre för Ericssons produkter, och ligger i samma härad för många stora svenska industriföretag oavsett om de är verksamma inom telekom, försvar, medicinsk teknik, fordon, industriautomation eller finanssektorn.

Ändå är det inte förrän relativt nyligen som de svenska mjukvaruintensiva företagen har insett att de sitter i samma båt och att deras utmaningar är gemensamma. Just därför måste samarbetet tas till en ny nivå, det gäller såväl mellan industriföretag som mellan industri och akademi och mellan olika akademiska institutioner. Swedsoft är ett initiativ som syftar till just sådant samarbete, och aspirerar på att bli den arena där en rad samarbeten kan initieras och genomföras. Allt i syfte att öka konkurrenskraften för den svenska mjukvaruintensiva industrin.

**SWEDSOFT FÖRESLÅR I DENNA SKRIFT TRE ÅTGÄRDER** för att öka konkurrenskraften för den svenska mjukvaruintensiva industrin.

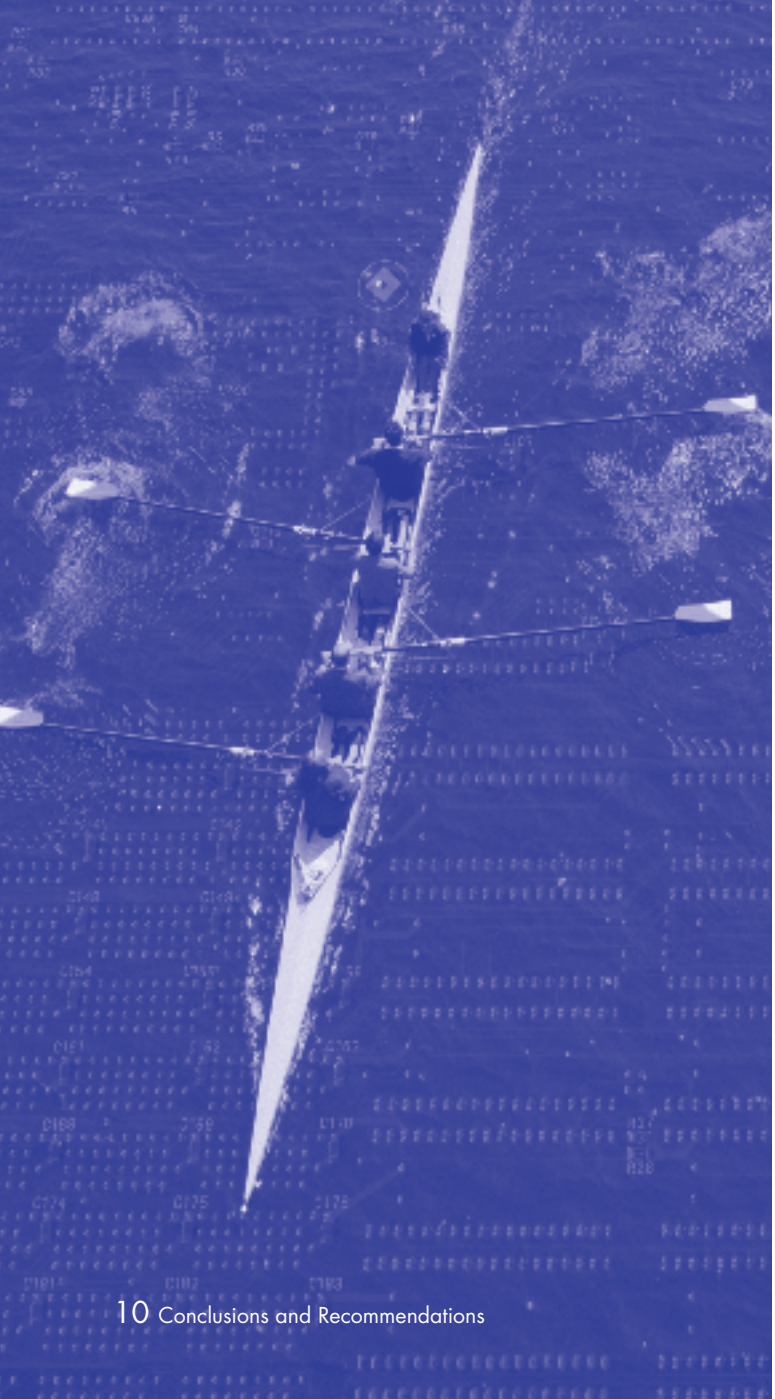
1 ***FÖR DET FÖRSTA BÖR MJUKVARUUTVECKLING ERKÄNNAS SOM EN EGEN DISCIPLIN,*** inom såväl industri och akademi. Det är en grundförutsättning för att komma bort från dagens fragmenterade utvecklingsansträngningar, och en rimlig konsekvens av att mjukvara idag är det främsta medlet för industrin att differentiera sina produkter.

2 ***FÖR DET ANDRA KRÄVS ÖKAD STIMULANS OCH NYA STRUKTURERADE MEKANISMER FÖR KORSBEFRUKTNING*** mellan mjukvaruutvecklingsaktiviteter, må de finnas inom industrin, akademin eller annorstädes.

3 ***FÖR DET TREDJE BÖR MER PENGAR ANSLÅS TILL FoU-SATSNINGAR INOM MJUKVARUUTVECKLING,*** och inte bara där mjukvara ses som ett medel för att uppnå andra mål. Ett nationellt program med fokus på att göra mjukvaruutveckling minst tio gånger mer effektivare än idag, vore ett viktigt bidrag till Sveriges industriella framtid.

En mer detaljerad redogörelse för våra mål, på kort, medellång och lång sikt, återfinns i avsnitt 4, Goals and Wanted Position.

Med denna skrift hoppas vi inom Swedsoft, som representerar en stor del av den svenska mjukvaruintensiva industrin och akademin, få till stånd de förändringar som krävs. Vi hoppas att på stöd från så många inflytelserika personer som möjligt. **SPRID GÄRNA SKRIFTEN.**



# 3 Conclusions and recommendations

**This document is set out as a Strategic Software Research Agenda by and for the software intensive Swedish industries. It aspires to highlight the importance of software for the Swedish industry, the needs and the sense of urgency whitin the Swedish software community for change and action. To that end we offer some recommendations for the immediate and long term future.**

As ever, new innovations built on the legacy of spirited Swedish engineering can lay a new foundation for future wealth.

But to pave the way, the software discipline needs to be strengthened and new bridges must be established between islands of software competence.

Attitudes must change and a new willingness to work hard to reach world championship must be established.

If the right measures are taken, Sweden could emerge as a global software leader, with a more complete understanding of the development process and an industry able to develop software at least ten times more efficiently than today.

THE CURRENT SITUATION IN THE SWEDISH SOFTWARE COMMUNITY, with development and resources scattered around in isolated silos and software development denied appropriate recognition as a discipline, is of course far from ideal, and hardly the bedrock foundation necessary for further innovation and economic growth.

**WE BELIEVE THAT THREE CHANGES ARE OF ESSENCE TO AMEND THE SITUATION:**

1

***RECOGNIZE SOFTWARE DEVELOPMENT AS A DISCIPLINE IN ITS OWN RIGHT***

First of all, since software is the prime differentiator for a large and growing number of Swedish industrial products, software development must be recognized as a discipline in its own right. Actions must be taken within industries, universities and other relevant bodies to modify and reorganize their software development research and related activities according to this recognition.

2

***CREATE INCENTIVES FOR CROSSFERTILIZATION***

Secondly, new incentives must be created to improve the current circumstances with software development mainly going on in isolated silos.

Crossfertilization between companies, between universities and between academia and industry needs further impetus to transform from what is currently a rosy vision to a situation of innovation-enabling cooperation.

A humble demand is also that more crossfertilization occurs among government bodies and foundations responsible for R&D funding, as well as between relevant government departments. Such positive developments would support and nurture the crossfertilization across the entire Swedish software development ecosystem.

3

***DEDICATE GRANTS TO A MAJOR NATIONAL SOFTWARE INITIATIVE***

As a consequence – since software is a significant driver for innovation – research and development grants should be dedicated to the specific field of software development, rather than confining allocations to domains where software is seen as a necessary prerequisite for success.

A major national initiative, focusing on making software development at least ten times more efficient, is called for in the interest of the Swedish industry's future competitiveness.

***"A major national initiative is called for in the interest of the Swedish industry's future competitiveness."***

# 4 Goals and Wanted Position

**Our vision is that by 2020, the Swedish software intensive industry should be second to none. It should exhibit drive, and flexibility of business models relevant for creating large complex software intensive systems.**

By 2020, we should fully understand how to efficiently develop software intensive systems, with high quality and excellent usability. And we should be able to do this at least ten times more efficiently than today, alternatively be able to produce ten times more complex software without additional manpower.

The goals are illustrated in this matrix:

| Strategic need | Short term, 2010-2012 | Mid term, 2013-2016 | Long term 2016-2020 |
|---|---|---|---|
| **Management of the SW discipline. Including increasing flexibility, realization of business models and recruitment of SW competence** | 1) Industrial and academic focus on SW as a discipline.<br>2) New model how to recruit students to university SW programs.<br>3) New teaching curriculum ore closely related to middle- and long term industrial needs. | 1) Large-scale national SIS development cooperation projects under way.<br>2) Structured methods for assuring supply of SW project management.<br>3) Good architecture support, i.e. easy to understand, maintain and extend. | 1) We fully understand how to develop SIS with efficiency, quality and cost control.<br>2) We develop and simulate systems (SW and HW) in an mixed and integrated development environment.<br>3) We have assured continuous competence supply for SIS development. |
| **SW Engineering (addressing the need for increased productivity in terms of reducing lead-time and cost, and dramatically improving the efficiency of software engineering and adding the aspect of swedish industrialization.) Includes cost efficient quality assurance, earlier verification of designs and improved relevant system understanding.** | 1) The tool environment should support fast feedback, e.g with simulation, for the user.<br>2) Expressiveness of design and architecture to stakeholders.<br>3) Expressiveness of formalism and consistency so the SIS does what it is designed to do.<br>4) Full utilization of domain specific "languages"/ instruction sets.<br>5) Scalability. | 1) Everything on the same abstraction level (implement, simulate, test and debug) including legacy systems.<br>2) More user friendly and easy to learn. Integration and full support throughout the way of working chain.<br>3) The IDE is more active in analyzing the design (in all phases) and providing feedback to the developer.<br>4) Verification of the design is done at high level, earlier than today. | 1) We can develop SIS at least 10 x more efficient than today. Or, alternatively, we can reduce lead-time with increased quality of end-to-end SIS development to 1/10 of today.<br>2) We have ensured working interchange of SW tools. |
| **SW execution environment (portability, scalability, safety critical systems, long life cycle, dependability).** | 1) We can implement, run and interchange SW components smoothly in specific domains (telecom, automotive, etc).<br>2) Verification on component/ subsystem level is sufficient for the integration | 1) We can implement, run and interchange SW components smoothly across domains.<br>2) Software developed for SIS is predictable in different configurations. | 1) We can implement, run and interchange SW components smoothly in all systems, including safety-critical SIS.<br>2) We can manage the variability of the SIS without adding to the maintenance cost or R&D.<br>3) Product test should not be needed since the general PLA is tested and the variants' quality is deduced therefrom. |

*(Note: SIS = Software Intensive Systems. IDE=Integrated Development Environment. PLA=Product Line Architecture)*

*"Too few people have the capacity and ability to successfully manage large software development projects"*

Many steps are necessary to achieve these goals. Improved understanding of the management of software development is perhaps the most pressing. Too few people have the capacity and ability to successfully manage large software development projects, resulting in products with less than optimal performance and/or lost windows of opportunity.

**NEW "BEST PRACTICES"** are needed to optimize the make/buy/borrow challenges. Large software projects could become much more efficient by raising the level of abstraction, with execution and testability on all levels and full maintainability of legacy software. Tools should have high level check functions – at the design level rather than the code level – and support instant feedback to the user, e.g. by simulation. Increased use of model-based development methodologies offers one possible way forward.

*"Universities need to focus more on software devlopment as an academic discipline and secure the understanding of the software product development cycle."*

**MORE STRINGENT SPECIFICATIONS AND STANDARDISATIONS** of the interfaces between different software components are necessary, not least to deal with the plethora of open source and other third party software. The performance of the software system must be also predictable in other, different configurations, eliminating some of the needs for test. Verification on subsystem level should also be sufficient for the integration, to decrease the number of loops in the development chain.

**BETTER PREDICTABILITY** is also needed, not least for automotive and industrial software to ensure product longevity and industrial competitiveness. The software should be developed for minimizing maintenance efforts, e.g. with possibility to upgrade continuously, in the field, sometimes decades after its initial deployment.

**OUR UNDERSTANDING OF SOFTWARE FUNCTIONALITY MUST IMPROVE** to a state where we have development environments supporting expressiveness, formalism and consistency to allow the software intensive systems to do exactly – and only – what it is designed to do. Domain specific languages or instruction sets could be required to fully leverage this potential.

**WE ALSO NEED IMPROVED CERTIFICATION PROCESSES,** not least for safety critical systems. These systems should not be harder or more costly to develop than any other systems. By 2020, safety critical systems should be executable on shared hardware and easily connectable to the outside world. Issues of flexibility, security, integrity, portability and migration must be natural ingredients of any software development ecosystem.

**A NEW MODEL FOR RECRUITING AND EDUCATING** students is of essence to assure a sufficient supply of competence for all these tasks. Universities need to focus more on software development as an academic discipline to develop and accredit graduates who understand the software product development cycle. A more industry-driven curriculum would also be appreciated by several software-intensive corporations.

The industry must be more vocal about the need for SW talent, to attract more young people to the field.

Industry and academia ought to become much better at synchronizing their software development roadmaps and match their strategies for software development management.

# 5 Current situation and challenges

### THE INDUSTRY THAT IS NOT

**One of the main challenges for the Swedish software industry is that it is in general not regarded as an industry.**

This might sound like a paradox, but software is still viewed as an add-on to many products and services. But software is in fact the main driver for innovation and product development within a spectrum of industries ranging from telecom and automotive to consumer products, industrial automation, medical equipment, defence and finance. Nevertheless it is only rarely treated as a core competence and seldom given the strategic attention it deserves. The current trend of outsourcing

software development to low-cost countries is unsustainable and runs the risk of seriously threatening Sweden's aspirations for world leadership in this field.

**SOFTWARE DEVELOPMENT IS MOST OFTEN SCATTERED**, the competence and products developed in isolated vertical silos with little or no peer exchange between software developers. This is true within industries and single companies, within academia and also within government and research funding bodies. One of Swedsoft's main ambitions is to change this situation, and have software development treated as a more coherent activity.

It should be stressed that the software product life cycle consists of much more than simply writing code. In fact, the coding itself amounts to only a small fraction of the development task. Large industries estimate that coding accounts for only about 7 percent of the total software development effort. The majority of the work has more to do with requirement handling and understanding, architecture design, system design, integration, specifications, documentation, verification, test and other higher-level activities. Creating brand new programs is a rare task – most of the time is spent making changes and additions to existing code or figuring out what has gone wrong and why. The all too common view that software development equals coding could be one reason why software development is not given the attention it deserves.

***"The software product life cycle consists of much more than simply writing code."***

### INCREASING CHALLENGES

The technical challenges for software development have been thoroughly analyzed and described by academics, industrialists and not least by the media. The picture is clear – this is an activity where customer demands and new technology increase the challenges of quality attributes such as complexity, flexibility, usability, safety, security, reliability, maintainability (and a growing list of other "–ilities"). A common, if perhaps surprising, conclusion is that many of these challenges cannot be addressed by simply adding more manpower. Even if there was an abundance of software engineers – which is not the case – all the recipes for success emphasizing using smarter methods, better design environment and higher levels of abstraction, rather than relying on increased headcounts within software departments.

This situation has paradoxically some beneficial implications for Sweden. The international software communities, i.e. the competitors to Swedish companies, face the very same challenges. Brute force is simply not the answer to these problems. By focusing

on the right issues, the Swedish industry thus has a golden opportunity to emerge as a leader in the software field.

Some challenges deserve to be set in somewhat broader context. Software development is today often a global activity, where a project could encompass several development teams around the world. While this in theory opens up huge possibilities for efficiency and time to market, handling distributed development with different competences in different places, more often than not including customization for several different end markets, it also generates a management challenge of great dignity.

**NEW RULES AND REGULATIONS REGULARLY AFFECT SOFTWARE PRODUCTS.** Regardless whether these regulations emanate from government bodies or from standards committees they should be taken into consideration as early as possible in the development projects in order to be properly fulfilled.

***"For industrial applications there's no room for failure."***

For industrial applications – such as telecom equipment or industrial robots – there is no room for failure. These devices can run for decades, 24 hours a day, 7 days a week, and simply cannot risk unscheduled unavailability. Safety critical software in e.g. defence products, automotive products or nuclear power plants adds another dimension of dependability demands. The renowned software scholar Barry Boehm states that "Better models for integrating 'systems of systems' are crucial for the future" and it is hard to disagree. *(Barry Boehm, A View of 20th and 21st Century Software Engineering, International Conference of Software Engineering, 2006).*

Legacy software is a major contributing factor to the overall complexity, often combined with increased demands for communication. Customer demands for devices with increased communication capability, lower power consumption, ease of use and high performance for lower costs mean increased challenges for efficient software development and proper understanding of software architectures.

**NEW TECHNOLOGIES**, such as new software and hardware architectures, multicore microprocessors or higher abstraction levels, have the potential to solve some of the performance issues needed to fulfil the demands of tomorrow. But they also add to the complexity. Evolved tools and methods are necessary to unleash the potential performance for the whole product life cycle development. Scalability should be stressed, though, since many new tools and methods may work well on a small scale, but fail to deliver when upscaled to industrial levels.

### BUYING AND SELLING ISN'T WHAT IT USED TO BE

Major international software companies – IBM, Sun (now Oracle), and several others – have changed the issue of open source software (software components available free of nominal charge, most often downloadable from the Internet) from a simple choice of yes or no, to today's more mature and nuanced approach of how much, and for what purposes, it makes best sense. Again, the issues of legacy, testability and efficiency must be dealt with, but with open source several legal issues are added to the picture.

A widespread insecurity of future business models for software is a further concern. The nature of software makes it possible to upgrade products as diverse as trucks, telecom equipment, industrial robots and defence equipment. Each of these can have new function and features added years or decades after the hardware was delivered. By that time the owner could be completely different from the original customer, the ecosystem which the product operates in could have changed radically, and so could business or regulatory demands.

To determine the proper value of the software, and find viable business models under such circumstances, is a challenge for many Swedish industries, and it does not get easier with the proliferation of open source software. Securing the right competence for IPR (intellectual property

rights) and business model development in this environment is of pressing concern to the software intensive industry, not least in order to treat open source software as a possibility rather than a threat.

The customers – the end users – hardly ever care about the intricacies of the software or how it is implemented. From the customer's point of view the software simply has to work. This requires the software to be developed in a mature way, conformant to standards and best practices. It should be mentioned, though, that some advanced customers can have demands like adhering to a certain standard (e.g. IEC61508 SIL) or that a specific technology is used to ensure interoperability with other systems.

### The good heart of Swedish engineering

The picture of the current situation would not be complete without a mentioning of the innovation climate in the Swedish industry, where many of these metaphorical clouds are lit up by proverbial silver linings. The Swedish industry is generally considered to be innovative, system-centric, with a high technological level. Scientists and engineers are usually willing to share their knowledge and create relations and connections without too much attention to prestige or status. There are elements of consensual decisionmaking in the

Swedish culture, with individuals willing and able to ask difficult questions as well as making tough decisions. Should a problem occur, the Swedish focus is usually on finding a solution, rather than searching for scapegoats to blame. This is also reflected in the Swedish management style, which is usually described as coaching, informal and supportive rather than hierarchical and bureaucratic.

These are all very valuable properties and are indeed almost prerequisites for more intense interaction between companies, teams, universities and research bodies. The fact that in general the Swedish software engineers are well educated, internationally experienced, proficient in English and committed teammembers brightens the outlook even more. There are of course counterexamples to all of these so called soft skills, but some of the Swedish industrial successes could still be attributed to this general description.

***"There are elements of consensual decisionmaking in the Swedish culture, with individuals willing and able to ask difficult questions as well as making tough decisions. Should a problem occur, the Swedish focus is usually on finding a solution, rather than searching for scapegoats to blame."***

### PIGGYBACKING ON RESOURCEFUL PEERS

Many of the aforementioned issues have of course been addressed by several national and international research programs.

Several noteworthy national programs have enjoyed support mainly from VINNOVA, the Swedish Research Councyil, the Swedish Foundation for Strategic Research and the Knowledge Foundation. VINNOVA supports several national programs where software development has great wieht, most notably Forska & Växa ("Research & Grow") and Banbrytande IKT ("Groundbreaking ICT"). The earlier program Nätverksbaserad programvaruteknik ("Network-based software technology"), conducted from 2001 to 2008, should also be mentioned in this context. These programs are, and have been, of tremendous value for the Swedish industry and academia. However, they have not been focused on software in itself, but instead on software as means to an end and only a fraction of their budgets have been allocated to software development. Thus, they do not specifically address the issues for this SRA: software development and the management thereof.

On a European level, the Framework Program 7 is, along with Artemis and ITEA 2, probably the most influential currently ongoing program with high software content. Artemis is supported by the European Union and the member states, and ITEA 2, also supported by the member states, is a program within the Eureka cluster. Much know-how and competence generated by these programs can and should be applied in the Swedish industry, and future national research programs would gain from close connections to these programs.


However, neither Artemis nor ITEA 2 addresses the fundamental issue of attitude towards software development as a discipline, nor management of software. And due to their scope they also don't address some of the hard issues of SW development – ITEA 2 is focussed on pre-competitive R&D for software intensive systems, and Artemis is geared towards embedded systems, including hardware. Both programs undoubtedly touch and overlap some of the issues pressing the Swedish software industry, but it is our belief that much more can be done by national and international bordercrossing exchange of knowledge, ideas and experiences.

In it's latest roadmap (*ITEA Roadmap for Software-Intensive Systems and Services, 3rd edition, February 2009*) there is a shift in perspective from a previous technology-centric to a more human centric approach. Technology development is treated in a domain named SSSC (Services, Systems and Software Creation), addressing some of the same issues as are considered in this document. Four points are emphasized for increased speed of software development: Better organization of the development process, Automation of component integration, Interoperability of components/services/systems and Automation of software generation (see pp 140-141 in the roadmap). Swedish R&D within software development should by all means keep an eye on the development within this part of ITEA 2, but it is Swedsoft's belief that even more targeted efforts are necessary to obtain the wanted goals, as laid out in some detail in Section 4 of this document.

***"Swedsoft believes that a national effort making Swedish software development at least ten times more efficient than today would be of huge national interest."***

Artemis also addresses some of the points raised in this Swedsoft document, not least the need for cross-fertilization between different industrial domains. Some of Artemis' objectives are also similar to Swedsoft's – reducing the cost of system design, managing increased complexity, reducing the effort and time required for validation and certification and achieving cross-industry reusability of embedded systems devices (*Artemis Strategic Research Agenda 2006, Design Methods and Tools, p 6*). However, Artemis' goals are not as far-reaching as the goals in this document, and the program has a different scope – hardware and software for embedded systems. Nevertheless, results and experiences from Artemis could kick-start a Swedish program on software development.


In the automotive industry there is today hardly any software development going on without adherence to AUTOSAR, Automotive Open Systems Architechture.

The AUTOSAR initiative, with roots in the European automotive industry and enjoying current backing by a majority of Western automotive companies and their supply chain, aims primarily to standardize software architecture and interfaces between software components for automotive purposes. AUTOSAR is arguably the most comprehensive industry standardization effort ever in the field of software. The goals are modularity, scalability, transferability and re-usability of functions to provide a standardized automotive systems platform, enabling configuration and optimization to meet runtime requirements. And although AUTOSAR was conceived to meet the needs of the automotive industry, there are several elements therein that could apply equally well to other industry niches.

Swedsoft believes that a national effort addressing how Swedish industry and academia should go about to make Sweden a global leader in software development, at least ten times more efficient than today as described in Section 4 “Goals and wanted position”, would be of huge national interest. Of course, such a program should draw upon experiences of the aforementioned programs, but retain a stricter focus on software development.

# 6 Threats

**If there ever was a moving target, then software development is certainly one of the most mobile.**

The international competition is fierce, to say the least. Parallels are plenty, but perhaps the story of Alice's Adventures in Wonderland best illustrates the situation for the software industry:

"Well, in our country," said Alice, still panting a little, "you'd generally get to somewhere else – if you ran very fast for a long time, as we've been doing".

"A slow sort of country!" said the Queen. "Now, here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!"

*(Lewis Caroll, Through the Looking Glass, ch 2).*

If, like the out-of-breath Alice, we stick to current technology, methodologies, design environments and management practices and don't act to improve the situation at home, the competition abroad will be ahead of us quickly. Closing the gap will under such circumstances be even more difficult, since many talented developers will look outside Sweden for more interesting opportunities. A diminishing software intensive industry will severely impact on not only the competitiveness of the Swedish industry, but also on the entire Swedish job market.

**SOME COMPETITORS HAVE ADVANTAGES OF SCALE.** Universities in China and India churn out software engineers in amounts Sweden could only dream of. Chinese and Indian researchers also play an increasingly prominent role at many of the best technical universities on earth. It would be naïve to claim that this geopolitical change would hold no implications for the Swedish software intensive industry.

Meanwhile, the apparent lack of interest among local young people in higher technical education is of great concern. Most software courses remain dominated by young men, with young women conspicuously absent. Solving these issues will go beyond the scope of this document – suffice it to say that neither academia nor industry, nor society as a whole, has done enough to address these pressing issues.

**THE COMPETENCE OF NEWLY GRADUATED ENGINEERS IS YET ANOTHER SOURCE OF CONCERN.** Industry and academia agree that the competence of the emergent graduates has great potential for better matching the industry needs. Ericsson claims that under current circumstances new recruits need training in software product development from day one, and would be very happy to see this situation rectified.

**A MISMATCH ALSO EXISTS BETWEEN THE RESEARCH UNDERTAKEN BY ACADEMIA AND THAT REQUIRED BY INDUSTRIES.** Although Swedish software research is very competititive in a select number of niches, many industry representatives feel that several other niches are not addressed as properly by the Swedish research community as the industry would like. Representatives from both sides agree that the academic community's understanding of industrial needs could be better, and both sides have called for incentives to encourage academia to better adapt to industrial needs.

# 7 Strategic needs

**Given the number of challenges and the shortage of resources, priority must be given to needs considered strategic to the industry.**

This list aims to emphasize some of the most pressing areas where work has to be done and new competence gained to reach the goals outlined in Section 4 and reiterated below.

### MANAGEMENT OF THE SOFTWARE DISCIPLINE

| Strategic need | Short term, 2010-2012 | Mid term, 2013-2016 | Long term 2016-2020 |
|---|---|---|---|
| **Management of the SW discipline. Including increasing flexibility, realization of business models and recruitment of SW competence** | 1) Industrial and academic focus on SW as a discipline.<br><br>2) New model how to recruit students to university SW programs.<br><br>3) New teaching curriculum closer related to middle- and long term industrial needs. | 1) Large-scale national SIS development cooperation projects under way.<br><br>2) Structured methods for assuring supply of SW project management.<br><br>3) Good architecture support, easy to understand, maintain and extend. | 1) We fully understand how to develop SIS.<br><br>2) We develop and simulate systems (SW and HW) in an mixed and integrated development environment.<br><br>3) We have assured continuous competence supply for SIS development. |

*(Note: SIS = Software Intensive Systems)*

**SOFTWARE MANAGERS NEED A BROADER UNDERSTANDING OF CONTEXT AND PROCESSES** required for developing software intensive systems. Such managers need to take into account not only the technical demands but also focus on the business models and market needs. A worthwile goal would be to better exploit the various existing business models that software offers, and perhaps develop new models where applicable.

For software development, management needs better environments to understand the make/buy/borrow issue, and the consequences of such decisions. Designing with the optimal software components regardless of source (a.k.a. Open Innovation) demands thorough understanding not only of the software development process, but also profound expertise in the domain where the design is aimed (e.g. telecom, automotive or defence). Another important aspect that needs improvement is the art of reaching solutions where hardware and software are as independent as possible. A better environment for understanding the consequences of different implementation alternatives would also be of great value in answering questions about what should be developed in software, in hardware or as a service.

**THE IMPORTANCE OF CULTURAL DIFFERENCES AND LEADERSHIP STRENGTHS SHOULD NOT BE UNDERESTIMATED.** While managing the software development team, managers must also find methodologies for determining what to outsource and procuring the best available third party services, sometimes in globally distributed teams.

**HARDLY ANY SOFTWARE IS DEVELOPED WITHOUT A STEADY SUPPLY OF COMPETENCE**, and assuring the competence supply is a pressing issue for all development of software intensive systems. A new model for recruiting students to higher software education is an issue that simply cannot wait. Tomorrow's software graduates need a much wider set of skills, where coding ability is probably the least worrysome. The industry already makes clear that it needs software-savvy engineers with a willingness to grasp industrial demands on complexity, efficiency and quality, paired with social, linguistic and management abilities. It goes without saying that this will not happen unless industry, academia and relevant government bodies rise to this particular challenge with a new attitude.

**THE INCENTIVES FOR THE INDUSTRY TO EDUCATE AND RECRUIT NEW SOFTWARE MANAGEMENT SHOULD ALREADY BE STRONG ENOUGH.** Key personnel will retire within a few years, and urgently need to be replaced without losing technical know-how or customer confidence. This topic might be ideal for inter-corporational workshops and exchange.

*"A worthwile goal would be to better exploit the various existing business models that software offers"*

| Strategic need | Short term, 2010-2012 | Mid term, 2013-2016 | Long term 2016-2020 |
|---|---|---|---|
| **SW Engineering (addressing the need for increased productivity in terms of reducing lead-time and cost, and dramatically improving the efficiency of software engineering, and adding the aspect of SW industrialization.**<br><br>**Cost efficient quality assurance.**<br><br>**Verification of the design is done for each step in SW development. Simulation of the system at each step during the end-to-end SW development.**<br><br>**Improved relevant system understanding.** | 1) The tool environment should support fast feedback, e.g with simulation, for the user.<br><br>2) Expressiveness of design and architecture to stakeholders.<br><br>3) Expressiveness of formalism and consistency to get the SIS to do what it is designed to do.<br><br>4) Full utilization of domain specific "languages"/ instruction sets.<br><br>5) Scalability.<br><br>6) Academic curriculum | 1) Everything on the same abstraction level (implement, simulate, test and debug) including legacy systems.<br><br>2) More user friendly and easy to learn. Integration and full chain support.<br><br>3) The IDE is more active in analyzing the design (in all phases) and providing feedback to the developer.<br><br>4) Verification of the design is done for each step in the SW development. | 1) We can develop SIS 10 x more efficient than today. Alternatively reduce lead-time with increased quality of end-to-end SIS development to 1/10 of today.<br><br>2) We have ensured working interchange of SW tools. |

*(Note: SIS = Software Intensive Systems. IDE=Integrated Development Environment)*

With every product being custom made, whether a mobile phone, an industrial robot, a vehicle or a military aircraft, managing variability is a key through the entire software design process. Developing the best architecture and applying new technology such as multicore or parallell computing where it makes best sense will become prime competencies, but need more research for full understanding and for reaching the goal of increased productivity by at least a factor of ten. Ostensibly reducing complexity by in fact adding complexity is an easy trap to fall in, and should of course by all means be avoided.

**RAISING THE LEVEL OF ABSTRACTION IS ESSENTIAL FOR HANDLING THE VARIABILITY ISSUES WITH IMPROVED DESIGN EFFICIENCY.** This area also deserve more research, not least since the solutions must

be paired with possibilities to execute, verify and test the software on all levels, with tools giving fast feedback to the user, using simulation and other verification technologies.

**DEVELOPING AN UNDERSTANDING FOR COST EFFICIENT QUALITY ASSURANCE** should also be high on the agenda. We need to reach a state where software for software intensive systems is predictable in different configurations, not least to deal with updates decades after the product was delivered. Ideally, later date product tests should not be needed since the quality of the integrated variants can be deduced from verification on component or subsystem level.

For safety critical systems the desirable position is to create systems which are as easy and as efficient to produce as any other system. Security and integrity aspects should and could become part of any system development. A more effective certification process is imperative for this goal.

### SOFTWARE EXECUTION ENVIRONMENT

| Strategic need | Short term, 2010-2012 | Mid term, 2013-2016 | Long term 2016-2020 |
|---|---|---|---|
| **SW execution environment**<br><br>**(portability, scalability, safety critical systems, long life cycle, dependability).** | 1) We can implement, run and interchange SW components smoothly in specific domains (telecom, automotive, etc).<br><br>2) Verification on component/ subsystem level is sufficient for the integration | 1) We can implement, run and interchange SW components smoothly across domains.<br><br>2) Software developed for SIS is predictable in different configurations. | 1) We can implement, run and interchange SW components smoothly in all systems, including safety-critical SIS.<br><br>2) We can manage the variability of the SIS without adding to the maintenance cost or R&D.<br><br>3) Product test should not be needed since the general PLA is tested and the variants' quality is deduced therefrom. |

*(Note: SIS = Software Intensive Systems, PLA=Product Line Architecture)*

**MAJOR PROGRESS HAS ALREADY BEEN** made in specific domains, not least in automotive systems thanks to the Autosar initiative. But Autosar is not necessarily the key to success. In an ideal world software should execute in an environment where different software components can be implemented, run and interchanged smoothly, more or less like Lego bricks. Upgrades or changes of electronic hardware platforms should be easy requireing no reverification or recertification. And although this ideal state probably won't occur within the foreseeable future, it is quite possible to close the gap between the ideal and the real.

# Appendix. About Swedsoft and the Strategic Research Agenda

Swedsoft is an industry initiative to strengthen Swedish competitiveness with regards to research and development of software intensive systems, services and products.

**SWEDSOFT'S FOUNDERS AND FINANCIERS** include ABB AB, Ericsson AB, Saab AB and Arcticus Systems AB from industry, as well as the Swedish Institute of Computer Science, Blekinge Institute of Technology, IT University of Gothenburg, Linköping University, Lund University, and Mälardalen University. They have all contributed to this Strategic Research Agenda, as have representatives from Volvo AB, Enea AB, Sony Ericsson and the KTH Royal Institute of Technology. Elektroniktidningen's editor Adam Edström wrote the document.

**SWEDSOFT'S BOARD MEMBERS ARE:** Anders Caspár, *Ericsson AB* (chairman); Lars-Olof Gustafsson, *Ericsson AB;* Magnus Larsson, *ABB AB;* Göran Backlund, *Saab AB;* Kurt-Lennart Lundbäck, *Arcticus Systems AB;* Staffan Truvé, *Swedish Institute of Computer Science;* Claes Wohlin, *Blekinge Institute of Technology;* Thomas Arts, *IT University of Gothenburg;* Kristian Sandahl, *Linköping University;* Per Runeson, *Lund University* and Christer Norström, *Mälardalen University*.

Swedsofts director, Christer Bengtsson, can be contacted at christer@swedsoft.se

More information about Swedsoft is found at www.swedsoft.se.

VINNOVA This report was supported by VINNOVA.

*“Attitudes must change. A national effort is of essence to become a world champion of software development.”*

# A unique cooperation

**THIS STRATEGIC RESEARCH AGENDA IS THE RESULT OF A UNIQUE COOPERATION** between software intensive industry corporations, universities and research institutes in Sweden. Never before has there been a consensus on this level about the needs and the challenges for the Swedish software intensive industry. Never before has the sense of urgency been so strong in this community.

**THREE THINGS NEED TO BE DONE.** Software has to be recognized as a discipline in its own right. New incentives must be created for crossfertilization, in order to create innovation-enabling cooperation. Last but not least we call for a national initiative with grants dedicated to the specific field of software development.

**THE AGENDA IS CONCEIVED AND PRODUCED BY SWEDSOFT,** an industry initiative aimed at strenghtening Swedish competitiveness with regards to research and development of software intensive systems, services and products. The content of the agenda is supported by a vast range of large and small Swedish industries, as well as most major Swedish universities. VINNOVA supported the production. Elektroniktidningen's editor Adam Edström wrote the document. Swedsoft would like to express gratitude to everyone involved in the process.



Produktion: HeadQuarters Annonsbyrå Tryck: Lövex, Växjö, 2010