

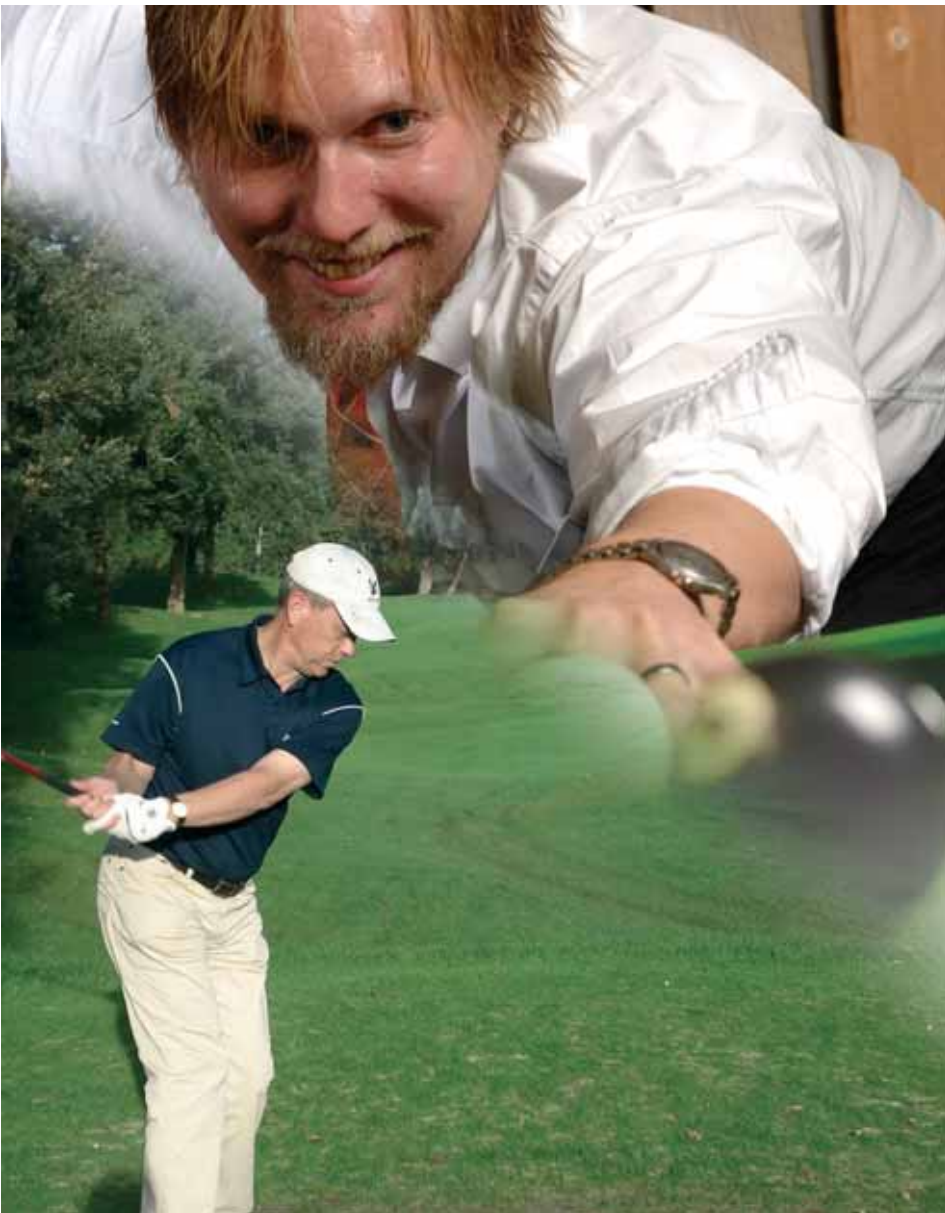


ITEA
INFORMATION TECHNOLOGY
FOR EUROPEAN ADVANCEMENT

Agile software development of embedded systems

AGILE

Newsletter #2/2006



PROJECT MANAGERS' GREETINGS

Few years ago, who would have believed that agile software development will penetrate to all industry sectors so rapidly. It is therefore no surprise that major IT corporations including Microsoft, SAP and others have announced publicly of trialling agile methods in their development settings. AGILE project has piloted different aspects of agile practices, techniques and methods in the embedded development settings. In line with large corporate adoption of agile methods, in this newsletter, Nokia Network reports the wide-spread results of their agile software development. Currently, it appears that the Scrum method is gaining the most attention in the embedded software domain. Scrum provides a simple to understand management framework for managing agile development.

Another clear indication of wide spread acceptance is the work undertaken by IEEE working group lead by Mr. Scott Duncan. Duncan's group aims at crafting a set of recommended practices for agile supplier and software acquirer. The work is planned to be concluded by the end of 2007. AGILE consortium has also taken part to the development of these guidelines and plans to trial the document's upcoming version.

This newsletter presents recent results from the AGILE project and cover interesting perspectives to agile software development of embedded systems. Two of the articles focus on testing issues and another two present novel views on business agility. F-Secure's localisation article presents an award winning solution on how to implement agile principles and development rhythm also in different organizational functions.

Agile greetings,
Pekka Abrahamsson, VTT
Ko Dooms, Philips

>> AGILE IN BRIEF

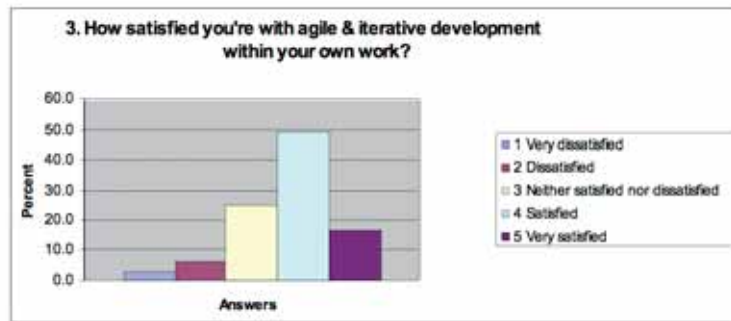
AGILE-ITEA project presents a new software development technology capable of producing a significant improvement for the competitive position of the embedded software industry in Europe. The project develops agile software development solutions for the embedded domain. The goal is to increase the reliability, productivity and reduce the risk of embedded software development. The solutions are validated by industrial trials in many different application domains.

Nokia Networks jumped into Agile wagon

Agile activities in Nokia Networks started about one year and a half ago. From a moderate start, the demand for agile and iteration development has expanded due to very good experiences in software development programs. Although programs are willing to apply agile methods, there are still issues that need to be understood and studied to gain even better results. In Nokia Networks we have a project called Flexible R&D project, which supports programs by adopting agile practices and disseminating agile development knowledge.

When a program shows willingness to use agile methods and starts agile activities, we ask if it can be called a Flexible R&D program. At the moment there are 15 Flexible R&D programs in Nokia Networks. The programs are quite different from each other; the environments vary from Java to the traditional code, and the amount of people can be from 10 to 150 persons. The work is done in one or several sites, even in different countries. Although the products have in many cases already passed their first releases, the programs are still willing to switch to an agile way of working.

Personal Satisfaction



Flexible R&D project was started in the beginning of 2005 with only a few members. As the interest in Agile and iterative development grows, the team has expanded and now we have about ten members. There are several ways for the Flexible R&D project team to support its programs. In addition to a general introduction to agile development, training is given on Scrum, TDD, A-TDD, CI and retrospectives. On request a Flexible R&D project member can facilitate the program's retrospectives, requirement workshops or iteration planning meetings, which often is educational when a program is taking its first steps towards agile. The Flexible R&D project also gives coaching e.g. in TDD. External consultants are used regularly in training, which brings a breath of fresh air from the agile development world outside Nokia.

As communication is very important we try to get people together. Flexible R&D project team has organized two Scrum Master Gatherings, which received quite positive feedback. People had a chance to meet, talk and share common interests and problems. There are communities and user groups created by subjects, e.g. CI, TDD or Scrum to get people with the same in-

terests together. Information about agile and iterative development is provided on Wiki pages, e.g. introduction material, blogs, information about agile books, upcoming agile sessions, etc.

Based on the Flexible R&D survey carried out last spring almost 70% of the people in organizations where agile methods are adopted are satisfied with the new way of working and did not want to go back to the old way. This encourages the Flexible R&D project to continue disseminating agile knowledge and supporting programs to adopt agile practices to get good quality products on to the market as fast as possible.

Total favourables & unfavourables



CONTENTS:

- Nokia networks jumped into Agile wagon.....2
- Experiences in combining professional testing and agile.3
- An experience of Agile Requirements and data management4
- Realizing Agility through Model Driven Architecture6
- Embedded testing - spare yourself the trouble7
- Automated testing of control system software8
- Business Modelling (BM)9
- Agility in Reconfigurable Manufacturing Systems' Control19
- The F-Secure Agile Localization Model11
- The use of Agile software development has an impact on a software development organization and its business on many levels11



EXPERIENCES IN COMBINING PROFESSIONAL TESTING AND AGILE

Maaret Pyhäjärvi, Senior Quality Engineer, F-Secure

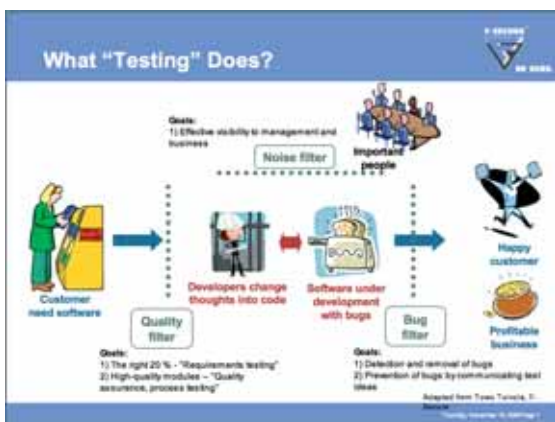
The past year has been a major source of learning, both for myself individually as well as for our development teams. A year ago we introduced two major changes into our development approach: changing product development practices towards agile development, and simultaneously introducing a product line to replace the separate products we had created until then. This article summarizes some of the lessons learnt, from the point of view of testing.

ROLE OF TESTING IN OUR DEVELOPMENT

Testing - or quality engineering, as it's referred to at F-Secure - has had a significant role in development. The ratio of Software Engineers to Quality Engineers is about 2:1, and ever since I joined F-Secure two years ago, I've been proud of the privilege of working with such a skilled and professional team. As we develop products, our testing has been driven-based with regard to the value it provides, focusing on risks. Resources are limited, and practices should fit our context to provide as much value for the effort invested as possible. This includes careful consideration of the value of test documentation, sharing testing work with developers, and close collaboration with business in decision-making.

THE CHANGES IN PRACTICE

Introducing agile development took place by introducing several Scrum-like practices: small teams with scrum masters, daily meetings and sprint



backlogs, product backlogs, and 30-day sprints. Whereas we previously had coordinated testing as

a major effort in the overall development, we now allocated Quality Engineers to teams to plan their work as part of the team. We did not move directly to Scrum, but introduced an adaptation of our own, a product development framework we refer to as F-Lex. In many ways the biggest change for us in introducing agile was the monthly planning and delivery, and changed team compositions.

The bigger changes result from the introduction of the product line. Having to shift our thinking to include all the products we would develop simultaneously has been a major challenge. Whereas previously we only had to think of one product and system to deliver and test, now we had to reuse results from one product to another, filling in the appropriate gaps in quality information without running exactly the same tests again. Combining this with the 30-day sprints and small teams resulted in many valuable lessons.

LEARNING COLLABORATION IN SMALL TEAMS

A change we have embraced and enjoyed the most is close collaboration within the small teams. I find that within F-Secure, the developers and testers were already working close to one another, but the introduction of agile teams has further enhanced this. Within the year, we've experimented with having all the testers in the development teams, and having some stay outside the teams as system testers. I've grown to prefer the first option, as it helps the teams grasp the system point of view and deliver more complete items, as the teams realize there is no one outside the teams to pick up on what falls outside the team scope.

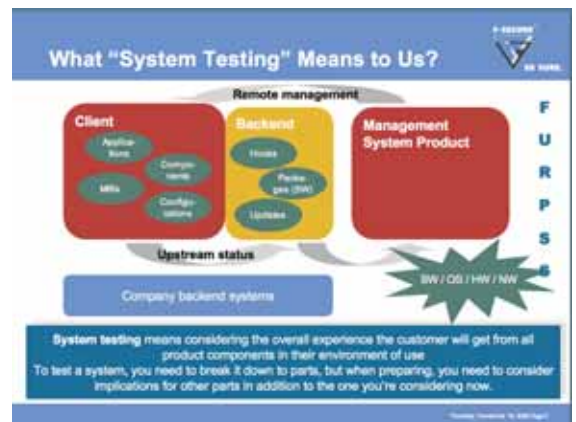
The major challenges in this area have been a great source of learning: integrity when working close to developers. Especially people with less experience seemed to easily agree to compromises and give up on important perspectives in testing if not supported. We found, however, that a better way of resolving this is to support the learning mindset for testers, rather than protecting from the pressure within the team.

Moving focus of testing investment in an agile way within sprints

When work is planned in the beginning of a sprint, we learned that this may cause unnecessary rigidity in how we invest the limited time we have for testing. There may be risks that would be more relevant to address than the ones that were identified in the beginning of the sprint, as testing is continuous learning. Especially if the changes in focus would mean overlaps between several teams, the strict team composition first made our testing less agile than we had been.

Role of test management

The role of test manager changed from overall test co-ordination to test investment co-ordination over the teams. There is no direct influence on what the teams would do as the teams are self-organized, so the focus of my



work as Test Manager has turned towards working with business owners on product backlog having items that are relevant deliveries and can be judged through testing.

Cross-team collaboration in testing

Avoiding overlapping areas in testing in small teams for the whole system requires working across the teams. We learned that it is easy to both miss relevant areas and duplicate work if mechanisms of collaborating on this are not included. Our approach has been to hold a meeting amongst team representatives with a testing focus on a weekly basis, and organize occasional planning workshops to agree on scopes. The flow of information from one team to another, as everyone needs to understand the types of changes introduced, still poses one of the biggest challenges.

LEARNING TO ORGANIZE FOR SYSTEM TESTING

The 30-day sprints and getting items completed and to delivery within that time is a target we are striving for. However, moving from a 3 month validation period that comprises a month of system testing and two months of beta, to having most of this happen within the sprints is not an easy task.

Main lessons for us in this include:

Stabilization sprints

While we work to avoid the need of stabilization sprints, we've noticed that there are major architectural changes that just cannot be done from the system and delivery points of view within a sprint. We've organized teams around stabilization targets, fixing known issues, even from previous product releases, allowing longer periods for looking at the system across the teams.

Longer learning period needed

People need time to come up with new ideas and to notice gaps. We've noticed quite often in testing a system that while the duration of the work would be a week's effort, the period needed to enable learning is longer. This also is quite dependent on the individuals.

Definition on the product backlog level

Instead of doing test plans, the focus has moved to analyzing whether the testing can be included in items we order and deliver on the product backlog level. We realised that the product backlog is not a replacement for the requirements document, but more for project and test plans that help us focus on continuous delivery.

Reporting on commitments and quality

At first we tried reporting our commitments to the product owner. However, in the size of our development effort, the big picture was not really communicated. Over the months we've learned to share information on commitments and status, although this is definitely still an area of continued progress for us. We've moved change control boards to the teams, and introduced a business-perspective change advisory board to facilitate learning on what would be important for us in quality.

Organizing continuous beta program

External deliveries to customers carry an overhead cost, and we needed to minimize the number of deliveries. In practice this meant

changing from per-product beta programs to a continuous shared beta program, into which we deliver on a monthly basis for on-time feedback of real-world compatibility issues relevant for our success, as well as real customer feedback.

LEARNING THE IMPORTANCE OF PLANNING

A major area of learning we've faced is about the role and practices of planning. Even with the focus on delivering ready features on a monthly basis, planning both what could be done within a month and what would be included on a particular product release has been an experience.

Planning is even more relevant, just somewhat different, and we've learned that:

Product backlogs are a collaborative effort

It seems we get them done best when it is shared work between the teams, overall project and test management and prioritized by the Product Manager. As the product backlogs should serve as our common understanding of what we would be delivering, we've tried different styles and levels of abstraction.

Planning ahead is included in the product owner role

It first we were really focused on learning how to deliver on a monthly basis, and looked less into the overall planning. It would seem, however, that the planning ahead in product backlog is included in the product owner role, that within our organization it is split into the Product Manager (priorities), Program Manager (resources), and Project Manager (communication). We learned that planning ahead is essential, even more so with agile where we want to adapt our plans.

Systems thinking and architectures are relevant

Understanding the system and its architecture is the basis on which our teams deliver. We find it as relevant as before, if not even more so with the agile product line development. Our experiences have been that the architecture could easily have a make-or-break impact on our development success.

CONCLUSION

Last year has been challenging as well as rewarding. We've learned a lot, and still have many challenges to tackle. It is however clear that we will continue on the chosen path towards agile.

AN EXPERIENCE REQUIREMENTS



Santiago Estela
santiago.estela@sqs.es

When facing the requirement engineering phase, each organization, even each team, has to overcome different challenges in order to come up with the best possible specification, since deficiencies in requirements are one of the biggest causes of failures in software projects. All these efforts, however, have a common weapon: INFORMATION. To have the most significant and up-to-date data is vital to success, especially when this data comes from within the same organization.

Agile methodologies have proven their worth in dealing with these issues. Their short-iteration strategy provides a good opportunity to access key data earlier and more easily.

Below is an outline of the process and results of our experience in the collection and usage of key data to increase efficiency and accuracy in the planning phase (i.e. requirements managing and evaluation)

Choosing the appropriate project for the experience: Project description

The features of the selected project (and company) were those that best fitted the objectives of the trial:

COMPANY	PROJECT
Medium-sized	Customer available
Hw and embedded sw development	Non-developed requirements
Inexperienced in Agile	Changes are likely to happen
Good relations with SQS	6 months
Eager to learn new methodologies	5 developers



About author

Maaret works as Senior Quality Engineer at F-Secure and participates in projects delivering security products on Windows workstations and servers. Her work includes test management and testing, as well as process and practice development within testing.



Realizing Agility through Model Driven Architecture

Modeling an application instead of coding it brings developers and customers to a higher level of abstraction. This makes it easier to make changes, to investigate alternatives, to derive variants of an existing model and to visualize changes. The actual implementation is then performed through model transformations and generators. This article describes the E2S tool supporting Model Driven Architecture and how it has been used within this ITEA project.

The goal of Model Driven Architecture (MDA) is to model an application in a platform independent way and transform it stepwise into executable code. The transformations and generations are performed automatically on user request. E2S has developed a tool supporting this MDA objective. Its structure is shown in:

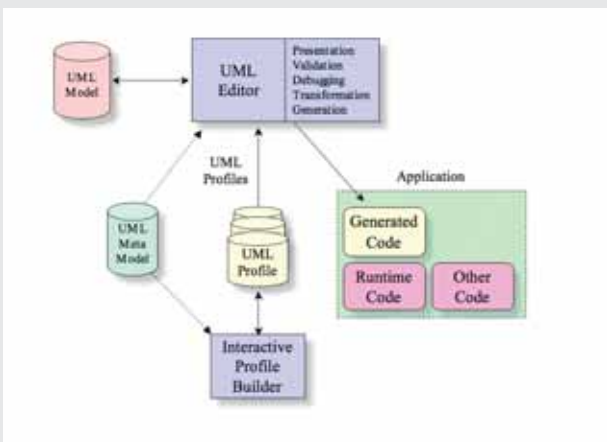


Figure : The structure of the MDA tool

The behavior of the UML editor is driven by a profile, containing a domain specific context (e.g. embedded systems):

- selection of the different model elements and diagrams that may be used within the domain;
- graphical representation of the model elements;
- definition of constraints on the modeling that apply for the domain;
- transformations and generators to produce code from a given model. The term code may refer to programming language code, but also to test scripts, documentation or model summaries.

The definition of a profile is the responsibility of the domain specialist. This is the person who is responsible for the (development) process definition and the quality of the development. Within the domain of embedded systems for Avionics, this is also the person who makes sure the development complies with the DO-178B standard. Note that this domain specialist is not involved in the actual application development.

The application developer starts the UML editor with a selected domain profile. From that moment on, the UML editor represents the model elements as defined in that profile, offers a validation function making sure the model conforms to the domain constraints and provides a list of available transformations and generators.

Depending on the generator, the generated code may be self-contained

or may require runtime code, requiring a less complex generator. Examples of runtime code are:

- Σ • object allocation and persistence modules;
- Σ • navigation primitives for the application objects;
- Σ • runtime support for OCL constraints.

Runtime code is (computer) language and domain dependent, but application independent. For embedded systems for Avionics, the use of runtime code means less code to certify.

In addition to the runtime code, an application may also require additional code, such as drivers.

With this environment set in place, the E2S tool encourages a user (i.e. an application developer) to model the application, investigate possible alternatives, create variants from an existing model and provide solutions for several platforms. gives an overview of how a user models an application, creates the code and tests using different generators. The generated application can then be tested using the generated tests.

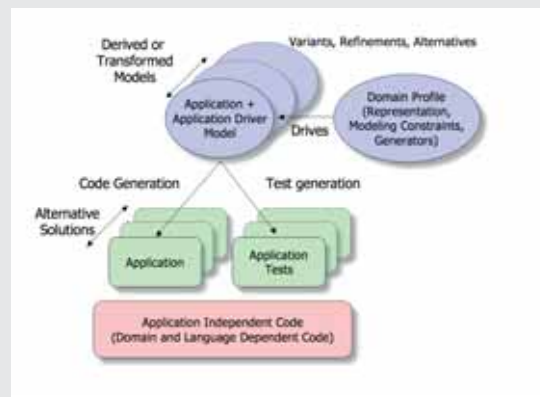


Figure: Modeling an application

There are several advantages to modeling an application rather than programming it:

- modeling is at a higher level than coding. It frees the developer from attending to low level details and decisions;
- Σ an important element of UML modeling is the use of several diagrams. This is an excellent means of communication with customers. Discussions at the modeling level in turn lead to better insights, both for the customer and the developers;
- modeling is reasoning about objects, relations, constraints, scenarios and views. It does not impose a sequential way of developing code. This is a more natural way of thinking about applications;
- the use of profiles makes sure an application stays within the limits of possible constraints imposed by a domain.

Within the domain of Avionics and the stringent coding standards and rules defined by DO-178B, externally defined generators are an absolute must for the certification of the applications. The E2S tool complies with this requirement as all transformations and code generators are defined by the domain specialist and are stored in the domain profile.

Embedded testing - spare yourself the trouble

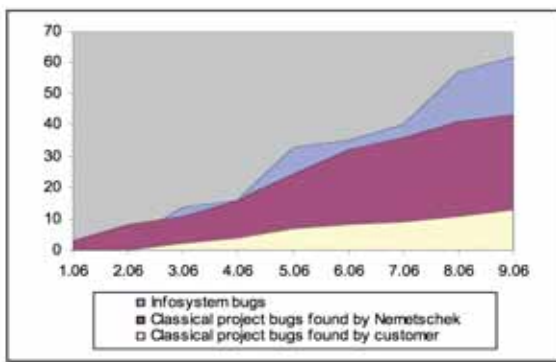
Everyone has heard of software testing – a subject addressed by every software development process out there. There are many ways to do it – black box, white box, automated, system, etc. Agile processes also provide practices addressing the need for testing – test driven development and acceptance tests (XP), automated regression testing (Crystal), inspection (FDD). But how effective the testing will be not always depends on implementing some of them.

From project to project, the techniques' effectiveness might vary depending on varying factors such as the business domain, schedule, availability of resources, technology dependencies, etc. When it comes to software development for embedded systems,

this is even more relevant since the software being produced might need some special environment.

The main goal of the AGILE project is to address the need for making software development of embedded devices more agile in all respects, including testing. Undoubtedly introducing some of the agile practices for testing would help in certain

cases. Nemetschek has tested a different approach in a project called "Infosystem" which included the creation of software for specific hardware devices. The approach consisted of the involvement of the customer in the testing process to the lowest level possible – we call this customer testing pattern. The need for thorough testing was a key requirement so, after evaluation of the available resources and the needed testing environment, it was agreed with



the customer that the most effective approach from the business point of view would be to let the customer perform the testing. Since the customer had all the needed re-

sources in terms of manpower and hardware environment, this made sense and agility in software development is exactly this – do the work the way it makes sense. The project was done in 2 month iterations and after each iteration the customer tested and generated bug reports.

What was the result? The diagram shows the

overall number of bugs found in the "Infosystem" project compared to the number of bugs found during testing in another project carried out in a more "classical" way – Nemetschek did its part of testing and the customer also found bugs afterwards. In the "Infosystem" project we got a lot of error reports from our customer. In a normal project this would spell disaster. But since the contractual agreement was that the customer will do the testing and report bugs, the project not only succeeded, but the client was happy to be involved and see the progress from the inside – another big win for the agile software development.

The moral of the story – agility can be applied in many ways in embedded software development, so one should carefully assess the customer and the project needs and not be afraid to go for what makes sense and involve the client to do his share of the work to secure a quality product.



Boyan Angelov
bangelov@nemetschek.bg

...Realizing Agility through Model Driven Architecture

All our Belgian partners have used the E2S tool:

Σ K.U. Leuven has evaluated several versions of it and defined a small Java generator. In addition, K.U. Leuven has compared it with other UML modelers, studied concepts such as runtime constraints and suggested solutions for several problem areas.

Σ Barco has used the E2S tool to define a profile for Avionics. This profile contains an Ada code generator and the conceptual scheme to test a

generator. Barco also used the E2S tool to model a template application for an embedded system in Avionics.

Σ E2S has replaced its original UML modeler and integrated Pascal generator with this version and has profiled a new Pascal generator. It is now using this tool for all its custom software projects.

The current version of the E2S tool will be available as a product at the beginning of next year.

However, E2S will continue to use it for its own custom software projects, propose it in ESA co-operation projects, use it as a basis for further research with K.U.Leuven and is also interested in using it as a full demonstrator for Avionics projects at Barco.



Michel Huybrechts
mhu@e2s.be
<http://www.e2s.be>



Stijn Rammeloo
stijin.rammeloo@barco.com



Stefan Van Baelen
stefan.vanbaelen@cs.kuleuven.be

Automated testing of control system software



Marcelino Novo
mnovo@fagorautomation.es



CONTROL-SYSTEMS RELATED PROBLEMS

Industrial automation and control systems have some characteristics that make the software developing process more complicated than that of other applications:

- They must be intrinsically safe, capable of responding immediately and meeting real time requirements (mission critical systems). The quality requirements are very high; safety certificates must be provided and they have to meet specific standards such as EN-954, IEC-61508-3, etc.
- Their resources are usually limited (little memory and just enough CPU power) for cost reasons. Open control systems have replaced the closed systems with proprietary hardware and software.
- Distributed systems are commonly used consisting of several elements connected through a network. As a result, information technologies (connectivity) become more and more important.
- Sometimes, there is no interaction with the user and, consequently, they must run without errors for a long time (10 to 20 years) and be capable of self-recovery when errors occur. Other times, they may interact with the user through a very simple display, with just a few keys or even through a full user interface, based on windows similar to those of a PC.
- Although the systems are increasingly complex, they must still be flexible, configurable and capable of incorporating new functionalities. On the other hand, their lifecycles grow shorter and shorter making it crucial to spend as little time as possible developing the software.

NEED FOR AUTOMATED TESTING SYSTEMS

In view of this situation, Fagor Automation decided to target the following goals:

- to shorten software version development time from 9 months to 3 months;
- to bring the validation efforts down to 20%;
- to improve the quality of the software by reducing the number of errors to a half.

The new light and agile methodologies were meant to meet these needs as they all have the goals of customer satisfaction, adaptability to changes, fast and right-to-the-budget product delivery.

However, using certain agile practices such as continuous integration or incremental development resulted in a considerable bottleneck when validating the software once it is developed. After analyzing the problem, we decided to develop an automated testing system because of the following factors:

- products with a lot of configuration possibilities;
- different software versions for different customers;
- limited and very expensive test resources;
- a limited number of technicians and working hours;
- results dependent on technicians' skills;
- difficulties to repeat certain failure situations.

SYSTEM DESCRIPTION

In the first stage, we evaluated several off-the-shelf tools. The ones available at that time, besides being very expensive, turned out to be more inflexible than we wished and, therefore, we decided to develop our own tailored system.

We then did a logic classification of the many existing test cases and defined the means (stand-alone product, mock-ups and real machines) to be used in each case.

Finally, we implemented the automatic test environment with two basic guidelines in mind:

- it should emulate at all times the operation mode of the end-user of the control system;
- the user interface and the language used to describe the test cases should be as simple as possible and oriented to non-computer-knowledgeable people.

As can be seen in the diagram, the resulting control system is connected to a PC through Ethernet. The heart of the system is an interpreter of scripts written in Visual Basic. The developed HMI lets the

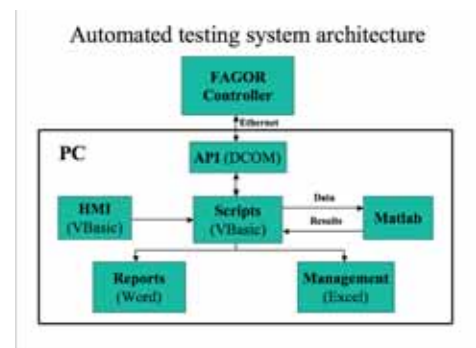
user choose the test cases they wish to execute among all the ones stored in the database. The engine, through a DCOM-based API, is in charge of giving the necessary commands to the control system and gathering the results. In some cases, it also communicates with the calculation and simulation application Matlab. Finally, it generates reports on the results obtained and the management data such as the time required, the percentage of coverage of the test cases, etc.

CONCLUSIONS

With this system, a lot of effort is required in the early stages, but it pays off later on by bringing the following benefits:

- a drastic reduction of validation time;
- increase in effective work hours (nights and weekends);
- more verified cases;
- the expensive testing brings pay off;
- greater repeatability of failure situations;
- the results do not depend on the technician running the test;
- the technicians are not tied up with repetitive tasks.

In short, it is an essential tool for reducing the time it takes to launch software versions using agile methods.



BUSINESS MODELLING (BM)

Nowadays, organizations that develop embedded software cannot be wedded to any particular market segment, and have to serve as brokers and middlemen with expertise not just about specific application domains but the connectivity elements and middleware needed to make a system work in a specific, but dynamically changing environment. Most importantly, the companies have to be fast on their feet, keeping track of changes in markets, standards and technologies and ready to adjust their mix of tools and building blocks and their expertise as needed. Therefore such companies need ways of rapidly adapting their business models to the changing business and market scenes.

A new business model is emerging, a model where "most key missions of the organization are distributed to the myriad individual pieces and unity comes from the vigor of people and the free flow of knowledge, not a burdensome central headquarters."

Today's most successful executives while still greatly concerned with cost structure, maximizing operational effectiveness, and business process reengineering - have shifted their focus to issues of how to build for faster, how to attract and retain the best people, how to develop at all levels in the company, how to effectively, how to become a true , and how to be more effective global corporations.

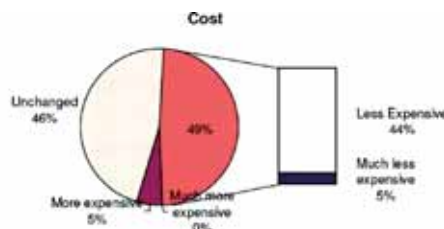
AGILE PRACTICES AND BUSINESS MODELS

An agile enterprise reacts quickly and efficiently to changes, such as market opportunities or mergers and acquisitions. The ability to react quickly translates to top-line revenue, and the ability to adjust efficiently equals lower costs.

There are different lists of reasons why agile methods should be practiced in the field of embedded software development and what the business benefits are from applying them. There are in any case certain core benefits which include:

- Improved return on investment (ROI)
- Early detection and cancellation of failing products
- Reduced delivery schedules
- Higher quality software
- Improved control of a project
- Reduced dependence on individuals and increased flexibility

Can Agile processes alter the cost of development? One of the main concerns for a company is the cost of development. As a result of a web-based survey run by Shine Technologies between November 2002 and January 2003 , we can conclude that this is the fundamental motive for using agile methods.



Across the respondents with an average knowledge or better, 48.6 % believed that development costs were reduced. Including the responses that indicated that costs were unchanged, a whopping 95% believe Agile processes have either no effect or a cost reduction effect.

The new era for BM is driven by the advent of more advanced approaches, methodologies, infrastructures and platforms and a need for families of solutions that allow predictable customisation.



Teodora Bozheva,
European Software Institute
teodora.bozheva@esi.es



Maria Elisa Gallo
European Software Institute
marialisa.gallo@esi.es

References

- [1] <http://digitalenterprise.org/models/models.html>
- [2] http://en.wikipedia.org/wiki/Business_model
- [3] <http://www.embedded.com/shared/printableArticle.jhtml?articleID=9900929>
- [4] Bruce A.Pasternack and Albert. J. Viscio. "The Centerless Corporation" (1998)
- [5] First version of state of the art in enterprise modeling techniques and technologies to support enterprise interoperability, Deliverable D.A1.1.1, ATHENA project (FP6-IP-507849)
- [6] Johna Till Johnson, Enabling the agile enterprise, Network World, 11/08/04 (<http://www.networkworld.com/cgi-bin/mailto/x.cgi>)
- [7] Scott W. Ambler: Agile Modelling, John Wiley&Sons, Inc. (2002)
- [8] Beck K: Extreme Programming Explained: Embrace Change, Addison-Wesley (2000)
- [9] Poppendieck M., Poppendieck T.: Addison-Wesley (2003)
- [10] Highsmith J.A: Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, Dorset House Publishing (2000)
- [11] Schwaber K., Beedle M.: Agile Software development with Scrum, Prentice Hall (2002)
- [12] <http://www.dsdm.org>
- [13] Palmer S., Felsing J., A Practical Guide to Feature-Driven Development, Prentice Hall (2002)
- [14] www.agilemanifesto.org/
- [15] P. Abrahamsson et al., Agile Software Development Methods Review and Analysis, VTT Publications 478, 2002
- [16] D. Cohen et al, Agile Software Development, A DACS State-of-the-art Report, 2003
- [17] B. Boehm, R. Turner, Rebalancing your Organization's Agility and Discipline, IEEE Computer, June 2003
- [18] Why Use Agile Methods? Steve Hayes, ZDNet Australia, 7July 2003, <http://uk.builder.com/manage/project/0,39026588,20275975,00.htm>
- [19] eXPERT project (www.esi.es), Nemetschek's case study
- [20] Agile Methodologies Survey Results, Shine Technologies, 2003, www.ShineTech.com

Agility in Reconfigurable Manufacturing Systems' Control



Authors:
Giovanni Aiello (giovanni.aiello@eng.it)
Marco Alessi (marco.alessi@eng.it)



Research in software engineering cites Agile methodologies as innovative processes able to carry out a logarithmic trend of the requirements change cost, according to the project duration. Moreover, recently, the embedded software market also seems to be interested in Agile methodologies, for example, the industrial automation area uses the term "Agile" to express the flexibility and reconfigurability of manufacturing systems. For this reason, Agile Manufacturing aims to achieve flexibility and responsiveness related to market needs.

Engisud demonstrates the application of Agile methodologies to control Reconfigurable Manufacturing Systems (RMSs) through Programmable Logic Controllers (PLCs) and proposes AGICOSD (AGILE Control System Development) to perform Agile manufacturing and Agile PLC-based control system development.

AGICOSD includes a development process, guidelines and tools for generating Ladder Logic and an Instruction List code for PLCs starting from a java code built by means of Agile best practices and Design By Contract (DBC). AGICOSD assumes rigorous tests during the development of the java code through Test Driven Development, white-box testing, black-box testing, regression testing and integration testing. Furthermore, the reliability after the deployment is provided by runtime testing of the java code based on contracts among entity, control and boundary classes. Entity classes represent real machines (e.g. Vertical Milling Machine), Control classes handle entity classes and Boundary classes are the interfaces among components. Contracts are expressed in Java Modeling Language (JML) and that the control system can be modelled and the runtime testing can ensure reliable deployment towards the embedded domain through code instrumentation and tests generation using DBC.

AGICOSD development process is shown in Figure 1.

The guidelines consist of mappings performed on three levels of abstraction, the: Organizational Block Layer, Component Layer and Class Layer.

The Organizational Block (OB) is Step 7 in the

OB layer mapping, while Step 7 functions (or sub-programs) concern both the Component layer and

JMLEclipse for editing and validating JML contracts, and ILGenerator, provided by Engisud, for JUnit

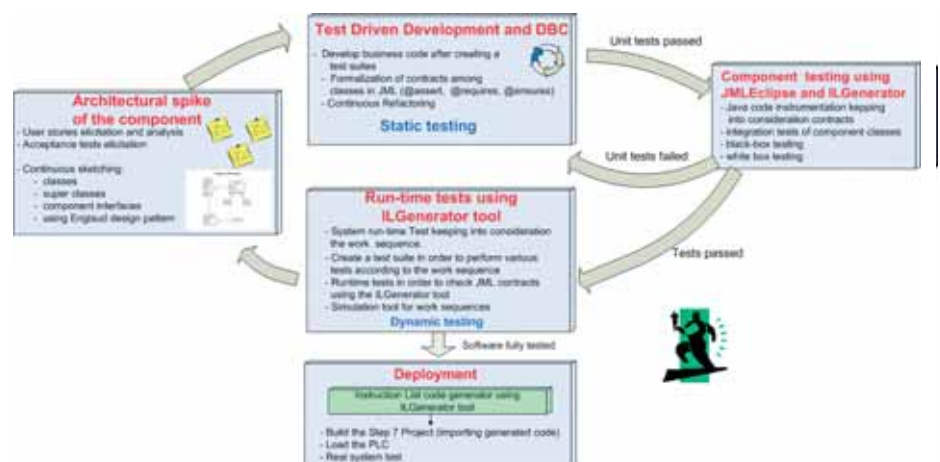


Figure 1.

the Class layer.

During the class layer mapping, all entity classes of a component are mapped in Step 7 functions, e.g. `goUp()` and `goBottom()`, each of which is mapped to a Step 7 segment performing a single work unit.

The component layer concerns the mapping between each software component and a Step 7 function; JML contracts can be translated into switch configurations. An example is shown in Figure 2.

AGICOSD also includes two eclipse plug-ins:

tests generation based on contracts, reusable components repository management and Instruction List code generation after the run-time testing, ensuring a reliable deployment towards PLC. Trials performed with the collaboration of the DTMPiG (Department of Mechanical Technology, Production and Managerial Engineering) of the University of Palermo have underlined the benefits brought by Agility: namely, agile re-configuration of RMSs, scalability, reusability of components and reliability in the deployment phase.

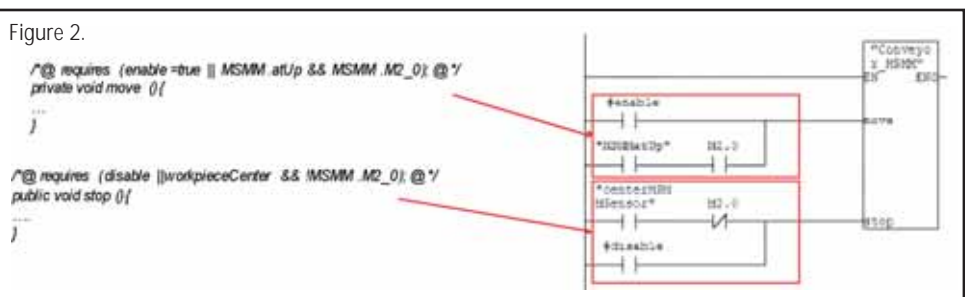


Figure 2.

The F-Secure Agile Localization Model



Mika Pehkonen
Manager, Documentation
and Localization
mika.pehkonen@f-secure.com

Our experience after a year of piloting an agile approach for localization has been positive and productive. The ROI for the process was met and exceeded within the year. The response from presenting the process to the localization industry and translator community has been encouraging. The process also yielded an "Innovator of the Year" award from Client Side News.

Localization is traditionally a waterfall process. Even in agile projects, we usually start localizing once we feel comfortable with the level of completeness in the mother project, and then use a waterfall translation-validation-fixing cycle. Starting localization late leaves little or no time for localization input due to time and resource constraints.

For successful incremental localization, we have



found the following two factors helpful. 1) Platform thinking - multiple products from a single source. 2) Using XML with unique string identifiers.

Our internal localization process is very translation memory (TM) dependent. For this purpose we have selected an open standard, TMX, for translation memory management. Development concentrates on the English master. Localization interfaces automatically with the version control, going through all files (not just project-wise), recreating localized versions on a weekly basis by using full string equivalents and contextual information of strings. This allows development freedom to change the code content of source files without needing to consider localized files as branches.

Translators receive an automatically generated Multilizer kit, in which they translate new material and review changed strings. We also send periodical screenshot kits, which allow the translators to validate and change the translations in context and fix resizing bugs by editing. This approach has decreased the development effort in fixing what are minor, but common defects.

Translation purchasing is based on bandwidth, not words, which is the traditional method. Edits and resizing work by the word are not productive for the translator. Since the model is dependent on the translator as cultural expert and creative writer, we want to foster community feeling and personal involvement.

Translation tasks are defined with a miniature backlog. If the time allocation proves insufficient to complete all work, the kit is still returned and incomplete tasks remain in the backlog. Once the kit is returned, the translations and edits go back to the system and new language versions of files are generated and fed into the version control. TMX files can be generated after each 'handback' and be updated into our termbank for reuse by parallel development projects.



No consultants or large software investments were needed in implementing the process, but we want to thank our tools supplier, Multilizer, for their help.

The future for agile localization and documentation in our organization will focus on using incrementally created and localized documentation and software as a feedback channel from day one of development. This will allow us to tap into the feedback potential of customers with less than perfect English skills not only in documentation and localization, but also in the agile product development.

On exploring the business impact of agile development

THE USE OF AGILE SOFTWARE DEVELOPMENT HAS AN IMPACT ON A SOFTWARE DEVELOPMENT ORGANIZATION AND ITS BUSINESS ON MANY LEVELS.



The Four Cycles of Control framework (Rautiainen 2004) and the IT Innovation model (Lyytinen and Rose 2005) both work as a tool for connecting software development and an organization, and analyzing the impacts. The Four Cycles of Control framework consists of four control cycles called Strategic Release Management, Release

Project, Iteration and Heartbeat. The Strategic Release Management is an interface between software development and strategic decision making. The IT Innovation model deals with IT innovations, exploration and exploitation. IT innovations mean different types of innovations which are used in an organization in exploration and exploitation. Exploration deals with finding new opportunities through risk taking and experimentation. Exploitation on the other hand is about using old practices through efficiency and selection. The purpose is to optimize different process features, including innovative content, speed, cost, quality and risk, during exploration and exploitation. It is difficult to optimize them all at the same time, so organizations might have to trade, for instance, agility against other criteria. The IT innovation model suggests that if agility is emphasized in an organization's development process the quality and innovative content of a product decrease. In addition agility increases the risks related to a development process, and it also increases the costs related to the development process according to the model.

A case study was performed where the impact of agile methods on an organization

was investigated. The results show that the use of agile methods in the development process has vast impacts on an organization. In the case organization the agile projects are well connected to the higher organizational levels. The decisions made on the management level affect the projects but on the other hand, the management has to consider different issues when the projects are agile. For example, the management needs to support agility. Agility affects all stakeholders: the management, the project management, the development teams that are using the agile software development methods, other teams which are not agile, and different departments of the organization e.g. marketing in the case organization. The main consequences are the demand for communication and collaboration. Concerning the IT innovation model, the case organization has passed the exploration phase and is currently moving towards the exploitation phase. The process features - namely, innovative content, speed, cost, quality and risk - which can be affected during these phases were also affected in the case organization, but the effects varied from the effects in the IT innovation model. A salient outcome was if the speed of development is emphasised it does not necessarily have a negative effect on the other process features: quality, costs, innovativeness, and risks. This finding is important as it is in contradiction with the IT innovation model. Indeed, in the case organization, it appeared that the quality of the product and its innovative content was increased due to agility. Moreover, the costs of the development process decreased. However, the impact of agility on risks was more difficult to evaluate; agility does eliminate some risks but on the other hand it can bring other risks, such as miscommunication in the development team.

Kaisa Komulainen
kaisa.komulainen@vtt.fi





AGILE

AGILE - Agile development of embedded systems

21 partners, 8 countries, 171 person years

1.4.2004 - 31.12.2006

<http://www.agile-itea.org>

Project coordinator: Pekka Abrahamsson, VTT

(pekka.abrahamsson@vtt.fi)