SEMAT – Expectations from a Heterogeneous Software Development Organization

Martin Naedele
Industrial Software Systems
ABB Corporate Research
Baden-Dättwil, Switzerland
martin.naedele@ch.abb.com

Brian Robinson
Industrial Software Systems
ABB Corporate Research
Raleigh, NC, USA
brian.p.robinson@us.abb.com

***Abstract*— This position statement describes the situation and constraints of software development in ABB as an example for a large corporation producing software-based products, our experiences with software process improvement in the past, our current approach based on modular software engineering practices, and our resulting expectations on the SEMAT initiative.**

I. Situation and Challenges

Many advances have been made in software engineering in recent years. These advances have led to the creation of a large number of new software engineering techniques that address specific problems encountered in software development. Unfortunately, many of these problems still exist in industry, due to a low industrial adoption rate of new techniques. A key issue in industrial adoption involves determining the overall benefit that a technique will have and under what conditions and restrictions a new technique can be applied [4], [6]. Software development organizations, especially those where software is only a support technology, have limited time and money available for determining which of the available techniques will have a positive benefit. Therefore, industrial companies have to have confidence in the results shown by new techniques.

A. Company Background

ABB is a large multinational company with globally distributed business and software development activities. ABB products include components of critical infrastructures, such as industrial control systems and power grid management systems.

B. Characteristics of SW Development in ABB

Due to the wide spectrum of automation products that ABB supplies, our software development organizations are extremely heterogeneous. The development organizations in ABB range in size from three person development teams, for small embedded products, to organizations with several hundred software developers. ABB products range from small performance-constrained embedded devices to large, wide-area distributed systems. These products and systems include soft and hard real-time systems, as well as products developed towards safety standards, such as IEC61508. Most industrial customers want products that will have a multi-decade lifetime with as little change as possible, due to the inherent risk of injecting defects when upgrading software and the cost incurred when a prodcution system is stopped. On the other hand ABB also offers applications where customers demand frequent updates with new features.

With an increasing demand for vertical and horizontal integration (such as remote service), the traditional separation between software development for products delivered to a customer and for internal information systems and business IT of ABB as vendor is disappearing rapidly, therefore an alignment of software development approaches between the product R&D and business IT organizations is desirable.

C. Challenge

In order to increase efficiency and reduce costs, ABB aims at harmonizing its software development activities, which have developed based on the heterogeneity of the different development groups, as described above, as well as a corporate history characterized by mergers and acquisitions. By harmonizing tools and processes, ABB intends to reduce training effort, facilitate the moving of development staff between different organizations, reduce tooling costs, and continuously improve development efficiency and quality through the use of best practices.

II. History

ABB has started dedicated software engineering improvement efforts more than a decade ago with the creation of the ABB Software Process Improvement (ASPI) initiative, modeled around CMMI and driven by ABB Corporate Research. While this initiative did lead to increases in process maturity and measurable improvements in quality in ABB [2], the process-oriented approach had a large overhead and focused on what to do, not how to do it. The top-down nature of most global process improvement initiatives caused mismatches with the needs of the individual development organizations and thus limited their buy-in.

III. Current Approach

Recognizing the opportunities of further improvements in ABB's software engineering capability and, based on the learnings and experiences of the earlier initiative, ABB developed a different approach two years ago. This approach focuses on *principles* and *practices* instead of process.

A. *Software Engineering: a Set of Practices and Principles*

Process-based improvement initiatives, such as CMMI, tend to focus heavily on *what* must be done, *when* it should be done, and *who* should do it. What it usually fails to address is *how* it should be done. Yet, for most software development organizations, the details of how to do it best are what are needed for real improvement. For example, look at code reviews. It is a very simple concept, yet many people perform them poorly and adoption of this technique in industry is quite low. At ABB, we ran a set of workshops that included *why* we should perform code reviews and *how* to perform them effectively. The feedback from these workshops showed that many developers were not using code reviews to look for low level defects, but rather only to debate functionality or design choices among the development team.

Instead of focusing on heavyweight processes, our new approach involves identifying simple key *principles* that describe *what* must be done in our development projects to be successful. The majority of the focus goes into developing a set of *practices* that detail *how* these principles can be met by development teams around ABB. These practices represent a number of possible techniques that have been used successfully in other industries or projects. In this way, we do not prescribe a "one size fits all" solution, but rather identify many effective techniques and tools that could be used, and work with the development teams to find the ones that fit the product and people best.

At the top level, areas of software engineering, such as requirements or testing, are listed as *process areas* in the ABB Software Engineering Framework, shown in Figure 1. Each of these process areas is then broken down into a small set of key *Principles* that describe what must be done in these areas. These principles are single sentence statements that capture the essence of what must be done, as opposed to heavyweight process descriptions. An example principle for requirements engineering involves "Capturing, documenting, and prioritizing customer, market, and technical needs." This single principle covers identifying and documenting needs, as well as prioritization, in one statement. Finally, each *principle* has a set of *practices* that can be used to address it. For example, Gemba Visits and Kano Modeling can be used to meet the requirements principle stated above.

B. *Structure*

A core team of representatives from all our development organizations, supported by full-time staff from the CTO office, as well as experts from Corporate Research, is responsible for:

- Breaking down the software development lifecycle into process independent, problem-oriented pieces, which we call "process areas".
- Producing guidance for *practices*, including methods, tools, and metrics that may be applied in each process area.
- Selecting and deploying a common set of development tools.

Based on this set of well documented and piloted methods and tools for the different process areas, each business unit selects those suitable for its own situation and defines priorities of introduction. The representatives in the central core team are the same persons that are also responsible for, and spend most of their time in, driving implementation in their units and achieving measurable results. This ensures that the cross-unit work in the core team is always business relevant.

Thus, today the individual development organizations drive the improvements themselves based on their needs, while the software engineering researchers in ABB Corporate Research with expertise in, for example, testing [6][8][11], defect prediction [5][10], requirements engineering [7], safety [2], and security [1], develop modular guidance on methods and tools, based on external state-of-the-art, their own research, and experiments and pilots in our development organizations.

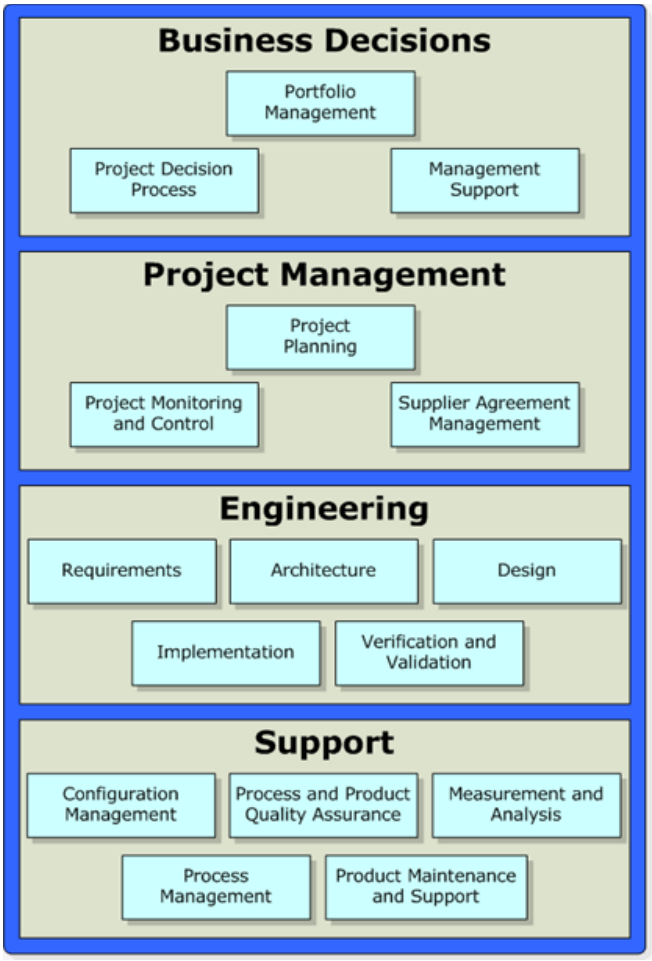

Figure 1. ABB Software Engineering Framework

C. *Status of Implementation*

At this point, we have made considerable progress in our *principles* and *practices* approach. We have guidance on techniques, tools, and metrics, as well as training materials for initial process areas, including requirements engineering,

verification/validation, and project planning. Deployment of the *principles* and *practices* in our development units is ongoing and some development organizations can already show sustained measurable improvements over multiple releases.

The organizations, and the developers themselves, have been very positive about the initiative. In fact, the current demand for the practices and principles for process areas not yet started exceeds our expectations.

In addition to the practices and principles, a common global tool environment supporting the development lifecycle has been selected and deployed and is being introduced to development units over time. The selected tools are simple and extensible, allowing for small and large development units to get benefit. Currently, requests from additional units, especially smaller units that historically did not have these kinds of tools, is also exceeding our initial expectations.

Category	Requirements Engineering Method Name		L	E	Page
	Data Modeling	Data Structure Modeling	A	M	74
		Domain Dictionary Development	B	L	76
Requirements Clarification	Linguistic Techniques for Disambiguation		A	M	77
	Stakeholder Reviews and Inspections	(also Validation)	B	M	90
	Customer Voice Analysis	(see Elicitation)	B	LM	40
	Requirements Abstraction Model (RAM)		A	M	?
Requirements Negotiation	Win-Win	(also Elicitation)	A	M	?
Requirements Prioritization	Priority Assignment Techniques		B	L	80
	Nominal Ranking by Importance and Urgency		B	L	83
	Volere Prioritization Method		B	L	84
	Maximum Value Table Analysis		A	M	-
	Value/Cost/Risk Prioritization		A	M	85
	Kano Model Classification	(also Elicitation)	B	M	28
	Analytic Hierarchy Process (AHP)		A	M	86
	Blitz QFD	(also Elicitation and Clarification)	A	H	-
Non-Functional Requirement (NFR) Analysis	Scenario-Based Analysis of NFRs		A	M	88
	Detection and Classification of NFRs		A	M	-
Requirements Validation					
Validation Reviews	Stakeholder Reviews and Inspections	(also Analysis/Clarification)	B	L-M	90
	Reference Team Reviews		B	L	92
	Checklist-Based Reading Techniques		B	L-M	93
Analytical Validation	Quality Analyzer for Requirements Specifications (QuARS)		A	H	-
	Low Fidelity Prototyping	(also Elicitation and Analysis)	B	L-M	95
	High Fidelity Prototyping	(also Analysis)	A	M-H	97
	Model Validation (includes requirements model animation)		A	H	-
	Acceptance Test Planning		A	M-H	99
	Requirements Traceability Analysis	(also Management)	B	M	101
Requirements Management					
Change Management	Change Control Board (CCB)		B	L	104
	Requirements Version Control		B	LM	106
	Requirements Status Tracking		B	L	107
	Change Management Models		A	M	?
	Release Planning / Product Line Planning		A	H	-
Change	Impact Analysis (multiple methods)		B	L-H	111
	Requirement Traceability Analysis	(also Validation)	B	MH	101

Figure 2. Example Requirements Practices

IV. EXPECTATIONS TOWARDS SEMAT

We feel that the spirit of SEMAT is very much aligned with the approach ABB has been taking over the past two years towards organizing and driving our software engineering capability. We expect that SEMAT activities will be complementary to our own efforts to produce a set of software engineering practices and principles for usage in industrial settings.

A key need to our approach is for better empirical evaluations of techniques. While most evaluations in the research domain involve showing how technique A is superior in all ways to technique B, a more pressing need is to identify the specific criteria and circumstances that each technique works best in. If we as a community can identify instances where methods work well, and where they work poorly, then adoption of these techniques can improve and they can result in real benefit for the community and the industry.

We hope that we can use SEMAT work products to refine and augment our own collection of practices, as well as the guidance for and against using those practices in certain situation and context. We also expect that we can provide relevant input to SEMAT based on the work we have already done, and can provide feedback from experimental evaluation within our R&D organization.

REFERENCES

[1] Brändle, M and Naedele, M. Security for process control systems an overview from the vendor perspective. *IEEE Security & Privacy*, vol. 6, no. 6, pp. 24–29, 2008.

[2] Dagnino, A., Cordes, A. Coordinating Process Improvement in Multiple Geographically Dispersed Development Organizations Using CMMI. CMMI Technology Conference and User Group. 2004. http://www.sei.cmu.edu/library/abstracts/presentations/Dagnino-Cordes-2004.cfm

[3] Hu, Z. and Bilich, C. Experience with establishment of reusable and certifiable safety lifecycle model within abb. In *28th International Conference on Computer Safety, Reliability and Security* (Safecomp 2009), September 2009.

[4] Kitchenham, B., Linkman, S. and Law, D. DESMET: a methodology for evaluating software engineering methods and tools, *IEEE Computing & Control Engineering Journal*, June 1997, pp. 120-126.

[5] Li, P., Herbsleb, J., Shaw, M., and Robinson, B. Experiences and Results from Initiating Field Defect Prediction and Product Test Prioritization Efforts at ABB Inc. In the Proceedings of the International Conference on Software Engineering (ICSE), May 2006.

[6] Robinson, B. and Francis, P. A Defect-Driven Process for Software Quality Improvement. In the proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM), September 2008.

[7] Robinson, B. Ho, C-W. and Williams, L. Examining the relationships between performance requirements and ”not a problem” defect reports, in *International Conference on Requirements Engineering* (RE 2008), September 2008.

[8] Robinson, B. and White L. On the Testing of User-Configurable Software Systems Using Firewalls. *Journal of Software Testing, Verification and Reliability* (JSTVR), to appear 2010.

[9] Shull, F., Carver, J., and Travassos, G. 2001. An Empirical Methodology for Introducing Software Processes. In *Proceedings of the 9th ACM SIGSOFT International. Symposium on Foundations of Software Engineering* (September 10 - 14, 2001). ACM, New York, NY, 288-296.

[10] Snipes, W., Robinson B., and Brooks, P. Approximating Deployment Metrics to Predict Field Defects and Plan Corrective Maintenance Activities. In *International Symposium on Software Reliability Engineering* (ISSRE), November 2009.

[11] Zheng, J., Robinson B., Williams, L, and Smiley, K.. Applying Regression Test Selection for COTS-based Applications. In the Proceedings of the International Conference on Software Engineering (ICSE), May 2006.