7-2009

Building Process Improvement Business Cases Using Bayesian Belief Networks and Monte Carlo Simulation

Ben Linders

Software Engineering Institute

Building Process Improvement Business Cases Using Bayesian Belief Networks and Monte Carlo Simulation

Ben Linders

July 2009

TECHNICAL NOTE
CMU/SEI-2009-TN-017

Software Engineering Measurement and Analysis

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

Table of Contents

List of Figures

Abstract

Many organizations require quality improvement initiatives to be based on quantified business cases. This leads some organizations to start measurement programs to collect data about current performance—a lengthy and expensive process that requires a strong commitment from management. This report describes a collaboration between the Software Engineering Institute and Ericsson Research and Development, The Netherlands, to build a business case using high maturity measurement approaches that require limited measurement effort.

For this project, a Bayesian belief network (BBN) and Monte Carlo simulation were combined to build a business case for quality improvement. Using a BBN provided quick insight into potential areas of improvement based on relevant quality factors and the current performance level of the organization. Monte Carlo simulation enabled a detailed calculation of the likely business results in the areas of potential improvement. This approach led to a decision to implement agile methods to improve the quality of requirements.

1 Introduction

Many organizations require quality improvement initiatives to be based on quantified business cases, which leads some organizations to start measurement programs to collect data about current performance. But creating and implementing a measurement program can be a lengthy and expensive step that requires a strong commitment and plenty of patience from management. Many organizations end up killing their improvement programs before any benefits are obtained. Thus, a better approach is needed that would allow organizations to build a credible business case with limited but sufficient measurement effort.

The Software Engineering Institute (SEI) and Ericsson Research and Development, The Netherlands, conducted a collaborative investigation of high maturity measurement approaches that could be used to create compelling business cases for process improvement. The analytical techniques described in this report are practical but leading-edge and show promise for quickly building business cases for improvement initiatives based on minimal data.

This technical note describes a two-step approach combining subjective expert opinions and objective process and product measurements. The first step was to use a Bayesian belief network to identify the areas with the most potential for improvement. Then in the second step, data was gathered for only the promising alternatives and further quantified using Monte Carlo simulation. The pilot described in this paper used this two-step approach within a Define, Measure, Analyze, Improve, and Control (DMAIC) framework to develop a business case for deciding among various approaches for improving quality.

2 Building Business Cases for Improving Quality

Defects are commonly used as an indication of quality. A high number of defects affect cost and schedule; therefore, reducing the number of defects can bring significant benefits. But although everyone agrees it is better to prevent defects or detect them earlier in software development than to let end users find them, many companies still struggle with changing their process from "testing quality in" to achieving quality through design and inspection. One of the main reasons is that they have difficulty defining a business case for a quality improvement program.

2.1 Problems Building Business Cases

A business case for improving quality has to describe the areas with quality problems and suggest potential solutions to solve them. It also needs to include quantified information about costs, benefits, and the time needed before results will be realized. Many organizations have difficulty producing the quantified information, leading to a delay in the decision to start improvements or to abandonment of the improvement initiative before any benefits are gained. A solution is needed to help get data quicker and at a lower cost so valuable improvements can be started.

Why is it so difficult to gather data? Because the process required for gathering valid data is complex and time consuming and needs to be followed carefully to ensure people trust the resulting data. Depending on the scope and level of the measurements, it can take a long time to assemble sufficient data points. Validating data depends heavily on expert knowledge, which is scarce in most companies, and the people involved usually have full agendas.

Organizations could instead use industry data, but doing so is not easy. There is a limited amount of industry data available since most organizations do not want to share data on defects. The data that is available is often difficult to understand and use because it was not sufficiently defined and documented. Common definitions are not used, so combining data from different sources is complex and takes time. Finally, there is a risk that conclusions based on industry data will not be accepted. People sometimes state that "we do things differently here," and, based on that, decide that data from outside the company is not usable.

To be able to make quicker decisions about quality improvements, a business case is needed that can be created with limited but sufficient data. The solution to this problem is to measure *only* what is actually needed. To decide what to measure, the first step is to identify areas of potential improvement. Then measurements can be gathered from those limited areas to define the business case.

Several methods already exist to quantify the cost and benefits of quality improvement. These methods can and have provided strong business cases for improving quality, but their major drawbacks are the lead time and costs involved in building the business case. The rest of this section describes these technologies and their limitations. Appendix A contains further explanations of different quality modeling techniques.

2.1.1 Using Defect Data

When using defects to measure quality, frequently used measures are

- number of defects found during test
- number of defects found by customers
- number of defects found in late test phases that could have been found earlier
- lead times between reporting, analysis, providing a solution, and closure

Most business cases are based on increasing the number of defects found, decreasing defects inserted, or reducing defect slippage by finding defects as early as possible. Models such as Cost of Poor Quality can be used to determine the benefits to be gained.

Industry data on defects has been collected from various companies and projects. This data can be used in an initial business case if no other data is available. Or, a combination of organizational data and industry data can be used to estimate the results of quality improvement and to benchmark organizational performance if an organization has collected its own data.

The problems most organization face when building a business case on defect data become clear as they attempt to answer the following questions:

- What will have to be improved to reduce the costs of defects?
- Where can we get the biggest benefits?
- How soon can we get these benefits?

A large amount of data would need to be collected to get enough information to help determine where to intervene based on defect data alone. Another method or technology is needed to determine where improvements would bring the biggest benefits.

2.1.2 Using Expert Opinion

Most business cases use expert opinion to support decisions about which areas to improve. The problem is figuring out if expert knowledge is valid and applicable to the situation at hand.

Many organizations use audits or appraisals to determine which improvements are needed. They are usually based on standards, research models, or models based on industry best practices, like ISO 9000 and CMMI. Though these models support organizations in finding areas for improvement, they are less helpful in defining the actual benefits that can be gained in those areas. And, since the models are generalized, it is difficult to know if the areas identified will bring benefits in a specific organization.

Others have tried to solve this problem with technologies to quantify expertise, such as Bayesian belief networks [Bibi 2004, Fenton 1999]. These networks can be used to connect variables influencing certain situations and then model the distribution based on a combination of data and expert opinion. Attempts have been made to build general models for quality improvement, but these models were too complex and required too much data and input to deliver valid results.

2.2 Using Defect Data in a Two-Step Approach to Building Business Cases

Ericsson has been collecting data on defect introduction and detection for many years. Initially, the data was used to analyze development and testing processes and benchmark performance by comparing the company's performance against industry data. While much insight was gained

using this approach, Ericsson wanted to further use this data to help define business cases for further quality improvement.

Ericsson uses fault slip through (FST), also known as phase containment, as a quality performance indicator. This indicator shows defects found in later test phases that should have been detected earlier when it would have been more cost effective. Using the FST indicator helps to quantify quality performance and reduce costs. And, since defects found late in a project often impact delivery and release dates, FST can be used as a leading indicator. Therefore, the decision was made to use a decrease in FST as the primary resulting performance indicator in building a quality improvement business case.

The business case for Ericsson was built using a two-step approach that combined subjective expert opinions and objective process and product measurements. First, a global quality performance factors model was developed covering all major contributors to quality at Ericsson. The model, described in Section 3, quantifies quality performance based on expert opinion. The influence of different factors on the quality of intermediate and final products was investigated by gathering expert opinions into a Bayesian belief network (BBN) to model the likelihood of quality improvement in relation to current performance levels. After potential areas of improvement were identified using the BBN model, the second step was to use a Monte Carlo simulation of the promising quality improvements to build a business case. A general overview of BBNs and Monte Carlo simulation can be found in Appendix A. Figure 1 shows a diagram of the two-step approach used in this pilot.

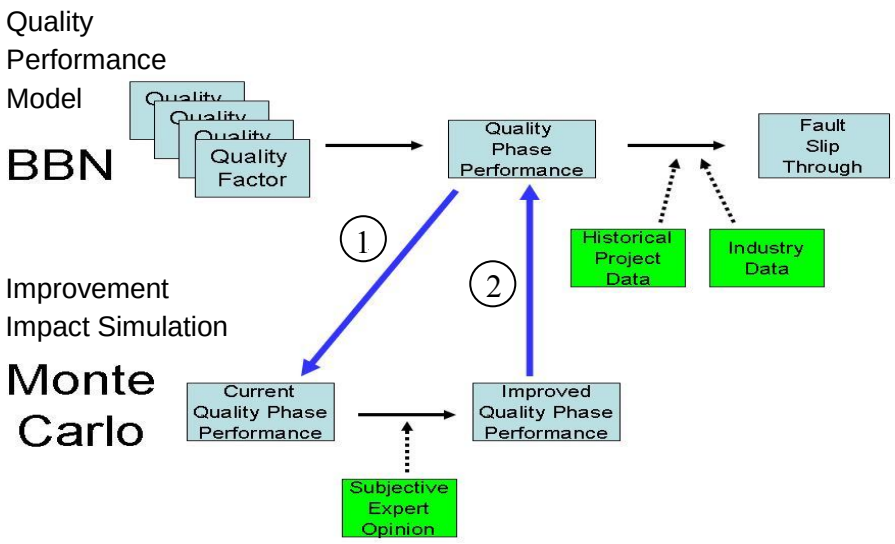

Figure 1: Combined Use of BBN Model and Monte Carlo Simulation at Ericsson

Combining a Bayesian belief network with Monte Carlo simulation provides significant advantages. A BBN can give quick insight into potential areas of improvements based on relevant quality factors and the current performance level of the organization. Monte Carlo simulation enables a detailed calculation of the potential business results for candidate improvement proposals. Since only the identified areas are calculated in detail, the lead time toward a business case is shortened significantly. This enables a company to make quicker decisions when improving quality.

3 Quality Performance Factors Model

This section describes the quality performance factors model developed for Ericsson as a Bayesian belief network (BBN). The purpose of the model was to investigate the usefulness of BBN technology in developing business cases for identifying and evaluating quality improvements. This is by no means a general model.

Factors that could influence quality (i.e., "Quality Factors") were modeled to predict the quality of the products delivered by the organization (i.e., "Quality Performance"). Full descriptions of all quality factors can be found in Appendix B. Factors were selected if they were expected to be important for analyzing quality at Ericsson.

3.1 Development and Use of the Model

The quality factor model for Ericsson was defined based on research, industry reports, and the combined experience of the author and the SEI in the areas of quality, measurement, and analysis. The model contained both technical phases (e.g., requirements, design, implementation, and test) and management activities (e.g., line management, process management, and project management). The model was validated by review with people inside and outside Ericsson.

The model was then used to investigate quality factors that might affect managerial and technical performance. A BBN was built with nodes representing factors influencing quality. Arrows connect the nodes into phases. See Figure 2 for a high-level representation of this model.

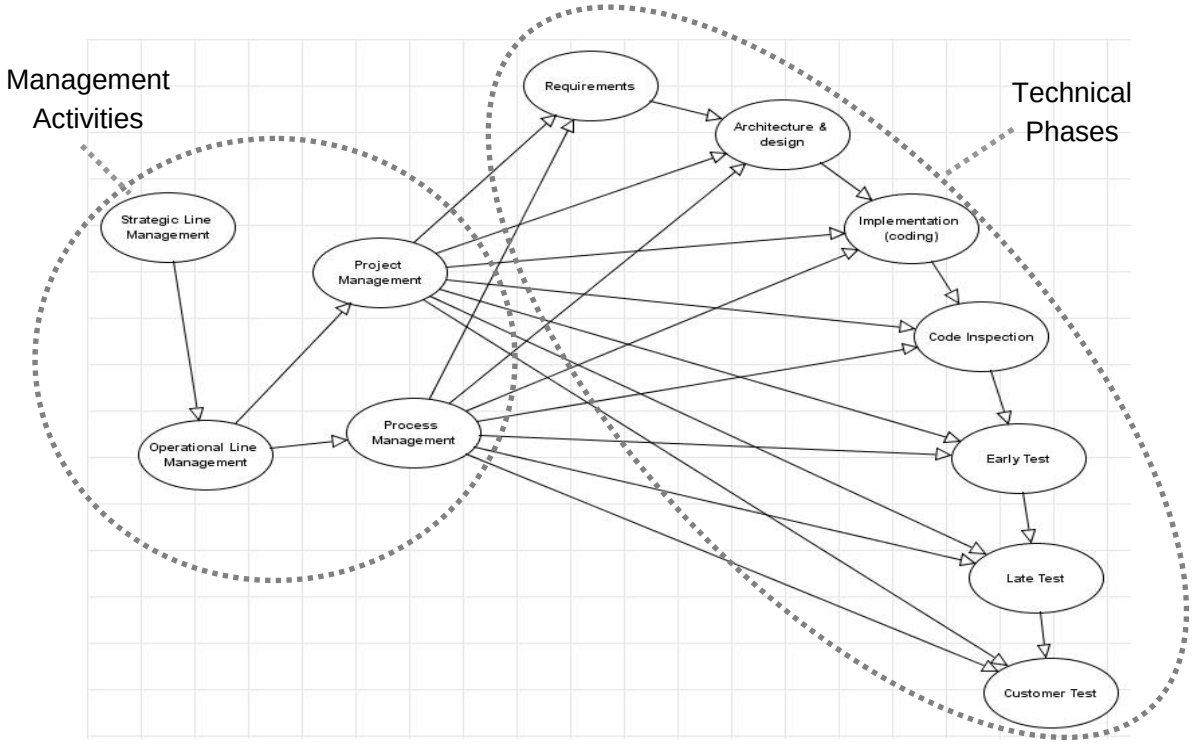

Figure 2: Quality Phases Model

Each phase had a set of factors that influenced its quality performance. The factors were chosen based on industrial and research papers. Figure 3 shows one part of the model, the factors influencing requirements quality performance.

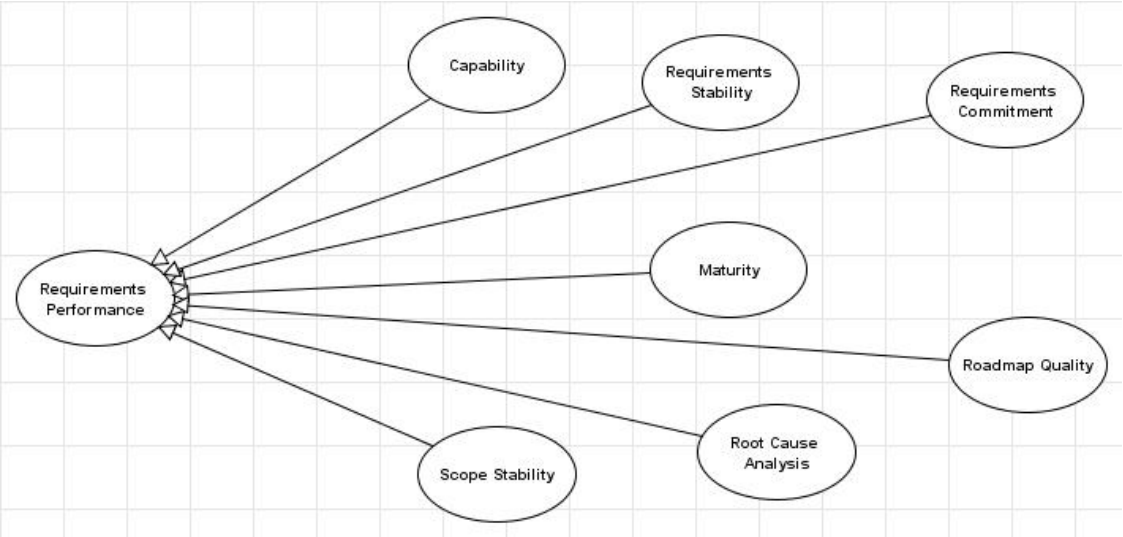

Figure 3: Requirements Performance in Quality Phases Model

Every quality factor is defined. For example, "Requirement Process Maturity," called "Maturity" in the diagram, is defined as "The quality of the defined and baselined requirements management processes, including all supporting material such as training and templates." Appendix B contains descriptions of all the quality factors.

A survey was created with questions grouped according to the phases in the model. These included development phases such as "Architecture and Design" and "Customer Test" and managerial phases such as "Operational Line Management" and "Process Management." Questions about work products from prior developmental phases were included where appropriate.

The survey included two questions per quality factor. The first asked about the extent to which the factor influenced phase performance and the second asked for a judgment of how well the factor itself was implemented. The questions were (1) "How relevant is the factor when we want to improve quality?" and (2) "How well are we doing currently?" Possible answers for phase performance influence were "little if any," "moderate," "substantial," and "extensive." The selections for implementation level were "poor," "fair," "good," and "excellent."

The result of this process was a quality phase performance score. These scores were used to create a quality factor distribution as an indication of the quality of that phase. Higher values mean better quality. The distribution of values can be used as a performance indicator of quality in the organization. Figure 4 shows some examples of the distributions of quality factors that resulted from the survey.

Factor		Q factor A				Q factor B			
Influence Levels	Implemented Levels	Count Influence	% Influence	Count Implemented	% Implemented	Count Influence	% Influence	Count Implemented	% Implemented
Little if Any	Poor	1	3%	8	30%	0	0%	5	16%
Moderate	Fair	7	23%	13	48%	5	16%	17	53%
Substantial	Good	13	42%	5	19%	13	41%	10	31%
Extensive	Excellent	10	32%	1	4%	14	44%	0	0%

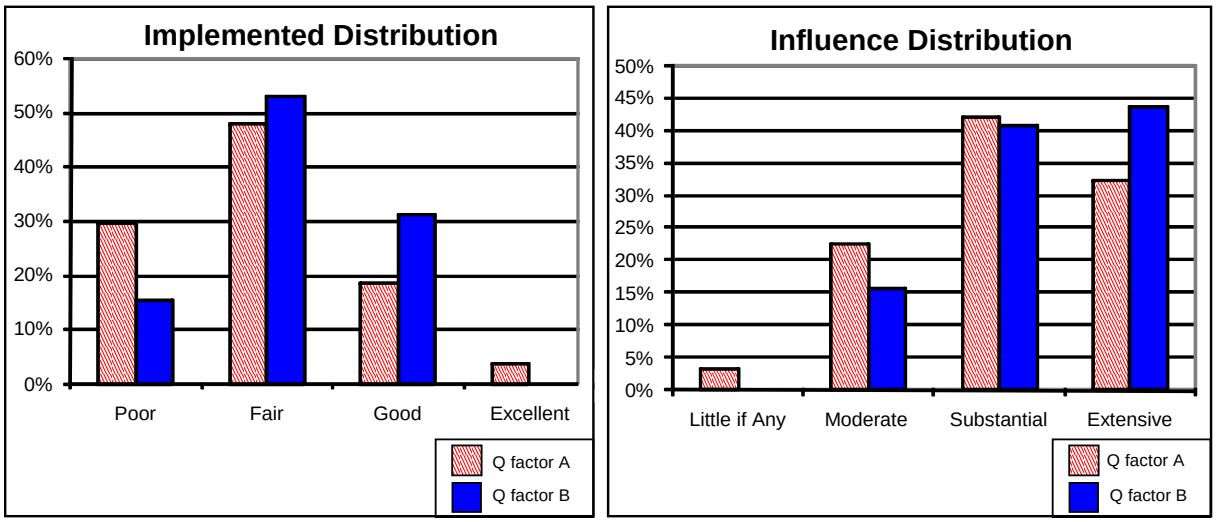

Figure 4: Example Distributions from Quality Factors

The distributions were entered into the BBN model and were used to calculate the quality phase performance. Note that using distributions as input resulted in a quality phase performance distribution. This distribution provides additional insight into the chances that improvements of quality factors would indeed lead to improved performance, a great advantage of BBN modeling when compared with singe point calculations from traditional models.

The quality phase performance was connected in the model to fault slip through (FST) using historical data from the organization and industry data. This modeling assumed that a certain performance level, calibrated using industry data, results in a certain FST.

4 Validating the Business Case for Quality

This section describes the efforts to improve requirements quality, a promising area for improvement identified by the quality factors model. Qualitative and quantitative improvements gained in this effort were used to evaluate the two-step approach for creating improvement business cases.

4.1 Improving Requirements

The quality factor survey described in Section 3 was taken by 34 respondents from management and technical areas. The answers from the survey were entered in the BBN model as distributions. Based on the results, five phases (from the 11) were identified as potential areas for improvement. Potential benefits of improvements in the requirements phase (one of the five areas of improvement) were calculated using Monte Carlo, and based on that the decision was made to investigate whether an agile development approach would decrease defect slip through, supporting earlier delivery of products with at least the same quality. The resulting business case developed for Ericsson involved lowering the number of defects that slipped through to later test phases. Such an improvement would enable an earlier release of products and lower development costs.

The quality factors were linked to historical defect introduction and defect detection data from the organization. Defect data has been collected since 2002 and has been used in projects to provide feedback to the development team and insight for the project steering group on the expected and actual quality of the delivered products. The data was benchmarked with industry data that provided insight into the current performance level and the potential effects on defects if the improvements were made in the requirements process.

The model was calibrated by combining the historical process performance data of projects with industry data. Calibration is very important because it translates the quality factor value into actual process performance. Because we already had data from more than 25 projects, we had the advantage of being able to look at the spread in data, trends, and factors related to the different performance levels of the various projects instead of using a single value. But even with this amount of data, the level of certainty and accuracy of the predictions was limited.

4.1.1 Monte Carlo Simulation

While a BBN can be useful in creating "what if" scenarios to determine the phases where quality improvements would bring the most benefits, more detail on the required investments and expected benefits is needed to build a business case. This can be done using Monte Carlo to simulate the results of quality improvements. Monte Carlo simulation provides a distribution of the estimated benefits instead of a single figure, giving an organization more insight into the chance specific benefits can be realized. The level of uncertainty is made visible, allowing an organization to decide on the amount of risk they are willing to take.

In the requirements phase we saw potential for improving the definition and clarification of requirements and improving the stability of individual requirements and the scope of projects. Since Ericsson was interested in deploying agile methods, a Monte Carlo simulation was done to calcu-

late potential reduction of defects slippage based on the calibrated data from the historical project performance and the industry data.

People who were to be involved in the agile pilot project were interviewed regarding requirements. The expectation was that agile practices such as the "planning game" (the primary planning process in extreme programming), involving stakeholders, and the definition of prioritized user stories in the backlog would improve the quality of the requirements. The results from the interviews were documented and quantified to calculate the expected improvement of individual requirements quality factors. Since no industry data was available on the effect of agile requirements practices on defect levels of delivered products, several assumptions were simulated and a selection was made based on expert opinion of which performance improvements would result from the agile practices.

Next, Monte Carlo was used to simulate the results of using agile development to improve in requirements. Base data for the Monte Carlo model was extracted from the BBN network. Given the spread in data from the survey, a normal distribution was chosen for the influence and level of the individual quality factors of requirements. The median and standard deviation of the individual quality factors were used as input for the simulation. The simulation was run using the calculated quality improvement factors and resulted in a total quality performance distribution for requirements.

The requirements quality performance distribution was fed back into the BBN model to calculate the number of requirements defects created during a project. Data from earlier projects was used as a baseline for this calculation to determine the effect on defect slippage into test. The simulation showed a small reduction in defect slippage within the project and a larger reduction of defect slippage toward maintenance for defects that would have been found after release of the product.

Using Cost of Poor Quality data, savings could be calculated as fault slippage decreased. The business case with cost calculations showed significant savings during the product life cycle. This supported the decision to go ahead with agile development.

4.2 Piloting Agile to Prevent Requirements Defects

The agile pilot project consisted of three teams that developed new functionality for an existing mature product. Iterations of two or three weeks were done, starting with a planning game and finishing with a demo and a retrospective. Within the iteration, no logging of defects was done.

The approach used to measure and track quality consisted of the following steps:

1. Estimate the amount of latent product defects after demo using the planning game.
2. Collect all defects during the test phases (after the demo).
3. Classify defects in "Introduction" phase and "Could Have Been Detected" phase.
4. Do root cause analysis on defects that should have been found before the demo.
5. Decide on improvement actions and present them to the project team.

Specific attention was given to defects related to requirements. Only a limited number of defects were classified as requirements-related. Given the significant cost that requirements defects usually have [Linders 2003b], we chose to do a root cause analysis on them. Analysis of these defects

showed that some defects had to do with the planning game, where the functionality for the iteration was agreed upon. Root cause analysis showed that the development team and the test team did not have the same understanding of the scope, which resulted in defect reports on not-yet-released functionality. Other defects were related to changes in platform products. The impact of some of the changes was not sufficiently clear, leading to defects found at a late testing stage. However, these defects could not have been found earlier in an economical way and had only minor impact on the project. Only one major requirements defect remained, related to an issue that should have been discussed and solved in the planning game. This defect was found during the first test runs. No similar defects were found in the later iterations, so the assumption is that the team improved planning games, which resulted in no similar requirements defects slipping through.

The pilot team recognized the following benefits from using the agile approach:

- team understanding of the requirements increased due to the planning game and stand-up meetings
- improvements were suggested in the retrospectives that increased the cooperation between the development team and the product owner
- perception among team members that requirements quality had increased in the organization

The pilot results matched the predictions from the BBN models of a small reduction of the already low number of requirements defect slippage.

5 Conclusions and Future Work

5.1 Evaluation of the Model

The primary aim of this research was to explore technologies used to model quality and evaluate the applicability of the model in an industry setting, not to build a generic, reusable quality model. The model defined for Ericsson contains quality factors applicable only to that specific organization and has been deployed in only one pilot project so far. It is too early to draw general conclusions about the use of the quality model.

However, this effort was successful in developing a first approach to using Bayesian belief networks and Monte Carlo simulation to combine subjective expert opinions and objective process and product measurements to build a business case for effective and efficient improvement of software quality.

Based on the pilot, the following conclusions were drawn about the application of the two-step approach in this particular context. Using a BBN to identify the areas with the most potential for improvement was successful in helping the organization decide to focus on improving requirements quality. The BBN was useful in creating a picture of the factors that influence quality. These factors were based on objective data and expert opinion. The BBN combined the factors into one model, useful both for analyzing the current situation and exploring "what if" scenarios for alternative solutions to improve quality.

The most promising alternatives were further quantified using Monte Carlo simulation. This requires additional data, but only for a limited number of quality factors—reducing both cost and time to get the data. Calculating potential results of the improvement using Monte Carlo simulation was difficult to evaluate in this pilot. Monte Carlo simulation helped Ericsson calculate the potential benefits of reducing requirement defects. However, since only a limited number of defects existed from that phase, it proved to be difficult to define the benefits in quantitative terms. That is why root cause analysis was then used to get a better insight into the benefits, which confirmed that our improvements had been beneficial. To do that calculation, data was used from past projects to define a baseline and estimate the potential level of improvement.

Overall, the pilot showed that the decision to focus on improving requirements led to improved product quality. Defect slippage was reduced and understanding of the requirements increased using agile. Although positive results were gained from improving in this area (identified by the BBN), we do not know if improvements in other areas would have been more beneficial since a cost/benefit analysis was not done. This step required a limited investment in time and money.

Using a BBN and Monte Carlo significantly reduced the time needed to build a business case. The two-step approach proved to be a practical and efficient way to develop a well-reasoned and robust business case for improving requirements engineering and management using agile development. It also helped prioritize improvements based on the expected value for the business, which led to a quicker return on investment.

5.2 Future Work

There is much potential for the techniques used in this investigation. Models similar to the BBN quality factor model can help organizations decide what to measure (i.e., which data is most relevant to improving their performance). Since collecting a full set of data can be very expensive, the BBN model can help an organization estimate the business benefit that collecting certain data can bring.

These techniques could also be applied to investigate aspects of improvement beyond fault slip through. One interesting measurement to consider for requirements improvement is measuring waste. Waste, or non-value adding activities, would in this context mean requirements that were developed but never used by customers. Many publications suggest that agile can significantly reduce waste because it helps set and evaluate priorities, determine why a requirement is needed, and provide early feedback.

More work could also be done in the research and deployment of data for business cases using distributions, which can provide insight into the chance that a certain improvement will bring benefits. This research has taken a first step in that direction. Potential next steps include building and investigating different distributions for quality factors based on industry data and expert opinions and using these distributions in process performance models for measuring and steering performance.

Appendix A: Techniques Used for Modeling Quality

Models show the relationships among process tasks to provide a better understanding of the entire process. A business case for quality improvement should include an estimate of the expected results; a quality model is often used to produce this estimate. The model used should support an organization in making good decisions about which technology to use based on the circumstances and desired results.

As stated by Tore Dyba, "We do not expect a technology to be universally good or universally bad, only more appropriate in some circumstances and to some organizations than to others" [Dyba 2005]. This appendix provides an overview of techniques that can be used for developing a business case and selecting among quality improvement alternatives. Together the techniques provide a set of strengths that none of the techniques can provide alone.

Defect Introduction & Detection Models

Defect introduction and detection models use defects as indicators of product quality. Software development is measured by looking at the introduction and detection of defects. Defects are introduced into documents or the actual product during the specification, design, and coding phases. Measuring defect introduction can indicate the quality of the development phase. Detection of defects is done through inspections and testing during all phases of the project. Measuring defect detection gives insight into the effectiveness of verification phases. Using these two measures, a project can determine if there is a quality risk and where that risk originated. The measures can indicate whether there were too many defects in the product or if there was insufficient inspection and testing to capture the defects—or both.

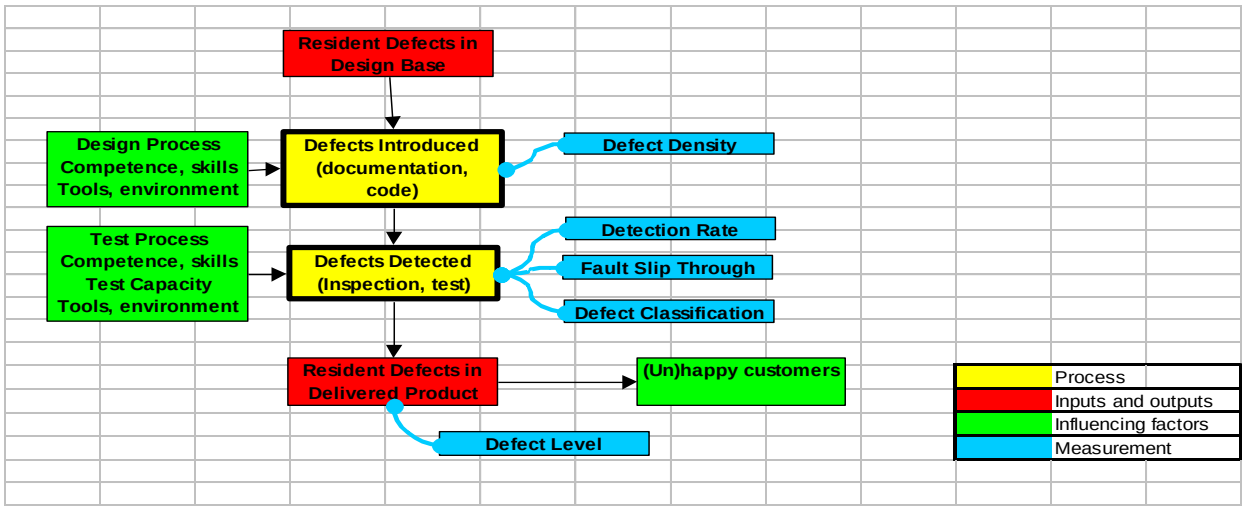

Figure 5: Defect Introduction and Detection Model

Some defect introduction and detection models can be found in books by Watts Humphrey and Stephen Kan [Humphrey 1989, Kan 2002]. Several improvement scenarios based on such models are described in "Improving Defect Removal Effectiveness for Software Development" [Leung 1998]. Application of a defect introduction and detection model for a research and development

center at Ericsson is described in "Controlling Product Quality During Development with a Defect Model" and "A Proactive Attitude Toward Quality: The Project Defect Model," [Linders 2003a, Linders 2004].

This kind of model can be used to plan and track quality activities in projects and to tailor the development process in order to meet quality requirements. Data from the model can support decision making in projects. The application at Ericsson shows how the defect detection rate indicated possible quality risks and can be used to inform project decisions [Linders 2003a]. Also, combining quality data with cost helped to guide investments in a project that would improve quality [Denneman 2005].

Bayesian Belief Networks

Bayesian belief networks (BBNs) offer another approach to modeling the quality of software products. An overview of BBN usage in software development can be found in "Bayesian Belief Networks for Test-Driven Development" [Periaswamy 2006]. While covering BBNs in depth is often done in semester-long graduate courses, this section presents the basic principles behind their development and use. This section will allow practitioners to gain an appreciation for the use of BBNs and to consider using them in concert with the other modeling techniques discussed in this report.

Bayesian belief networks are probabilistic models with sets of nodes connected with arrows that represent relationships of one of the following forms: a) cause and effect b) leading indicator without a confirmed cause and effect, or c) an observed statistical correlation between two nodes. Essentially, any node (to be called a "child" node) may have one or more incoming arrows from what are called "parent" nodes. For example, a node with four incoming arrows has four different parents that each influence or help to explain what to expect for behavior of the child node. In a BBN, most nodes have parents. Nodes that do not have parents are called "root" nodes and operate strictly on observed historical or subjective distributions of behavior of the root node itself.

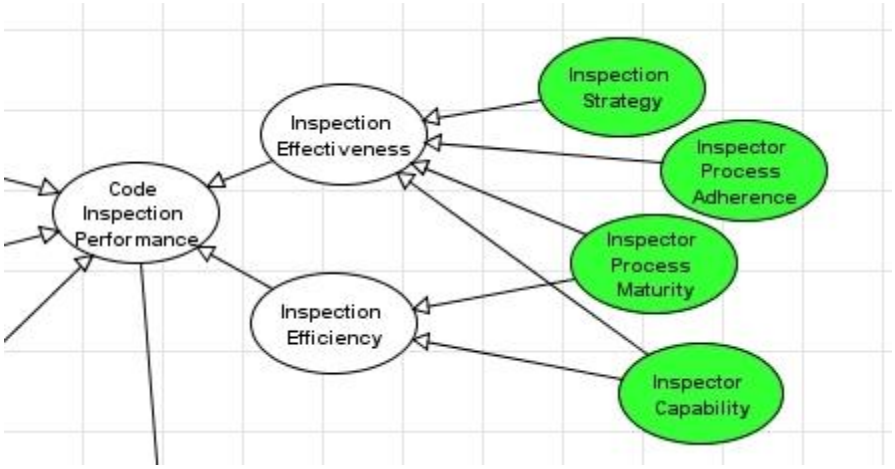

Figure 6: Nodes in a BBN

In practice, BBNs can be constructed to cover a time dimension such as a life cycle. In this manner, the BBN is constructed from right to left. On the far right of the model, the final performance or quality outcomes are identified as child nodes. Then an iterative process is conducted by identifying what factors occurring just prior to the final outcome child nodes might explain the child

nodes. The iteration continues in that each parent node can be explained in terms of its own parent nodes occurring earlier in the life cycle.

The benefits of such a model include the ability to tie early management or technical decisions to the final outcome. A BBN can be an invaluable management tool by enabling a "what-if" capability during project planning, and subsequently during all processes and subprocesses in the life cycle. For this reason, BBNs represent an ideal approach to CMMI process performance modeling as discussed in the CMMI Quantitative Project Management (QPM) and Organizational Process Performance (OPP) process areas, and as used in the Causal Analysis and Resolution (CAR) and Organizational Innovation and Deployment (OID) process areas.

In "Software Process Modeling with Bayesian Belief Networks," the advantages of BBNs are summarized: "BBNs are highly visual tools that can be easily explained, indicating which workflows affect others" and "the final objective … is the ability of BBN to facilitate effective planning, control, and operational management of the process" [Bibi 2004]. The paper shows a generalized model usable for effort estimation. With some adaptations this model can be used to estimate quality within a project, but the question remains as to whether a model could be made to predict quality improvement benefits on an organizational level.

A Bayesian approach to developing business cases has the following strengths:

- takes prior knowledge onto account
- can be used in situations with scarce failure data
- can predict situations that are new
- can formally incorporate objective and subjective information
- can include decision nodes [Stoddard 2003]

Theory and application of BBNs for defect prediction is investigated by Martin Neil and Norman Fenton [Neil 1996, Fenton 1998, Fenton 1999]. They modeled the quality of the software products in single, isolated projects in several organizations. For a business case, a model on an organizational level is needed that is capable of predicting one or more business target results. This is called a process performance model and is used to build a business case and plan and track organizational quality performance.

Hadar Ziv and Debra Richardson, in their Maxim of Uncertainty in Software Engineering (MUSE), have stated that "uncertainty is inherent and inevitable in software development processes and products." They conclude that Bayesian models could provide solutions to manage with uncertainties. In "Bayesian-Network Confirmation of Software Testing Uncertainties," they identify several areas of uncertainty in software development and describe an approach to test case selection based on uncertainties [Ziv 1997]. They also state that "Predicting certain qualities or properties of planned development activities or artifacts is most difficult but also most beneficial to developers." Again this is a single-project approach. One management decision they discuss is how long testing should take; however, this decision only has limited use since the damage has already been done by the time testing takes place. Organizations need information to prevent problems and reduce the need for testing, not to reduce test time. In an earlier paper on the uncertainty principle, they mention the importance of monitoring the test process [Ziv 1996]. For organizations with insufficient defect detection practices, this could be used to build an interesting

business case. However, for defect prevention, we need models focusing on the requirements, architecture, and design processes instead of testing.

Defection prediction models for maintenance resource planning are compared in "Selecting a Defect Prediction Model for Maintenance Resource Planning and Software Insurance" [Li 2003]. This paper mentions as strengths of Bayesian models their ability to use prior information for initial estimation and to use information as it becomes available to improve and adjust estimates. The model is deployed, however, to predict product maintenance while we would like a model that predicts design and test work and can be used to investigate and decide on process improvements in that area.

Research has been done as to whether BBNs could be applied for risk management [Fan 2004]. Detailed models have been developed using factors that influence design quality and remaining defects, but the large amount of factors make them difficult to use. This again highlights the problem we have: How can the model be reduced to the factors that have major impact? For a general purpose model, doing so is not easy. Furthermore, the models appear to be complex, which can hinder the validation and adaptation in industry.

BBNs look promising for modeling quality on an organizational level. Some attempts have been made to build probability models for testing using BBNs. However, none of these networks is complete or has been validated. BBNs also struggle with complexity of the modeled processes, which can make it impossible to deploy them at an organizational level [Neil 1996, Bibi 2004, Ziv 1997, Wooff 2002]. A usable model should include only the main influencing factors of quality at a high abstraction level.

Monte Carlo Simulation

Monte Carlo simulation is a quantitatively linked model of factors, some of which are uncertain and represented by a distribution. Monte Carlo simulation shows the distribution of the outcome factors (child nodes) based on the certain and uncertain parent nodes. Monte Carlo simulation tools use random number generators to select a value for each parent node from the parent node's distribution and then compute the outcome child node. This step is repeated thousands, if not hundreds of thousands, of times so that a distribution of values for the child node is formed. The distribution for each child node can be used in Monte Carlo simulation for any successive child nodes of the child node. Additionally, the Monte Carlo simulation provides distributions for all child nodes such that a confidence level can be established for any given value of the child node. This proves quite helpful in using results from Monte Carlo simulation in project forecasting, business case development for process improvement, and risk management.

Monte Carlo simulation can be enhanced by adding decision variables within a network of parent and children nodes. In this manner, the normal Monte Carlo simulation iterations are performed within a loop that controls the settings of decision variables. In effect, each setting of the collection of decision variables receives its own Monte Carlo simulation iterations.

Monte Carlo is a proven simulation technique that has been used to predict the benefits of process improvements. However, a complete Monte Carlo model for an entire organization could become very complex and would require specific knowledge of relationships.

Appendix B: Quality Factor Model

This appendix describes the quality factors used in the pilot. Please note that these factors have been selected and defined based on the needs of the pilot organization, Ericsson R&D in The Netherlands. This is not a generic, universal model.

Strategic Line Management Performance

Strategic Line Management includes the R&D Management Team (R&D Manager and Unit Managers) and the strategic work of the department managers. It does not include their operational work, such as resource management for projects.

Quality factors:

1. Quality Commitment – The amount management is willing to invest in quality.
2. Quality Reward Policy – How employees are rewarded for their quality-related behavior. (You can stimulate behavior that leads to quality by specifically recognizing and rewarding it.)
3. Leadership Capability – The capability of strategic managers to affect the organization's quality mission (e.g., their focus on quality priorities, insight into the operation, and how much they value quality performance and results).
4. Quality Culture – A culture that enables people to strive for quality.
5. R&D Balanced Scorecard Deployment – Usage of the scorecard to steer quality (i.e., setting targets on quality, overseeing and managing the relationship between quality targets and other targets on the scorecard, communicating, measuring and analyzing quality targets, defining corrective actions and following them to compliment, and evaluating the scorecard with regard to the quality strategy and feedback from operational results).

Operational Line Management Performance

Operational Line Management includes department managers in their role as responsible for short-term activities.

Quality factors:

1. Resource Capability Management – Ability of line management to improve and sustain improvements in the skills of employees, both technical skills and interpersonal (e.g., collaboration, communication).
2. Resource Allocation – Allocation of developers, testers, and support staff to projects.
3. Resource Stability – The same people remain on a project until completion without being replaced, with a focus on key resources such as technical coordinators, design leaders, test leaders, team leaders, and subproject managers.
4. Schedule Pressure – The way deadlines are used to put pressure on projects and people to deliver on time.

5. Operational Overview and Insight – Insight into the status of ongoing projects (e.g., processes used, documents delivered, quality of the documents).
6. Decision Making Capability – The ability to balance quality, time, cost, and functionality and to make timely decisions that involve the right people. Also to assure that decisions are communicated and that the work is followed up to completion.

Project Management Performance

Project Management includes main project management and subproject management (e.g., design and test projects); all activities normally performed by a project manager.

Quality factors:

1. Decision Making Capability – The ability to balance quality, time, cost, and functionality and to make timely decisions that involve the right people. Also to assure that decisions are communicated and that the work is followed up to completion.
2. Project Portfolio Management – Planning and tracking of the set of projects, including project steering groups and all decisions made to start, continue, cancel, and conclude the project.
3. Project Management Capability – Skill and experience level of project managers.
4. Risk Management Process Capability – Awareness of project risks, the maturity of the process, and the capability of managing risks.
5. Planning Capability – The ability to estimate, plan, and track projects with respect to the quality of the delivered product.
6. Scope Stability – Impact of major changes in the projects (e.g., those related to stability of the products to be developed), the development teams involved in the projects, and major changes in project funding or delivery dates. These changes are often related to changes in the product roadmap.
7. Schedule Pressure – The way deadlines are used to put pressure on projects and people to deliver on time.
8. Operational Overview and Insight – Insight into the status of ongoing projects (e.g., processes used, documents delivered, quality of the documents).
9. Operational Line Management – Activities done by department managers responsible for the short-term activities.
10. Project Management Process Adherence – Checks (e.g., audits or assessments) to determine whether the baselined processes are being followed and if they are effective and efficient.

Process Management Performance

Process Management includes defining and baselining the processes to be used for management and technical work; the support in operational usage of the processes in training, instruction, tools and templates; and the availability of websites and experienced personnel.

Quality factors:

1. Process Management Capability – Capability to develop and baseline processes and support their deployment in projects and other line activities.
2. Organizational Learning – The capability to learn and improve processes and to have these improved processes accepted and deployed in projects and other line activities. Learning can be based on feedback from process deployment (e.g., from assessments, audits, root cause analysis, or retrospectives).
3. Development Organization Flexibility – The ability of the organization to adapt to changing circumstances while maintaining the quality of the delivered products.
4. Development Organization Efficiency – The capability of processes to ensure that customer-specific products are developed quickly, with low cost and high quality.
5. Operational Line Management – Activities done by department managers responsible for the short-term activities.
6. Process Management Process Adherence – Checks (e.g., audits or assessments) to determine whether the baselined processes are being followed and if they are effective and efficient.

Requirements Performance

Requirements performance means specifying the products to be developed and supporting activities such as requirements clarifications, reviews and inspections, and tracing.

Quality factors:

1. Requirements Management Capability – Skill and experience level related to the requirements managing activities (done by the projects requirements manager).
2. Requirements Commitment – Agreements between the strategic product manager and the project managers, where the project/team commits to deliver certain functionality.
3. Requirements Stability – Inverse of the amount of requirement changes over time. (The less changes, the higher stability.)
4. Requirements Process Maturity – The quality of the defined and baselined requirements management processes, including all supporting material such as training and templates.
5. Roadmap Quality – Usability of the roadmap with respect to requirements management for projects (e.g., information about when to develop which product versions; quality of the business case for a product version; allocation of scope to a version, decisions about product introduction, end of maintenance, and phase-out dates).
6. Scope Stability – Impact of major changes in projects that are related to changes in the product roadmap, including stability of the products to be developed, development teams involved in projects, and major changes in project funding or delivery dates.
7. Root Cause Analysis – Capability to learn from defects found during development. By analyzing defects, determining common causes (related to processes, tools, and development environment, capabilities, management, and organization), and defining actions to prevent them from recurring.
8. Requirements Definition Capability – The skill and experience level of the people doing requirements definition (e.g., product managers).

Architecture and Design Performance

Architecture and design performance includes the activities to define and support the solution architecture (i.e., functionality) and the software architecture (i.e., structure of software and non-functional requirements), and the global and detailed designs prepared by development teams before writing the code.

Quality factors:

1. Architecture Stability – The quality and stability of the architecture definitions that are defined in solution architecture documents and supported by system management.
2. Analysis & Design Capability – The skill and experience level of architecture and design teams. This includes the capability of communication and cooperation between the teams.
3. Root Cause Analysis – Capability to learn from defects found during development, (e.g., analyzing defects, determining common causes (related to processes, tools and development environment, capabilities, management & organization, etc), and defining actions to prevent them from recurring).
4. Requirements Performance – Result of the previous phase (i.e., specifying the products to be developed and the supporting activities such as requirements clarifications, reviews and inspections, and tracing).
5. Project Management Performance – Definition, planning, tracking, and control of quality in the development projects and the delivered products.
6. Process Management Performance – Defining and baselining the processes to be used for management and technical work. The support in operational usage of the processes for training, instructions, tools, and templates and the availability of websites and experienced personnel.
7. Architecture & Design Process Maturity – The quality of the defined and baselined architecture and design processes, including all supporting material such as training and templates.

Implementation (Coding) Performance

Implementation (coding) means writing the modules in a programming language.

Quality factors:

1. Design Environment – The quality of the coding environment (e.g., supported functionality, stability, and performance).
2. Root Cause Analysis – Capability to learn from defects found during development, (e.g., analyzing defects; determining common causes related to processes, tools, development environment, capabilities, management, organization, etc; and defining actions to prevent them from recurring).
3. Implementation Capability – The skill and experience level of the design teams in coding.
4. Architecture & Design Performance – Result from the previous phase (i.e., the usability and quality of the architecture and design for the development teams writing the code).
5. Project Management Performance – Definition, planning, tracking, and control of quality in the development projects and the delivered products.

6. Process Management Performance – Defining and baselining the processes to be used for management and technical work. The support in operational usage of the processes for training, instructions, tools, and templates and the availability of websites and experienced personnel.
7. Design Base Quality – Quality of the legacy product used as a design base for developing the new product or product version.
8. Implementation Process Maturity – The quality of the defined and baselined implementation processes, including all supporting materials such as training and templates. This includes, where appropriate, the usage of agile or other programming methods, such as user stories, pair programming, or patterns.

Implementation (Code) Inspection Performance

Implementation (code) inspection means finding defects directly in the source code using techniques such as walkthroughs, reviews, and formal inspections.

Quality factors:

1. Inspection Strategy – The strategy describing which inspections are performed in the project and which kinds of defects are expected to be found in them.
2. Inspection Process Adherence – Checks such as audits or assessments to determine whether the baselined processes are being followed and if they are effective and efficient.
3. Inspection Capability – The skill and experience level of the people doing inspections.
4. Inspection Process Maturity – The quality of the defined and baselined processes, including all supporting materials such as training and templates.
5. Implementation (coding) performance – Quality of the code resulting from the previous implementation phase prior to inspection.
6. Project Management Performance – Definition, planning, tracking, and control of quality in the development projects and the delivered products.
7. Process Management Performance – Defining and baselining the processes to be used for management and technical work. The support in operational usage of the processes for training, instructions, tools and templates, and the availability of websites and experienced personnel.

Early Test (BT) Performance

Early test means basic tests done by the developer on a module or unit level.

Quality factors:

1. Early Test Process Adherence – Checks (such as audits or assessments) to see if the baselined processes are being followed and if they are effective and efficient.
2. Early Test Strategy – The strategy that describes which early test phases are performed in the project and which kinds of defects are expected to be found in them.
3. Early Test Capability – The skill and experience level of the people doing the testing.

4. Early Test Process Maturity – The quality of the defined and baselined processes, including all supporting material such as training and templates.
5. Early Test Environment – The quality of the test environment (e.g., tools and supported functionality, stability, and performance).
6. Platform Quality – The functionality and stability of the product platform (e.g., operating system, middle ware, and support libraries).
7. Requirements Performance – Result of the previous phase (e.g., specifying the products to be developed and the supporting activities such as requirements clarifications, reviews and inspections, and tracing).
8. Implementation (Code) Inspection Performance – Result of the previous phase (i.e., all ways to find defects directly in the source code, such as walkthroughs, reviews, and formal inspections before the code is submitted to test).
9. Project Management Performance – Definition, planning, tracking, and control of quality in the development projects and the delivered products.
10. Process Management Performance – Defining and baselining the processes to be used for management and technical work. The support in operational usage of the processes for training, instructions, tools and templates, and the availability of websites and experienced personnel.

Late Test (FT, LSV, VFE&VPE, etc) Performance

Late Test includes all test phases done by someone other than the developer of the code (after basic test) until testing for delivery (MDA), at Ericsson typically including FT, LSV, VFE&VPE, EtE or NIT.

Quality factors:

1. Late Test Process Adherence – Checks (such as audits or assessments) to see if the baselined processes are being followed and if they are effective and efficient.
2. Late Test Strategy – The strategy that describes which late test phases are performed in the project and which kinds of defects are expected to be found in them.
3. Late Test Capability – The skill and experience level of the people doing the testing.
4. Late Test Process Maturity – The quality of the defined and baselined processes, including all supporting material such as training and templates.
5. Late Test Environment – The quality of the test environment (e.g., tools and supported functionality, stability, and performance).
6. Platform Quality – The functionality and stability of the product platform (e.g., operating system, middle ware, and support libraries).
7. Requirements Performance – Result of the previous phase (e.g., specifying the products to be developed and the supporting activities such as requirements clarifications, reviews and inspections, and tracing).
8. Early Test Performance – Results of the previous phase (i.e., output quality after basic tests are done by the developer on the module/unit level).

9. Project Management Performance – Definition, planning, tracking, and control of quality in the development projects and the delivered products.
10. Process Management Performance – Defining and baselining the processes to be used for management and technical work. The support in operational usage of the processes for training, instructions, tools and templates, and the availability of websites and experienced personnel.
11. In what major ways, if any, do the quality factors described in this survey have **different** effects on the various approaches used in the late testing phase (FT, LSV, VFE&VPE, EtE, NIT)? (Please describe here.)

Customer (MDA, FOA) Test

Customer Test includes all testing done together with the customer, either at an Ericsson test plant or the customer site. This would normally be the FOA test. Also included is the MDA test (i.e., the last installation and orderability check before the product is released).

Quality factors:

1. Customer Test Capability – The skill and experience level of the people doing the testing.
2. Late Test Performance – Results of the previous phase (i.e., the output quality after test done in late testing [LSV, VFE&VPE, EtE, or NIT]).
3. Customer Test Environment – The quality of the test environment (e.g., tools and supported functionality, stability, and performance).
4. Customer Test Strategy – The strategy that describes which late test phases are performed in the project and which kinds of defects are expected to be found in them

References

[Bibi 2004]
Bibi, S. and Stamelos, I. "Software Process Modeling with Bayesian Belief Networks," presented at 10th International Software Metrics Symposium (Metrics 2004), Chicago, IL, 2004.

[Denneman 2005]
Denneman, M. "Cost of Quality at Ericsson R&D Rijen: The Cost of Defects," *Quality and Reliability Engineering*. Eindhoven, The Netherlands: Eindhoven University of Technology, 2005.

[Dyba 2005]
Dyba T., Kitchenham, B. A., and Jorgensen, M. "Evidence-Based Software Engineering for Practitioners." *IEEE Software 22*, 1 (January 2005).

[Fan 2004]
Fan, C. F., and Yu, Y. C. "BBN-Based Software Project Risk Management," *Journal of Systems and Software 73*, 2 (October 2004).

[Fenton 1998]
Fenton, N. E. and Pfleeger, S. L. *Software Metrics: A Rigorous & Practical Approach*, 2nd edition, Course Technology, 1998.

[Fenton 1999]
Fenton, N. E. and Neil, M. "A Critique of Software Defect Prediction Models," *IEEE Transactions on Software Engineering 25*, 5 (September/October 1999).

[Humphrey 1989]
Humphrey, W. *Managing the Software Process.* Addison-Wesley Professional, 1989.

[Kan 2002]
Kan, S. H. *Metrics and Models in Software Quality Engineering*. Addison-Wesley Professional; 2nd edition, 2002.

[Leung 1998]
Leung, H. K. N. "Improving Defect Removal Effectiveness for Software Development," presented at IEEE 2nd Euromicro Conference on Software Maintenance and Reengineering, Florence, Italy, 1998.

[Li 2003]
Li, P. L., Shaw, M., and Herbsleb, J. "Selecting a Defect Prediction Model for Maintenance Resource Planning and Software Insurance," presented at the Fifth Workshop on Economics-Driven Software Research, Portland, Oregon, 2003.

[Linders 2003a]
Linders, B. "Controlling Product Quality During Development with a Defect Model," presented at the European Software Engineering Process Group, London, England, 2003.

[Linders 2003b]
Linders, B. "The Business Benefit of Root Cause Analysis," presented at Software Management & Applied Software Measurement (SM/ASM), San Jose, CA, 2003.

[Linders 2004]
Linders, B. and Sassenburg, H. "A Proactive Attitude Toward Quality: The Project Defect Model," *Software Quality Professional 7,* 1 (December 2004).

[Neil 1996]
Neil, M. and Fenton, N. "Predicting Software Quality Using Bayesian Belief Networks," presented at 21st Annual Software Engineering Workshop NASA/Goddard Space Flight Center, 1996.

[Periaswamy 2006]
Periaswamy,V. and McDaid. K. "Bayesian Belief Networks for Test-Driven Development," *Proceedings of World Academy of Science, Engineering, and Technology 11*, (February 2006).

[Stoddard 2003]
Stoddard, B. "You Are a Bayesian - You Just Don't Know It!" presented at the Six Sigma for Software Development Conference, San Jose, CA, January 2003.

[Wooff 2002]
Wooff, D. A., Goldstein, M., and Coolen, F. P. A. "Bayesian Graphical Models for Software Testing," *IEEE Transactions on Software Engineering 28,* 5 (May 2002).

[Ziv 1996]
Ziv, H. and Richardson, D. J. "The Uncertainty Principle in Software Engineering," University of California, Irvine, Technical Report UCI-TR96-33, August 1996.

[Ziv 1997]
Ziv, H. and Richardson, D. J. "Bayesian-Network Confirmation of Software Testing Uncertainties," *Lecture Notes in Computer Science Volume 1301*. Jazayeri, Mehdi and Schauer, Helmut (Eds.), 1997.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	July 2009	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Building Process Improvement Business Cases Using Bayesian Belief Networks and Monte Carlo Simulation	FA8721-05-C-0003

6. AUTHOR(S)

Ben Linders

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2009-TN-017

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Many organizations require quality improvement initiatives to be based on quantified business cases. This leads some organizations to start measurement programs to collect data about current performance—a lengthy and expensive process that requires a strong commitment from management. This report describes a collaboration between the Software Engineering Institute and Ericsson Research and Development, The Netherlands, to build a business case using high maturity measurement approaches that require limited measurement effort.

For this project, a Bayesian belief network (BBN) and Monte Carlo simulation were combined to build a business case for quality improvement. Using a BBN gave quick insight into potential areas of improvement based on relevant quality factors and the current performance level of the organization. Monte Carlo simulation enabled a detailed calculation of the likely business results in the areas of potential improvement. This approach led to the decision to implement agile methods to improve the quality of requirements.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Bayesian belief networks, BBN, Monte Carlo simulation, DMAIC, Six Sigma, business case	35

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL