An Industrial Case Study on Visualization of Dependencies between Software Measurements

Ludvig Johansson
IT University of Göteborg
412 96 Göteborg
ludvig@ituniv.se

Wilhelm Meding
Ericsson SW Research
Ericsson, Sweden
wilhelm.meding@ericsson.com

Miroslaw Staron
IT University of Göteborg
412 96 Göteborg
miroslaw@ituniv.se

ABSTRACT
Managing large software projects requires working with a large set of measurements to plan, monitor, and control the projects. The measurements can, and usually are, related to each other which raise an issue of efficiently managing the measurements by identifying, quantifying, and comparing dependencies between measurements within a project or between projects. This paper presents a case study performed at one of the units of Ericsson. The case study was designed to elicit and evaluate viable methods for visualizing dependencies between software measurements from a perspective of project and quality managers. By developing a series of prototypes, and evaluating them in interviews, we get results showing applicability of each visualization method in the context of the studied organization. The prototypes were used to visualize correlation coefficients, distribution dependencies, and project differences. The results show that even simple methods could significantly improve the work of quality managers and make the work with measurements more efficient in the organization.

Categories and Subject Descriptors
D.2.8 [**Metrics**]: *Process metrics, Product metrics.*

General Terms
Management, Measurement.

Keywords
Software metrics, visualization, quality management.

1. INTRODUCTION
Using measurements as a mean of monitoring software projects is a characteristics of mature processes and management practices. The larger the projects, the larger the data sets used and the more measurements collected. One of the daily works of quality managers is to work with measurements to assure the quality of the final product which involves identifying anomalies in data sets. Currently, the identification is based on experience and monitoring of limited number of measurements. Better use of the measurements in projects requires automated support in identifying and visualizing dependencies between measurements, especially when data sets are large. The existing visualization solutions require extensive customization work in order to be adjusted to the processes used at Ericsson and to integrate with existing toolset at the company. In this paper we identified and evaluated a set of visualization methods which do not require any initial investments nor entail large customization/integration costs.

Hence, in this paper we address the following research question:

How can dependencies between measurements be quantified and visualized in the context of a software development unit at Ericsson?

We consider both the dependencies between the measurements, and, to a limited extent, measured entities. Our research is performed in the context of software development organization, which in particular means that our work focuses on project and process measurements.

By *dependency* we mean such relationship between measurements in which a change of value of one measurement causes a change in another measurement.

The results of this study show that simple visualization techniques integrated with MS Excel and mind mapping tools could significantly improve the work of quality managers. Using mind maps to visualize dependencies between the whole (or part of) sets of variables were found to be the method which suits the evaluation criteria best, and visualizing correlation coefficients using colored MS Excel worksheets was found to be the most useful method.

The paper is structured as follows. Section 2 presents the most related research relevant for our work. Section 3 presents the context of the study – measurement systems being used at Ericsson. Section 4 describes the design of the case study and Section 5 presents the results from it. Section 6 addresses the main threats to validity of the study and Section 7 presents the conclusions.

2. RELATED WORK
In the area of software measurement and measurement dependencies, the ISO standards ISO/IEC 15939 [1] and ISO/IEC 9126-1 [2] provide a standardized way of structuring the software measurement process and preserving product quality during the process. These standards, however, are high level standards and

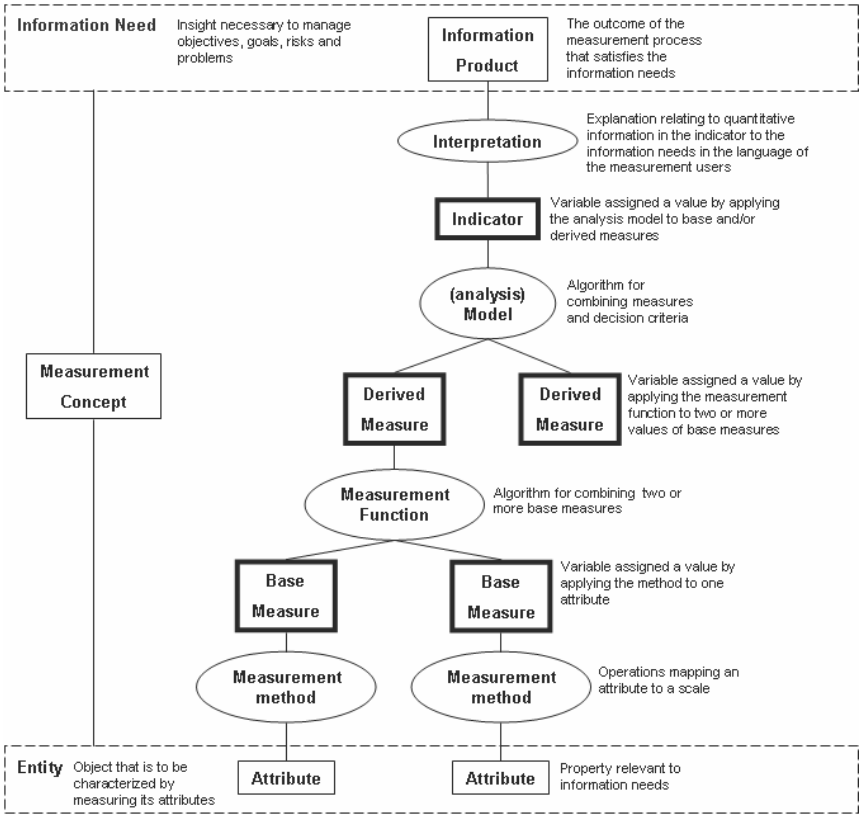

Figure 1, Software Measurement Model [1]

do not give solid theoretical background on how measurements should be used. Such fundamental application of measurement theory for software engineering is provided by Fenton and Pfleeger [3]. This study combines these two views on measurements, i.e. (a) the measurement theory perspective and (b) the ISO/IEC industrial standards perspective, in order to develop methods that are industrially applicable on theoretically solid ground.

The discussion of introducing measurement systems into an organization is, however, not covered in this study. Authors like Clark [4], Kilpi [5] Dekkers and McQuaidfor [6], Pfleeger et al. [7] and Brökcers et al. [8], describe how to/why introduce measurement systems in to an organization, and reflect on problems/solutions that measurements can result in. Their findings are used in the process of introducing the methods described in this paper into the organization.

One of the challenges in this study was to quantify measurements in a correct way – that is, whether it is possible to statistically compute dependencies: methods for statistic calculations are presented in [9, 10].

Techniques for visualizing dependencies in other areas have a wider research base than visualizing measurements in software engineering. The main focus in the existing studies of visualization is on program code dependencies, for example [11-18]. Hence, visualizing measurements dependencies is mostly about visualizing large groups of information complexes, like visualization of code dependencies and SQL dependencies, visualizing techniques provided by Spencer [19] were used as ground for identifying problems around visualizing information. Software measurements are used in the process of visualizing such aspects as code complexity, but then the focus is on the complexity and not on measurement dependency. In our research the focus is on the identification of measurement dependencies, not on complexity or size of source code.

An interesting similarity can be observed between measurements visualization and neural networks, since both include similar calculations [20]. The case-by-case comparison (described later in the paper) is based on analogy-based estimation techniques from the neural networks [21-23].

More advanced techniques for visualizing large quantities of data can be found in [24]. Although the methods presented there are applicable for our context, they required advanced visualization tools, which contradicted the requirements from the organization.

3. MEASUREMENT SYSTEMS

Ericsson is a world-wide telecommunication manufacturer. Its projects vary in size, but the majority comprises of large and long-term projects that involve both hardware and software components. Ericsson has adopted and further developed mature methods for developing software and managing projects, including managing/assuring quality of Ericsson products. The management use measurements together with expert opinions of project managers and engineers as the provider for information and a basis for making decisions – which is a common situation in mature organizations. In large software projects, however, the situation becomes hard to manage since the number of measurements used is very large, which makes it hard to manage the measurements and therefore several decisions are based on experience. In order to make the work with measurements more efficient, the studied organization at Ericsson has adopted the

ISO/IEC standard for software measurements – ISO/IEC 15939 [1].

The ISO/IEC 15939 standard defines the elements of the measurement systems as presented in Figure 1. The measurement process is driven by an *information need* (top of Figure 1). The information need is what the customer (or a stakeholder) of the measurement system wants to know, for example: 'Is the project within budget?', or 'Is the project running according to the schedule?'

In order to satisfy the information need, a series of measurements need to be examined. The measurements are collected by measuring relevant *entities*, for example, a design model, project status, or a process. An entity is a real world entity which has measurable *attributes*. The standard defines an attribute as "a property or characteristic of an entity that can be distinguished quantitatively or qualitatively by human or automated means". The quantification of the attribute is the process of obtaining a *base measure*. Several base measurements can then be merged throughout a *measurement function* to a *derived measurement*. A measurement function is an algorithm or calculation performed to combine two or more base measures.

Further, *indicators* are created from the derived measures to provide an estimate or evaluation of specified attributes derived from the real world. It is the indicators that should fulfill the stakeholder's information need.

Table 1 presents a definition of an example measurement system which is based on a working measurement system at Ericsson.

Table 1. Defect reports measurement system - definition

Concept	Definition
Information Need	How much, compared to the budget of project X, is the cost of defect reports?
Measurable Concept	Budget deviation (budget is fixed, project cost on the other hand is dynamic)
Entity	Budget deviation
Attributes	1. Project X related defect reports 2. Cost of one defect report in project X 3. Budget of project X
Measurement Method	1. Count total number of defect reports 2. Calculate the number of hours per defect report based on data from previous projects [cost] 3. State the budget of project X (no need to calculate, it's only a number)
Base measures	1. NoD – Number of Defects 2. DC – Defect Cost 3. PB – Project Budget
Measurement Function	((NoD times DC) divided by PB) in percent
Indicator	Red/Yellow/Green
Analysis Model	Green if DM[1] < 1% Yellow if 3% > DM ≥ 1 % Red If DM ≥ 3%
Interpretation	If Red: Situation critical. Re-planning necessary. Inform steering group If Yellow: Take actions to avoid budget overrun and time plan delays If Green: No action

[1] Abbreviation for Derived Measurement

In this example the information need that the stakeholder, in this case the project manager, is concerned about is how much, compared to the project X[2] budget is the cost of defect reports.

The entity and measurable concept is the budget deviation. Attributes like project X defect reports [as a number]; Cost [in hours] of one defect report in project X and Budget [in hours] of project X are then chosen. These attributes are chosen out of experience by the developers of the measurement system (who usually have experienced as project managers). When having these attributes, a method is created to be able to measure the attribute, that is, convert the physical attribute to a numerical value to be used in mathematical calculations. The use of multiple measurements in a calculation results in obtaining derived measurements. In this example the measurement methods are: (a) count number of defect reports for X; (b) cost (in hours) of a defect report based on empirical experience and (c) budget (in hours) for project X. In this example, the indicators are set to green if the result value from the derived measurement is below 1%, yellow if between 1% and 3% and red if it is above 3%. These values are carefully selected out after a discussion with the stakeholders and based on experience from former and current projects.

An instance of this definition is presented in Table 2.

Table 2. Defect reports measurement system - instance

Concept	Definition
Values of measurements	4. NoD: 78 [defects] 5. DC: 3 [hours per defect report] 6. PB: 8000 [hours]
Derived Measurement (DM)	((58*3)/8000) * 100 % = 2,2 %
Indicator value	Yellow
Interpretation	Yellow: Project was slightly re-planned, more effort was put into solving defect before further development.

The example shows that even constructing simple indicators, one needs to be concerned with several measurements. Computing derived measures can require checking assumptions of measurements independence or dependence. The indicators are built based on these assumptions – slight deviations from established dependency relationships could make the indicators show false alarms or not indicating problematic situations. In the example above one such assumption is the cost of repairing one defect – if the cost is much lower than assumed, then this indicator would raise false alarms; if the cost is much higher than assumed, then the indicator would not inform in time about budget problems in the project.

4. CASE STUDY DESIGN

We performed our case study at Ericsson, a world-wide provider of telecom network equipment. The study was conducted at one of the quality management departments, working with measurements and measurement systems on a daily basis. The

[2] Project X can be compared to a real project at Ericsson however questions and values has been altered.

data which was used to evaluate the prototypes comes from several large software projects which the department is responsible for. The study is performed in a similar context as our previous studies (e.g. [25]).

The studied organization posed the following high-level requirements on the solutions which we should consider:

- The solutions should visualize large number of measurements
- The solutions should be able to compare dependencies between projects
- The solutions should use and/or integrate with existing toolset available at the company
- The solutions should follow the standards adopted by the company
- It should be possible to combine individual solutions into larger ones

The case study was divided into two parts – identification of viable visualization methods including elicitation of criteria for evaluating the methods, and evaluation of the visualization methods using data from historical and on-going projects at the studied unit at Ericsson. In short, our research process was:

1. Elicit criteria for comparing visualization methods – for assessing their applicability for the company.
2. Identify viable visualization methods via literature study.
3. Develop prototypes.
4. Evaluate prototypes on actual data from the company and through interviews.

The second author is working at the company and conducting both research and development in the area of software measurements. The results of the study are to be used in his work which makes this study an action research study. The third author is working closely with the company on the development of prototype measurements systems and evaluating them at the company. The first author spent the entire time of the study on site of the company.

4.1 Interviews

As the first step in the study we performed interviews with a designer of existing measurement systems, who is a quality manager with long term experience on working with measurements, project, and quality management at Ericsson. The purpose of an interview at the beginning of the study was to elicit criteria for assessing the usability of the tools. The goal of eliciting the criteria was to provide a basis for assessing the applicability of each prototype. By developing the criteria we also gained more knowledge of the non-functional requirements for each prototype. After eliciting the criteria the quality manager was asked to prioritize them using the $100 technique (in which a respondent is asked to distribute $100 for each prioritized element – larger amounts should got for the elements which are prioritized higher).

In the middle and by the end of the study we performed interviews with the same respondent, to evaluate the prototypes which were developed during this study. The criteria elicited from the interviews at the beginning were used to assess the prototypes.

The interviewer made notes during the interviews; the notes were used later during the study. All interviews were semi-structured as they contained both closed-ended, open-ended questions and the interviewee was allowed to make own remarks and comments.

4.2 Prototype development

After identifying the applicable visualization methods we created a set of prototypes to use these methods at the studied organization at Ericsson. In particular we developed a set of MS Excel add-ins using Visual Basic for Applications (VBA) that could parse the data, produce diagrams/charts, or export the data to other tools. One add-in was developed per visualization method.

We considered using dedicated visualization environments, but it was a strong requirement from the company to work with the toolset available and already adopted at the company. As a common ground, MS Excel 2003 was used in developing the prototypes as the used toolset at the company provided features to export data to MS Excel.

We used freeware mind mapping tools and hyperbolic browsers in more advanced visualization prototypes in order to test simple ways of presenting the information which MS Excel is not capable of.

The goal of developing the prototypes was to demonstrate the visualization methods and to provide our industrial partner with software to be used in their development of measurement systems.

4.3 Evaluation process

To evaluate the prototypes we used them on real data from on-going and past projects at the company. The results of running the prototypes on the data were shown to a quality manager who evaluated how the prototypes fulfilled the criteria.

The data from the ongoing project was a snapshot taken at the current time – this means that the data was not altered between evaluations of particular prototypes.

The weighted criteria are presented in Table 3.

Table 3. Evaluation Criteria

Criteria	Description	Weight
Usability (measurement systems developers)	It should be easy to use the prototypes, e.g. easy to fill in data, easy to start execution of macros, etc.	0.26
Time for execution	Execution should be performed very fast, that is, in less than a minute	0.24
Easy to overview and interpret results	It must be easy to overview and interpret results, e.g. tables, graphs, correlations, method.	0.12
Handle large sets of data	It should be possible to visualize dependencies in large data sets (e.g. more than 1000)	0.10
Comparing projects	Two different projects could be compared in the prototype showing how similar the dependencies are between the projects.	0.08
Parameters	It should be possible to use parameters to select a subset of measurements which are input to the add-in.	0.04
Maintainability	When prototype is finished, developers should be able to understand the concept and the code behind the prototype.	0.04
Magnitude of variables	If dependencies have different magnitudes (scale) it shall not affect the results	0.04
Strength of correlation	A strength of correlation should be calculated and shown in the resulting information	0.04
Usability (expert users)	The prototype could be used by other experts on measurements systems which has no prior experience in the current measurement system	0.04

During the interview the respondent was asked to assess to which degree the prototype fulfills the criteria using 5 point Likert scale: 1- totally unsatisfactory, 2- somewhat unsatisfactory, 3- neutral, 4- somewhat fulfills, 5- totally fulfills.

After the assessment we calculated the normalized score of the prototype. The normalized score was the product between the scores and the applicable criteria divided by the sum of weights of applicable criteria.

During the evaluation we recorded also qualitative comments from the respondents, including information how the prototype is supposed to contribute to the company.

5. RESULTS

By searching literature on visualization methods, we identified six viable visualization methods. Despite a significant body of research on visualization of source code, the methods were not applicable directly and required customization of the visualization tools, which in turn contradicting our requirements from the company.

In a series of interviews we evaluated the prototypes and identified their strengths and weaknesses. A summary of the interviews follows the results from the literature study.

5.1 Identified applicable visualization methods

Through literature study and the initial interview with the quality manager at Ericsson we identified two main ways of visualizing the dependencies (dependencies between measurements and dependencies between measured cases), grouped into three categories below.

5.1.1 Correlation visualization

The basic dependency between measurements is the correlation between two measurements. The correlation is an important indicator of dependency as correlated measurements should not be used when building predictive models. As the number of measurements collected in the organization was rather large, one could not be expected to manually run computations pair-wise. The rationale behind the developed prototypes was that they should support the users of measurements in their work by decreasing the time required to identify correlated variables.

As an extension to correlation visualization we also considered visualizing the results of Principal Component Analysis (PCA). PCA, however, had the disadvantage that it was hard to interpret and required visualization in more than three dimensions which was hard to obtain using available or freeware tools.

The most basic and well-known way of visualizing dependencies between two variables (measurements) is using scatter plots. If produced automatically for a set of variables, the scatter plots have an advantage that they could be used for more detailed examination of variables. An example scatter plot is presented in Figure 2.

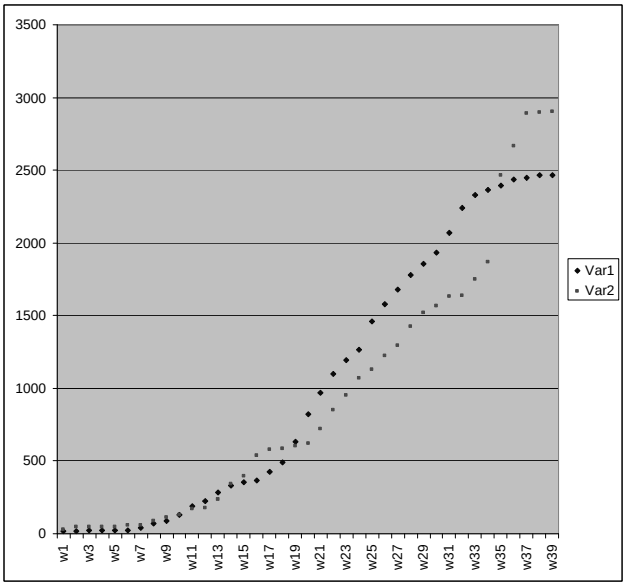

Figure 2. Visualization of dependencies using a scatter plot

The figure presents a scatter plot of two variables Var1 and Var2, which are correlated as the growing trends are observed for both variables.

Another way of visualizing the correlation between measurements is to use a matrix and a list containing colored cells with values of the correlation coefficient. An example is presented in Figure 3 as a matrix.

		Var1	Var2	Var3	Var4
1					
2	Var1		0,958656	0,746652	0,390364
3	Var2	0,958656		0,854176	0,287389
4	Var3	0,746652	0,854176		-0,18485
5	Var4	0,390364	0,287389	-0,18485	
6	Var5	0,989982	0,975872	0,767876	0,390388

Figure 3. Visualization of correlation coefficients – a matrix

Figure 4 shows a subset of the matrix as a list.

	Variable 1	Variable 2	Strength
2	Var1	Var2	0,958656297
3	Var1	Var3	0,746652468
4	Var1	Var4	0,390364068

Figure 4. Visualization of correlation coefficients – a list

These prototypes are intended to provide an overview of correlations within a single data set – e.g. measurements for one project.

When building measurement systems, however, examining a single data set is sometimes insufficient. In measurement systems measurements are used based on assumptions about their dependencies, which reflect the process followed by the company. The measurements, nevertheless, tend to change over time and hence the same measurement system might provide misleading information when used at two different projects if dependencies between variables are different.. Therefore a support is needed to check whether the dependencies between measurements in two projects are indeed the same. For this purpose we created the correlation differences prototype, which visualizes the differences in correlation coefficients between two sets of measurements. An example is presented in Figure 5. Once again the colors are used to emphasize the magnitude of differences. The colors are chosen as parameters of the prototype and therefore highlight differences important for the user.

	Pair	Difference	Sign differ
2	Var1 & Var2	0,0232875	
3	Var1 & Var4	0,0461366	
4	Var1 & Var3	0,0182627	
5	Var1 & Var5	0,0985069	
6	Var2 & Var1	0,0232875	
7	Var2 & Var4	0,0033687	
8	Var2 & Var3	0,0018969	
9	Var2 & Var5	0,1930474	
10	Var4 & Var1	0,0461366	
11	Var4 & Var2	0,0033687	

Figure 5. Visualization of differences of correlation coefficients

The result of running the prototype on two sets of data is a list of pairs of measurements and the differences between the correlation coefficients of the measurements in the pair in the two projects. The column labeled sign differ indicates whether there was a difference in the sign of the correlation coefficient (i.e. actual pair had the opposite behavior/trend in project B compared with project A?).

The difference between the correlation coefficient is to be interpreted manually based on the need. For example, when predicting quality of the project one uses regression equation which are built on one data set to predict quality using another data set. If the correlations between variables in these two data sets are significantly different, then the predictions might not be accurate. Therefore, significant differences between the correlations can be seen as an indicator of small accuracy.

Visualizing dependencies using a matrix or a list does not show a transitive dependencies – e.g. measurement A depending on B, B depending on C, etc. Therefore we developed the so-called X-Centric model prototypes using external freeware viewers: H3Viewer [26] and FreeMind [27]. This visualization method shows a network of dependent measurements, centered on a single measurement (Var1 in the example below). An example output from the FreeMind tool is presented in Figure 6. The numbers in the figure are correlation coefficients.

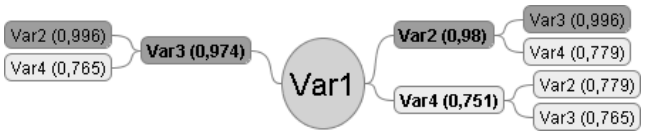

Figure 6. X-centric model visualization - FreeMind

The figure shows Var1 in the center and Var2, Var3, and Var4 which are correlated with Var1 with the strengths given in brackets – 0.98, 0.974, and 0.751 respectively. Var2 and Var4 (top left-hand corner) are correlated with Var3 with strengths given in brackets – 0.996, and 0.765 respectively.

An example visualization using the H3Viewer tool is presented in Figure 7.

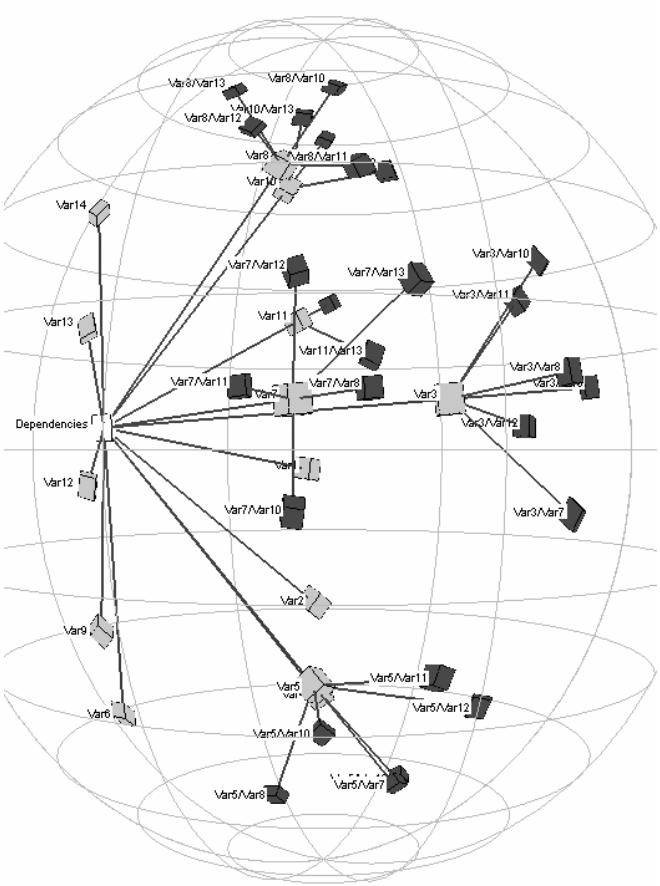

Figure 7. X-Centric model visualization – H3Viewer

Visualizing the transitive dependencies is used when building the measurement systems to identify measurements which can (if they are strongly correlated) be used interchangeably for some purposes (e.g. when building prediction models).

5.1.2 Distribution visualization

Visualizing correlations between variables shows whether the trends in the measurements are the same. The measurements, however, might be of different magnitude and/or distribution. The differences are important since the indicators in the measurement systems are built based on the values of measurements. The interpretation of indicators might depend on the distribution. Hence we developed a prototype to compare distributions. The prototype results in a bar chart with distributions of a pair of measurements and a p-value from the Chi-Square test for independence. The p-value denotes the probability that the two variables are indeed dependent.

In order to visualize the distributions between variables we used simple bar charts for graphical presentations and the chi-square test for independence to obtain the chi-square value and the probability of the measurements of being independent. An example is presented in Figure 8.

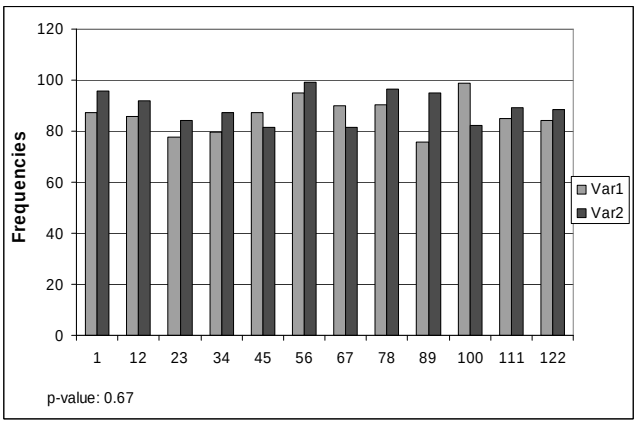

Figure 8. Visualization of distributions – bar chart showing frequencies for Var1 and Var2

The example shows that the distributions of the two variables are not different from each other, and that there is a significant probability that they are dependent on one another.

5.1.3 Case dependencies

Dependencies between the measurements provide only partial information. The information can be complemented by visualizing the dependencies between particular cases (or data subsets in the extended version of the prototype). The idea is that this comparison can identify two most similar vectors of measurements. The most similar cases to each other are believed to be dependent on each other. In this particular context we perceive this as a variation of analogy-based comparison – i.e. identifying similar cases by computing a distance between them. Analogy-based estimation has its foundation in project cost estimation [23, 28]. There, the elementary belief is that similar projects are probable to have the same behavior, for example estimated cost. In our case the rationale is that similar weeks (w.r.t. test effort) in two projects are probable to have the same characteristics (e.g. defect inflow). In analogy-based approach the estimations are derived from historical measurements. A distance function δ is calculated on l number of measurements. Weighting can be used to alter how much a measurement is supposed to affect the result. Scaling can be used if the two compared measurements are of different scale.

The distance is a weighted Euclidean distance δ, calculated using the formula:

$$\delta(p, p') = \sqrt{\sum_{i=1}^{l} w_i s_i^2 (d_i - d_i')^2}$$

Equation 1, Distance calculation for Analogy-based Estimation [21]

In Equation 1, δ stands for distance, p for points, w for weight, s for scale, d for value of a variable, while i is the index over measurement values for the data point.

The results of case dependencies is radar plot showing the most similar cases and the distance between them – an example is presented in Figure 9.

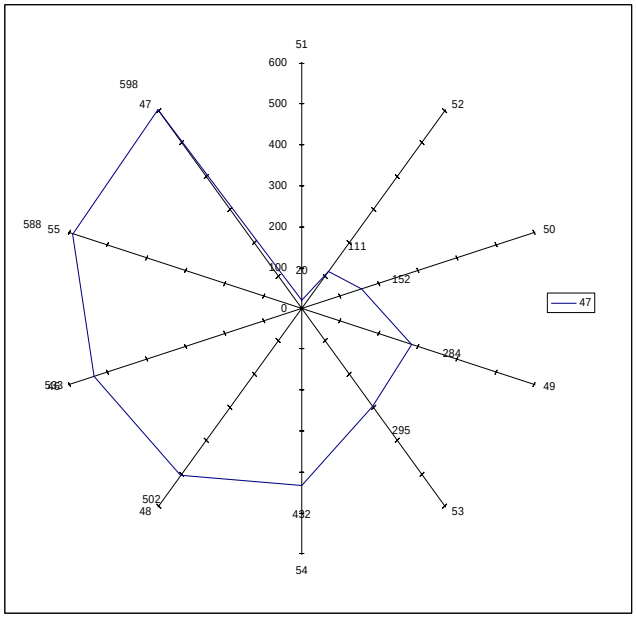

Figure 9. Visualization of case comparison – a radar plot

Figure 9 shows distances of 10 most similar cases to case 47. Each axis shows the distance between pairs of cases: case 47 and the case which is used as the name of the axis. In this example, the most similar case is case 51, as its distance to case 47 is shortest. The number of cases shown in the plot is an arbitrary number, which is a parameter in the prototype.

As an extension of comparing a single data point, the developed prototype provided a possibility to compare a series of data points and identify the most similar series in a reference data. Each data point from the series was then visualized separately using the radar plot. The similarity between the series was visualized using a colored list, as presented in Figure 10. The result is *δ/d*100%* using the symbols from Equation 1.

	Case Cmp	Case Original	Result
2	w15	w12	89%
3	w16	w13	18%
4	w17	w14	27%
5	w18	w15	22%
6	w19	w16	63%
7	w20	w17	69%
8			
9	w15	w14	81%
10	w16	w15	29%
11	w17	w16	92%
12	w18	w17	76%
13	w19	w18	95%
14	w20	w19	89%
15			
16	w15	w15	100%
17	w16	w16	78%
18	w17	w17	120%
19	w18	w18	88%
20	w19	w19	107%
21	w20	w20	100%

Figure 10. Visualization of similarities between series of data points

The example shows three series of data points (column Case Original) similar to the given series (column Case Cmp) and the differences as percentages (Result).

5.2 Evaluation of the methods

The evaluation of the methods is presented in two parts – the evaluation against the criteria and qualitative evaluation (including how the method is supposed to be used in the company).

5.2.1 Evaluation against criteria

The evaluation against the criteria is presented in Table 4. The evaluation was conducted by the quality manager. The visualization that was chosen as the best one is the X-centric view using mind-maps, although its maintainability was very low. The reason for the low value is the fact that the creation of mind-maps using the available tools could not be automated and required manual intervention every time new data points are added to the data set.

5.2.2 Qualitative evaluation

The qualitative evaluation is a summary of respondent comments recorded during the interviews when the quality manager evaluated the prototypes.

The scatter plot prototype could be used directly and for example be used at project meetings to show how measurements depend on one another. As an overview it could also be used to compare different projects which would be of use for project managers that test various changes to see how these would affect the dependencies. Today, such comparisons are not done, as the manual creation of so many plots is very time consuming.

When using the scatter plots on the real data at the company, the resulting scatter plots had one large disadvantage, namely the magnitude of the values. If two measurements had values in different scales, the scatter plot could result in that only one variable could be seen and the other variable would not be visible (due to the scaling of the plot itself). Despite this, if a basic knowledge around the dependencies exists among the stakeholders, the magnitude problem can be overseen and/or examined through the other prototypes, making this prototype a good starting point for identifying correlated measurements. .

The problem of different magnitudes of measurements in scatter plots is solved by using the correlation prototype. In the prototype another method for showing correlations was used, the Pearson's Product correlation coefficient. Using this coefficient the trends of the curves were compared while the magnitude was not crucial. Because of this, the prototype was easier to follow and interpret.

In MS Excel a list with results could easily be sorted given different criteria, which was a big benefit for the respondent. It allows easier searches in the data or shows only a subset of dependencies.

In the matrix result, a full overview of all dependencies could be seen. This gave the possibility to spot if some dependencies were

Table 4. Evaluation against the criteria (the highest score in boldface)

Criteria	Scatter plots	Correlation	X-centric (mind-map)	Correlation compare	Distribution	Analogy
Usability (developers)	5	5	5	5	5	5
Time for execution	5	5	5	5	5	5
Easy to overview and interpret results	5	4	5	4	2	4
Handle large sets of data	5	5	5	5	5	5
Comparing projects	N/A[1]	N/A	N/A	5	5	5
Parameters	5	N/A	5	N/A	5	5
Maintainability	3	2	2	3	2	3
Magnitude of variables	2	5	5	5	2	3
Strength of correlation	N/A	5	5	5	4	5
Usability (experts)	5	5	5	5	2	4
Normalized score	4.77	4.73	**4.87**	4.79	4.24	4.68

of exceptionally high or low correlation by examining the overview.

This prototype has the potential to improve the measurement systems being currently built at the studied organization at Ericsson.

H3Viewer was at an early stage rejected as a solution for modeling dependencies because of its low configurability. Strengths and correct colors for the dependencies could not be included. A hyperbolic browser was created and dependencies could be visualized, but due to the above limitations we did not include it in the evaluation.

FreeMind on the other hand, which used XML syntax with full configurability through the input file, was of good help. The clear overview with colors and correlation strengths gave a good overview of the network of dependencies. This could be used to easily and understandable show the dependency tree on how measurements were related.

One drawback of FreeMind was when more than two levels of dependencies were visualized. The resulting image spanned over a large area which was hard to get overview of when using computer screen.

Like the scatter plots, this prototype can be used to show an overview for the surrounding stakeholders during presentations. It is not certain, however, that the result will be used in the company to the same extent as the Pearson correlation.

When having a new project and a new measurement system is to be built upon the assumptions on older projects, this prototype could be used to see if the dependencies are the same in the two projects. The task of comparing projects is almost an impossible task to do by hand.

For project managers the prototype and the method could be used to track changes in the project progress/behavior compared to past experiences. When a change is introduced, a new project could be compared to older projects to see if the changes had any affect on the measurements. In this way the experts get a support in answering the question if the measurements measure the same things in the same way in the new project as in the old projects.

This prototype will, as Pearson correlation, also be useful for the company. It will be integrated in the core of the measurement systems. This prototype makes it possible for comparison of large sets of data and gives an accurate result. Today, to do this kind of comparison by hand is not possible due to the time it would take.

The comparison of the distribution of values shows how distributions of two variables could be related to each other as they have similar distributions.

This method uses the Chi-Square test for independence to obtain the p-value. During the evaluations the Chi-Square was shown not to work perfectly on the real data sets since the distributions differ too much to be compared with the Chi-Square, at least to give a meaningful result. The implementation of Chi-Square has also a limitation that it can't be computed if zero exists in the expected range. This affected the frequency table to be altered accordingly to the excepted range of values since it had to be re-configured in a way that all frequencies had at least one value. This altered frequency table gave some kind of a manipulated result which was not sufficiently good.

The magnitude of variables was also a problem. Large projects could not be compared to smaller projects since this would affect the outcome of the distribution table. In this case the measurements need to be standardized first. The magnitude of differences, however, was found to be important for the company.

Since the frequency table had to be altered to avoid the division by zero, the result could not be relied on and was difficult to interpret; hence the prototype will not be used.

The analogy-based comparison prototype had features for scaling projects to avoid magnitude problems which were found to be useful. It will be used to find matching groups of weeks in different projects to identify the most similar weeks. One drawback with the prototype is that it could be hard to overview when comparing a large number of weeks.

A particularly useful feature was the comparison between series of cases, which could help the experts to identify a series of similar data points (e.g. weeks close to finishing the project) and the similarity between them.

The prototype will be used by the developers and the analysts of the measurements systems. It will be used to compare groups of weeks to adjust the measurement systems, if needed, and could also be used to find similarities in projects. As the Pearson's correlation coefficient, this method will also be a useful for the company.

6. Validity evaluation

As every empirical research, our case study exposes some threats to validity. The validity evaluation follows the framework presented in [29].

The main *external validity* threat of the results is that this case study was performed at a single company, at one of its organizations. Even though the company cannot be regarded representative for all software industry, the context of this study is general. The evaluation criteria, however, have not been generalized to other organizations than the studied one. We are currently collecting more data from the use of measurement systems in order to increase the external validity of these results.

The *internal validity* threat, which seems to be the most influential, is the fact that the study was performed on a "static" data set – i.e. a snapshot of the data at a current time in the study. This was dictated by the time frame of the study. We intend to further evaluate the prototypes after they are integrated with the measurement systems developed at the company.

The main *construct validity* threat is that we developed the evaluation framework as part of this study. This might bias the results as there is a danger that the framework is not complete. In order to minimize this threat we took two measures: (a) developing the framework before developing the prototypes, and (b) recording the interview data to identify additional evaluation criteria (which did not happen).

7. CONCLUSIONS

Working with large number of measurements is a characteristic of large and mature organizations. As the maturity of the organizations increases the organizations seek improvements in their processes, optimizations, and better control. This leads to using more sophisticated methods for working with data being collected. In this study we evaluated several basic methods for identifying, quantifying, and visualizing dependencies between

measurements. The identified methods were evaluated empirically on data from large software projects and through a series of interviews with the quality manager working with measurements.

During the study we identified a set of criteria used to evaluate the methods. The criteria reflect the main requirements from the organization on the toolset used to work with measurements.

The results show that these simple methods are indeed very useful in working with large number of measurements as they allow identifying dependencies very efficiently. Using the evaluation criteria resulted in identifying mind maps as the best visualization method. Qualitative analysis showed that the expert found visualization of correlations between large data sets to be useful method in his work.

Our further work is focused on integrating the presented prototypes into measurement systems used at the studied Ericsson's organization.

ACKNOWLEDGMENTS

The authors would like to thank Ericsson Software Research and Software Architecture Quality Center for their support in the study. We would also like to thank managers at Ericsson who made this work possible and supported us – thank you!

REFERENCES

1. International Standard Organization and International Electrotechnical Commission, Software engineering – Software measurement process, ISO/IEC, Editor. 2002, ISO/IEC: Geneva.

2. International Standard Organization and I.E. Commission, Software engineering – Product quality Part: 1 Quality model, ISO/IEC, Editor. 2001: Geneva.

3. Fenton, N.E. and S.L. Pfleeger, Software metrics : a rigorous and practical approach. 2nd ed. 1996, London: International Thomson Computer Press. XII, 638 s.

4. Clark, B., Eight Secrets of Software Measurement, in IEEE Software. 2002. p. 12-14.

5. Kilpi, T., Implementing a Software Metrics Program at Nokia, in IEEE Software. 2001. p. 72-77.

6. Dekkers, C.A. and P.A. McQuaid, The Dangers of Using Software Metrics to (Mis) Manage in IT Professional. 2002. p. 24-30.

7. Pfleeger, S.L., et al., Status Report on Software Measurement, in IEEE Software. 1997. p. 33-34.

8. Brökcers, A., C. Differding, and G. Threin. The Role of Software Process Modelling in Planning Industrial Measurement Programs. in METRICS. 1996: IEEE.

9. Walpole, R.E., Probability & statistics for engineers & scientists. 7th ed. 2002, Upper Saddle River, NJ: Prentice Hall. xvi, 730 p.

10. Anderson, T.W., An introduction to multivariate statistical analysis. 3rd ed. Wiley series in probability and statistics. 2003, Hoboken, N.J.: Wiley-Interscience. xx, 721 p.

11. Alfert, K., F. Engelen, and A. Fronk, Experiences in three-dimensional visualization of java class relations. SDPS Journal of Design & Process Science, 2001. 5(3): p. 91-106.

12. Alfert, K. and A. Fronk. Manipulation of three-dimensional visualization of java class relations. in The Sixth World Conference on Integrated Design & Process Technology,. 2002.

13. Hendrix, D., J.H.C. II, and S. Maghsoodloo, The effectiveness of control structure diagrams in source code comprehension activities. IEEE Transactions on Software Engineering, 2002. 28(5): p. 463-477.

14. Voinea, L. and A. Telea, Visual data mining and analysis of software repositories. Computers & Graphics, 2007. 31(3): p. 410-428.

15. Kuhn, A., S. Ducasse, and T. Girba, Semantic clustering: Identifying topics in source code. Information and Software Technology, 2007. 49(3): p. 230-243.

16. Umphress, D.A., et al., Software visualizations for improving and measuring the comprehensibility of source code. Science of Computer Programming, 2006. 60(2): p. 121-133.

17. Noser, H. and P. Stucki. Dynamic 3D visualization of database-defined tree structures on the WWW by using rewriting systems. in Advanced Issues of E-Commerce and Web-Based Information Systems, 2000. WECWIS 2000. Second International Workshop on. 2000.

18. Hing-Yan, L., et al. A multi-dimensional data visualization tool for knowledge discovery in databases. in Computer Software and Applications Conference, 1995. COMPSAC 95. Proceedings., Nineteenth Annual International. 1995.

19. Spence, R., Information visualization: design for interaction. 2nd ed. 2007, New York: Addison Wesley.

20. Abdi, H., A Neural Network Primer. Journal of Biological Systems, 1994. 2(3): p. 247-283.

21. Auer, M., et al. Implicit analogy-based cost estimation using textual use case similarities. in International Conference on Intelligent Computing and Information Systems. 2005. Cairo.

22. Bode, J., Decision support with neural networks in the management of research and development: Concepts and application to cost estimation. Information and Management, 1998. 34(1): p. 33-40.

23. Sheppard, M. and C. Schofield, Estimating software project effort using analogies. IEEE Transactions on Software Engineering, 1997. 23(12): p. 736-743.

24. Traina, C., Jr., et al., Fast indexing and visualization of metric data sets using slim-trees. Knowledge and Data Engineering, IEEE Transactions on, 2002. 14(2): p. 244-260.

25. Staron, M. and W. Meding. Short-term Defect Inflow Prediction in Large Software Project - An Initial Evaluation. in International Conference on Empirical Assessment in Software Engineering (EASE). 2007. Keele, UK: British Computer Society.

26. Munzner, T., H3Viewer. 2001, Stanford University.

27. Freemind, FreeMind - free mind mapping software. 2007, Sourceforge.

28. Huang, S.-J. and N.-H. Chiu, Optimization of analogy weights by genetic algorithm for software effort estimation. Information and Software Technology, 2006. 48(11): p. 1034-1045.

29. Wohlin, C., et al., Experimentation in Software Engineering: An Introduction. 2000, Boston MA: Kluwer Academic Publisher.