Towards Reusable Measurement Patterns

Mikael Lindvall[2], Paolo Donzelli[1], Sima Asgari[1], Vic Basili[1,2]
{donzelli,basili,sima}@cs.umd.edu
mlindvall@fc-md.umd.edu

[1]*Computer Science Department - University of Maryland*
College Park, 20742 MD, USA
[2]*Fraunhofer Center for Experimental Software Engineering,*
College Park, 20742 MD, USA

Abstract

Software measurement programs can help organizations make better decisions regarding their software projects. However, creating and establishing software measurement programs can be both costly and difficult. This paper addresses the problem by focusing on reusability of metrics for software measurement programs through the identification of measurement patterns. We illustrate our work with identifying measurement patterns by providing an extensive and detailed measurement example that is broken down into interdependent building blocks and activities.

1. Introduction

Creating software measurement programs from scratch requires a great deal of time and is too costly for most organizations. We believe identifying and defining measurement patterns is the key in order to reduce both time and cost because measurement patterns will make it easier for organizations to develop their own measurement programs without having to start from scratch each time. In this paper we describe our work with identifying and defining goal-based measurement patterns.

The patterns are based on the Goal Question Metrics approach (GQM) [1,2]. GQM is commonly used to define measurement programs [2], however we have identified that GQM is relatively open for interpretation and is often used differently depending on who applies it. The drawbacks are that not only it is harder than necessary for someone not used to GQM to apply it, but it is also difficult to reuse and compare measurement programs, and it is difficult to aggregate and generalize measurement results.

In order to illustrate the work on identifying patterns, we provide a detailed and extensive example of a GQM application in which we systematically identify the emerging pattern components and their interdependencies. Once the building blocks are identified, a general pattern emerges that can be reused to address similar goals.

The paper is organized as follows. Section 2 briefly describes GQM and introduces a goal hierarchy that can support pattern building and selection. Section 3 introduces the general idea of measurement patterns, borrowing from the software design domain. Section 4 shows how a measurement pattern can be identified through an extensive example. Finally, conclusions are given in Section 5.

2. GQM and relationship among GQM goals

GQM is a systematic approach for building, tailoring, and selecting models of and metrics for software processes, products and quality properties in order to address specific goals of software projects of an organization.

In GQM, a goal is operationalized by refining it into a set of quantifiable questions that are used to identify which data need to be collected to support the decision-making process. The required data provide guidance in building and selecting appropriate metrics and models. In addition, the questions provide a framework for interpretation of the collected data.

Goals may be defined for any object, for a variety of reasons, with respect to these quality attributes, from various points of view, relative to any particular environment. A GQM goal is thus defined by filling in a set of values for various parameters in a template: The *study object*, the *purpose*, the *quality focus*, the *point of view*, and the *context*.

Typically, the GQM process is described as Goals generate Questions, and Questions generate Metrics. We have, however, discovered that questions from

higher-level goals do not naturally generate metrics; rather they lead to the identification of lower-level goals.

As a matter of fact, GQM goals can be naturally seen as "acting" at different levels of complexity, with higher-level goals that may generate *lower-level* goals as part of the goal definition process. The goal attribute that determines its level of sophistication is the *purpose,* which can have five different values:

1. Characterize. This is the most basic measurement purpose; it involves describing and differentiating software processes and products
2. Understand. This measurement purpose mainly involves explaining associations, dependencies, and casual relationships between processes and products
3. Evaluate. Evaluation involves assessing the achievement of software project goals (e.g. reaching a certain quality level, being more productive, producing to less cost, conforming to a defined process), or the impact of a technology/process on products for some goals. It usually involves characterizing or understanding a situation.
4. Predict. Prediction is similar to evaluation, but slightly more sophisticated. While evaluation typically uses a model based on the characteristics of one set of objects, prediction builds a model based on the characteristics and correlation of data-pairs distributed over two sets of objects.
5. Improve. This is the most sophisticated measurement purpose; it usually involves evaluating or predicting in order to identify the actions necessary in order to improve a process or a product.

3. Measurement patterns

We borrowed the idea of measurement patterns from design patterns [3]. Design patterns methodically name, explain, and evaluate important and frequent designs in object-oriented systems. Design patterns solve specific design problems and make object-oriented designs more flexible and elegant, and ultimately reusable. They help designers reuse successful designs by basing new designs on prior experience. A designer who is familiar with such patterns can apply them immediately to design problems without having to rediscover them. In each system, several design patterns can be used, and they can coexist with each other [3].

Our approach is to apply the idea of patterns to measurement in general and to GQM in particular. Thus, we say that measurement patterns solve specific measurement problems and make solutions reusable. They help analysts quickly identify and deploy solutions to address similar problems already addressed in the same organization or in different organizations. Measurement patters greatly support knowledge packaging and transfer within and between organizations in the same way as design patterns do.

In the following, we develop a detailed example to show how a measurement pattern can be identified while developing a measurement program. Then we will show how the same pattern can support design of similar solutions. In other terms, once the analyst has formalized the GQM goal, the analyst can use the pattern to rapidly identify the metrics to be collected.

4. Building a Measurement Pattern

In this section we illustrate how to identify measurement patterns. By developing a complete (and typical) GQM application, we highlight how GQM questions from higher-level goals may lead to lower level goals (and not to metrics directly), and show how the emerging goal structure can be turned into a reusable measurement pattern.

Consider the following situation: *The quality manager of an organization that develops software has decided that its customers are reporting too many failures and that most of these problems should have been caught during system test. In order to reduce the number of failures, the quality manager is considering adopting a new testing process. To make an informed decision, the quality manager first wants to apply the new test process to a pilot project and to compare the performance of the new process with the performance of the current test process through measurement.*

In GQM terms, this leads to a goal structured as follows:

- The *object of study* is the new testing process because the quality manager needs to decide whether or not it should replace the current process.
- The *purpose* is "evaluation" because it involves comparing two or more objects in order to rank-order them according to some criteria. The quality manager determines that the evaluation should be in comparison with historical data even though he does not know yet how much historical data actually exist.
- The *quality focus* is the performance of the new process because the final decision is

going to depend on whether the performance of the new process is better than the current process.

- The *point of view* is the quality manager because it is the one interested in the performance of the new process when applied to a representative project as compared to similar projects conducted in the past.
- The *context* is the software development unit (the environment in which the process is studied) and the assumptions (the requirements for which we say that the process is used in a correct way) are that the new testing process is followed and that the team that applies the new process has a good understanding of the domain.

In order to achieve this goal, on the basis of his experience, the quality manager recognizes that there are three main issues he needs to focus on. These can be synthesized in the following three questions:

- Q1: Is the domain understood? One needs to make sure that the people who applied the new testing process are indeed a representative sample of the people in his organization as regards domain understanding. A process conducted by an experienced (or inexperienced) team does not produce the same results as a process conducted by an average team.
- Q2: Is the process applied correctly? In order to make a decision regarding the performance of a process, it is not enough to state that the new process is followed; one needs to make sure that the applied process does indeed follow the prescription for the new process.
- Q3: Is the performance of the new process better? In order to evaluate the performance of the process, one needs to create an evaluation model to which the performance of the new process can be compared. In this case, the model will be based on the performance of the current process.

These three questions that typically occur in the application of GQM [2] will usually lead to a sub-set of more focused questions, refining and enriching the initial ones. In other terms, we could say that the initial three questions identify main areas to focus on. We claim that this process can be made more efficient if we recognize that each of these questions leads to a sub-ordinate goal: a more elementary GQM goal necessary to achieve the initial one.

Thus, the question Q1 "Is the domain understood?" indicates that it is necessary to *evaluate* the level of domain understanding of the team applying the new test process. This leads to the first derived goal, as illustrated in Figure 1, and forms a new evaluation goal: the *study object* is the team; the *purpose* is evaluation; the *quality factor* is domain understanding; the *point of view* is the quality manager and the *context* is the software development unit.

Similarly, the question Q2 "Is the process applied correctly?" leads the second derived *evaluation* goal (Figure 1). The study object is the new process because one needs to determine whether or not it was actually applied. This is an evaluation goal: the *study object* is the new testing process; the *purpose* is evaluation because it is necessary to compare the actual process to the prescribed process; the *quality focus* is process conformance; the *point of view* is the quality manager and the *context* is the software development unit.

Finally, the question Q3 "Is the new process better?" leads to the third derived goal (Figure 1). This is also an *evaluation* goal: the *study object* is the new testing process; the *purpose* is evaluation because it is necessary to compare the new testing process to the old one; the *quality focus* is process performance; the *point of view* is the quality manager and the *context* is the software development unit.

The result of this analysis is show in Figure 1. In this way we have seen how GQM goals may in fact generate lower level goals. Having this set of new goals, our analysis has to focus on these.

A closer look at these evaluation goals (Figure 1), reveals that each of them lead to a set of characterization activities. For example, in order to evaluate the level of domain understanding among the team in the pilot project, one needs to characterize domain understanding. Similarly, in order to evaluate whether the applied process conforms to the prescribed process, one needs to describe the process' main characteristics, which is called characterization using GQM terminology. Then, to evaluate the performance of the new testing process, one needs to characterize process performance. In addition, once the actual and prescribed process, the common and the pilot teams' domain understanding, and the common and pilot team's testing process performance have been characterized, each of them has to be evaluated. In other terms, each of these evaluations requires an evaluation model.

For evaluating domain understanding, a model that compares the pilot teams' domain understanding

with the common domain understanding has to be developed. For evaluating process conformance, a model has to be developed to compare the actual process characteristics with the prescribed, and to determine whether the process was conforming or not. For evaluating process performance, a model has to be developed to compare the pilot process' performance with the common performance.

As illustrated in Figure 1, therefore, each evaluation goal leads to a GQM characterization goal, from which a characterization model will be obtained (e.g. to characterize process performance), and to an evaluation formula, to compare the results of the applications of the characterization model (e.g., to compare the performance of the new process against the performance of the old one).

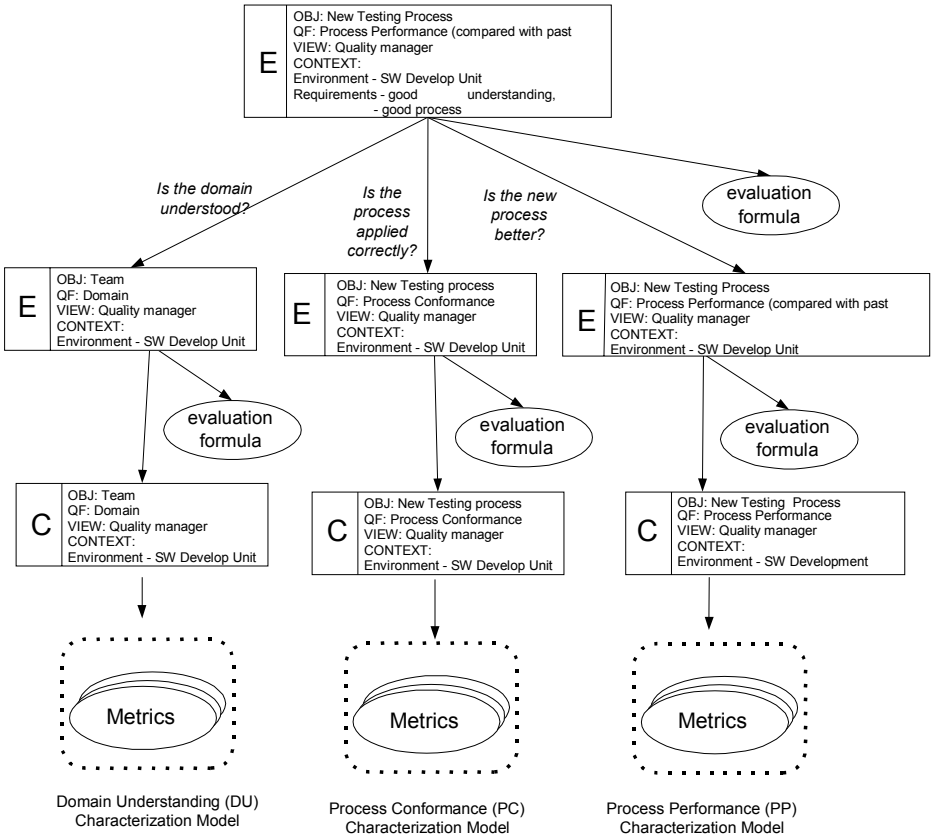

Figure 1. The breakdown of the overall GQM goal into smaller sub goals

The emerging characterization goals constitute the most basic GQM goals and will thus not lead to other goals. The characterization goals will instead lead to a set of metrics, according to the classical Goal-Question-Metric process.

The evaluation formulas, of course, are highly dependent on these questions and metrics and need to be defined contextually. In the following tables (from Table 1 to Table 3) we report the set of questions and the underlying hypothesis that lead to the identification of a specific set of metrics (characterization model) for each characterization goal and to the corresponding examples of evaluation formula:

- Table 1 shows the questions (Q) and the hypothesis leading to the metrics (M) representing the Domain Understanding characterization model (DU), and the corresponding evaluation formula;
- Table 2 shows the questions (Q) and the hypothesis leading to the metrics (M) representing the Process Conformance characterization model (PC), and the corresponding evaluation formula;
- Table 3 shows the questions (Q) and the hypothesis leading to the metrics (M) representing the Process Performance characterization model (PP), and the corresponding evaluation formula.

Table 1. Questions, Metrics, Hypothesis, and Evaluation Formula for Domain Understanding

Q1 - What is the experience of each single member of the team?
M11 – Questionnaire for each team member to complete:
How familiar are you with the problem domain?
0 – The problem domain is new to me
1 – I have attended a course on the problem domain
2 – I have been working in this domain for 2 or more years
Hypothesis: Work experience is better than having had a course on the subject

Q2 - What is the experience of the team?
M21 – Percentage of team members with M11=0
M22 – Percentage of team members with M11=1
M23 – Percentage of team members with M11=2

Hypothesis: Team experience is related to team member experience

Evaluation Formula (example)
Hypothesis: No more than 30% of the team members can be novices
Formula: If (M21/(M21+M22+M23) ≤ 30%) then Domain Understanding is sufficient (DU=Yes) else (DU = No)

Table 2. Questions, Metrics, Hypothesis, and Evaluation Formula for Process Conformance

Q1 – Which are the main steps that describe the process?
M11 – Main process steps
Hypothesis: It is possible to describe the process through main steps

Q2 - Did each member of the team follow the process?
M21 – Questionnaire for each team member:
Did you follow the process steps described by M11?
0 – no
1 – yes, but only partly
2 – yes, all of them

Q3 - Did the team follow the process?
M31 – Percentage of team members with M21=0
M32 – Percentage of team members with M21=1
M33 – Percentage of team members with M21=2

Evaluation Formula (example)
Hypothesis: At least 80% of the team should have followed (at least partly) the process
Formula: If (M31/(M31+M32+M33)≤ 20%) then Process Conformance is sufficient (PC=Yes) Else (PC = No)

Table 3. Questions, Metrics, Hypothesis, and Evaluation Formula for Process Performance

Q1 – What is the percentage of defects found by the testing process with respect to the total of the defects found during testing and during the α and β releases?
M11 – Defects found during testing
M12 – Defects found during α release
M13 – Defects found during β release

Q2 - What is the average cost for finding a defect during the testing?
M21 – Cost of testing (USD)
(M11 – Defects found during testing)

Q3 - What is the average time for finding a defect during testing?
M31 - Testing time
(M11 – Defects found during testing)

Evaluation Formula (example)
Hypothesis
New Testing Process P significantly better than current process if
It finds 30% more defects
The average cost is 30% lower
The Average time is 10% lower
Formula:
If (M11/(M11+M12+M13))[P] / (M11/(M11+M12+M13))[Historical Data] >= 1.3 AND
(M21/M11)[P] / (M21/M11)[Historical Data] <= 0.7 AND
(M31/M11)[P] / (M31/M11)[Historical Data] <= 0.9 Then (QF = Yes) Else (QF=No)

Once all metrics have been defined, the quality manager starts measuring. Activities involve applying the characterization model in order to characterize the team's level of domain understanding and applying the evaluation model in order to determine whether the domain understanding is indeed acceptable. In order to characterize the level of domain understanding, he hands out a survey to each of the team's members. The survey has the questions defined in table 1. In order to evaluate the team's level of domain understanding, he tallies the results from the surveys and compares the aggregated result to the evaluation formula. If less than 30% of the team members are novices then the team is considered having sufficient domain understanding, otherwise not. The result from this evaluation serves as input to the top-level evaluation goal. Figure 2 provides the dynamic view of this process. In this figure, hexagons represent activities as part of achieving measurement goals. Dashed arrows represent how output from the determination of one measurement goal serves as input to another.

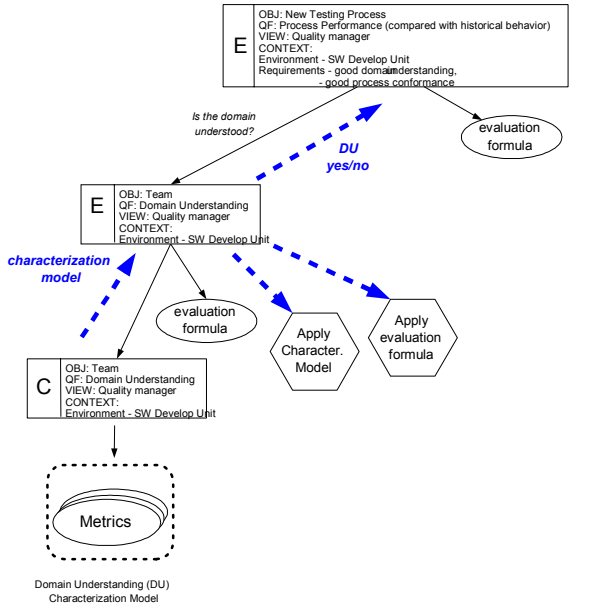

Figure 2. The dynamic view of the GQM application – Domain Understanding

A similar procedure is conducted in order to determine the level of process conformance in accordance with Table 3. Process performance involves more steps because it takes historical performance into account. The characterization model is first applied to the historical data. In essence, a model of the historical performance based on historical data has to be developed. The simplest possible model is one that is based on the average number of defects found in testing as compared to the average number of defects found by users of α and β versions of the software. More sophisticated statistical model can be developed, but is outside of the scope of this paper. When a model of the historical performance has been developed, the characterization model is applied to the pilot project. This means that the number of defects found in testing is compared to how many defects were found by the pilot's α and β users. When all the data is collected, the evaluation formula can be applied and the quality manager can determine whether the result was better or worse than historical performance. This results serves as input to the top-level evaluation goal. Figure 3 shows the dynamic view of this process.

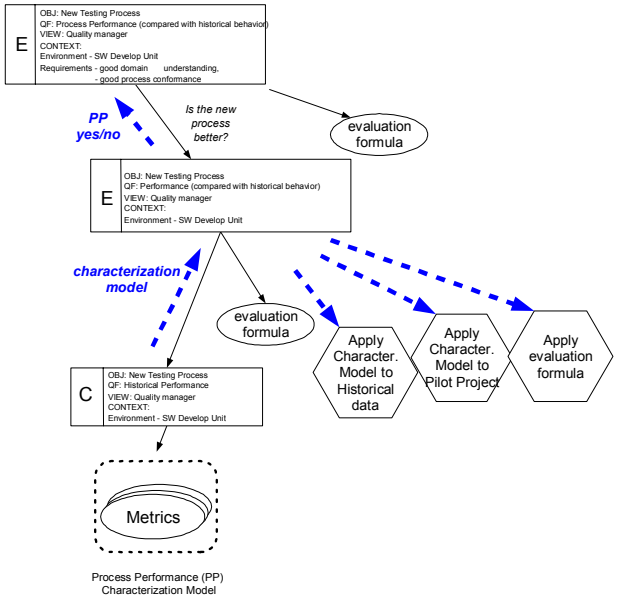

Figure 3. The dynamic view of the GQM application – Process Performance

When all results are fed back to the top-level goal, it is time to apply the top-level evaluation formula. All the results can be compiled into a table with a row for each of the quality factor (QF), the process conformance (PC) and domain understanding (DU). Different combinations of results are possible, but only when process conformance and domain understanding are both acceptable can the quality manager determine whether the new process is indeed

better than historically. Figure 4 illustrates the whole process including the static and dynamic activities.

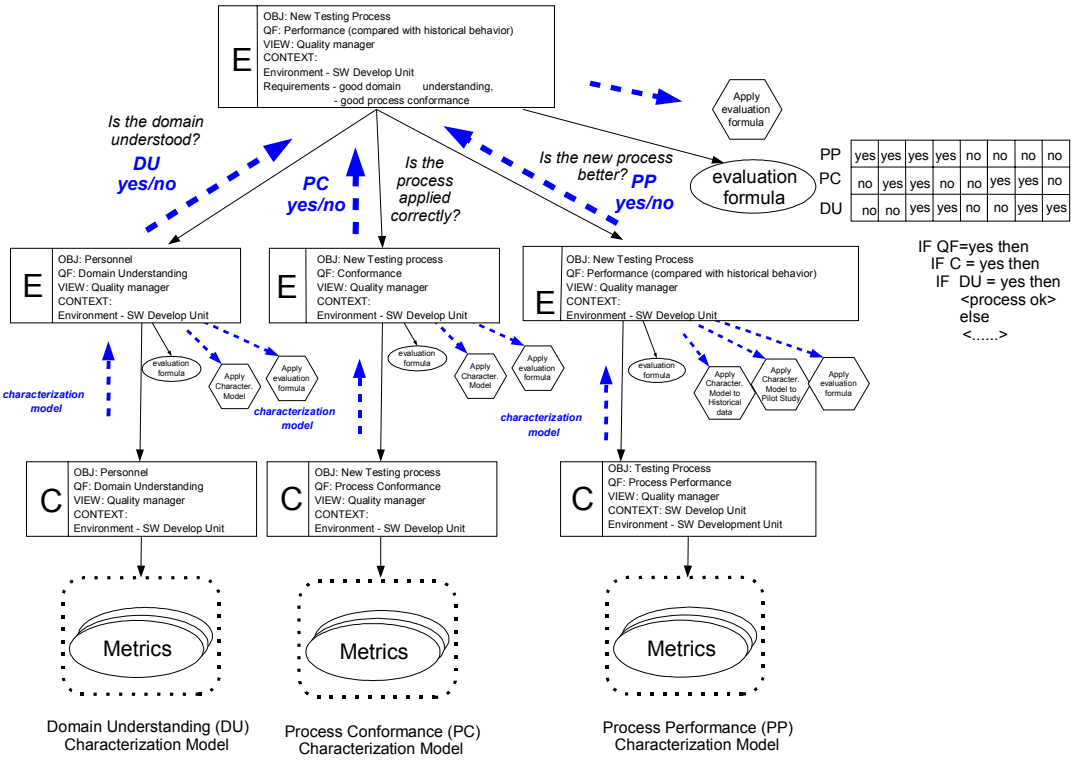

Figure 4. Full view of GQM application

In the previous example, we have seen how a top-level process evaluation goal was systematically broken down into a set of evaluation sub goals, and characterization sub goals, evaluation formulas, activities, and final decision table, as well as the flow (dynamics) between the entities. We believe this is a pattern that is commonly occurring and that the process can be generalized into a general pattern that can be reused in different contexts. Figure 4 illustrate the general pattern and indicate the different steps that are necessary in order to define and apply an evaluation of a process. Also notice that some of the information in the top-level goal is inherited (in bold) to the sub and sub sub goals. We believe there are similar processes for other objects such as software artifacts, as well as for other purposes, such as prediction and understanding.

5. Concluding Remarks

In order to avoid starting each measurement program from scratch, we have started the identification of measurement patterns that will help us define high-level patterns that can be applied in different contexts. We illustrated with an extensive example how the work of identifying patterns is conducted. The example we used, the evaluation of a new process, is commonly occurring and we expect to be able to reuse this pattern for similar situations. By identifying all the steps, activities, and building blocks that are necessary to empirically evaluate a new process, we described all the concepts necessary to define measurement patterns based on GQM. As new patterns are identified, they will be stored in an experience base together with lessons learned and frequently asked questions to allow for efficient implementation.

Measurement patterns do not only allow reducing a complex problem into a set of smaller ones (sub-ordinate more elementary GQM goals), but they also greatly improve reuse of available knowledge in a more focused way. Detailed Questions, Hypotheses, Formulas and Metrics developed for elementary GQM goals (e.g., characterization goals) can be more easily re-applied for two different reasons: first, because the analyst will be dealing with similar goals (characterization), then because the specific

characterization goals are in a similar context (e.g., evaluation goal).

6. References

[1] V. Basili, G. Caldiera and D. H. Rombach, The Goal Question Metric Paradigm, Encyclopedia of Software Engineering - 2 Volume Set, pp 528-532, John Wiley & Sons, Inc., 1994.

[2] R. van Solingen and E. Berghout, The Goal/Question/ Metric Method, McGraw-Hill, New York, 1999.

[3] E.Gamma, R.Helm, R. Johnson, and J.Vlissides, Design Patterns Elements of Reusable Object-Oriented Software. Addison Wesley, 1994.

Acknowledgements

The authors wish to thank Jennifer Dix for proof-reading this paper.