Measurement based quality control for real-time system specification

H. Kempter, H. Frank

Daimler-Benz AG, Research and Technology,
Wilhelm-Runge-Str. 11, 89081 Ulm, Germany

ABSTRACT

The ideas presented and discussed in this paper focus on the measurement based evaluation and improvement of the requirements model with regard to efficiency aspects in real-time systems. The requirements model as the result of the systems analysis based on the SA/RT-method (Structured Analysis and Real-Time extensions) is of significance for the software development in several fields of our companies, especially in car electronic, aviation, and robotics systems. First, this model serves as communication interface between the customer and the software manufacturer and forms the basis for the contract. Second, it serves as reference model and starting point for the software development and maintenance process. In summary, this model affects the quality of the software product, especially the fulfilment of efficiency as one of the most important quality criteria in real-time systems.

This paper describes a quantitative approach to software quality assurance with respect to efficiency in systems analysis. The content of this paper consists of two components. First, an experienced based guideline for structuring a goal-oriented quality model with exemplary results from a project within the software development for robotics systems, and second, a formal method for simultaneous specification of efficiency requirements in the context of the systems analysis method SA/RT.

INTRODUCTION

Motivation

In the development of real-time critical, software-based systems we are inevitably faced with the problem of fulfilling expected time and storage constraints, also called efficiency requirements. Traditionally we deal with
them in the implementation phase, but this is often too late. The defects and problems related to efficiency which are detected in the implementation phase are mostly caused in previous phases and therefore produce unnecessarily very high removal costs. A further problem in this context is to estimate in the early phases of the life cycle whether the efficiency requirements can be fulfilled. This is particularly important for the requirements document which serves frequently as basis for the contract between the software manufacturer and the customer.

Based on these existing problems the task of a quality oriented software organisation is to evaluate and improve the quality of documents developed in the early phases of the software life cycle, especially regarding the quality criterion efficiency, as well as to make appraisals of their fulfilment in the software product.

In this context we look at the requirements model created with the SA/RT-method (Hatley [1]). This model is a very important point of reference in the software development of real-time critical systems not least because this systems analysis method is used in our companies, especially in the project from which this paper arises. Additional to the use of SA/RT the software life cycle in this project context is supported by the employment of the CASE-Tool Teamwork (Cadre [2], [3]).

In the representing of requirements by SA/RT there is a essential weakness. The systems analysis based on SA/RT offers only the possibility to represent functional requirements. Against that, non-functional requirements as timing and storage constraints can not be represented in the SA/RT-model. If efficiency requirements are not considered in the systems analysis, the negative impact appears in the late development phases. In consequence, missing or inconsistent efficiency requirements involve the software developers in expense and high economic risk.

The purpose of the project was to develop methods for measuring and evaluating interesting aspects related to the assurance of efficiency in the requirements model. This methods ought to be integrated in a software quality model and serve for the systems analyst as means to evaluate and appraise efficiency requirements. To support a quantitative and measurement based evaluation of this quality goal a SA/RT-consistent description language for the specification of efficiency requirements is provided.

Quality Control Cycle

The approach for quality improvement in the systems analysis related to efficiency consists of the following components:

- A formal method to specify efficiency requirements in application of SA/RT. The specified efficiency requirements are named $E$-Spec and form an extension of SA/RT.
A goal-oriented software quality model, in which the interesting quality aspects for the evaluation and assurance related to efficiency in the context of systems analysis are specified.

The relationship between the main components in this quality improvement approach as well as the proceeding steps are described in Figure 1.

A typical scenery of quality-oriented proceeding in systems analysis consists of three steps which are gradually applied to achieve high quality:

- In the systems analysis based on SA/RT, the efficiency requirements are simultaneously specified by using our provided specification formalism. We call the integration of E-Specs in SA/RT an extended SA/RT-model.
- After completion of a first version of the extended SA/RT-model, required and in the quality model defined metrics are measured.
- Based on the quality model and the measured metrics the fulfilment of the defined quality goal related to the extended SA/RT-model is evaluated; measures to improve the quality in the system specification are derived and realised.

STRUCTURING THE SOFTWARE QUALITY MODEL

Introduction

In the construction of the software quality model we used several techniques and methods, especially components from the Goal-Question-Metric (GQM) paradigm (Rombach [4], [5], Basili [6]) and the Goal-tree method. The Goal-tree method is an appropriate means we use to support the GQM-paradigm by refining the abstract quality goal into more comprehensible subgoals.
The purpose of this paper is not to present every detail of the achieved software quality model for the quality criterion efficiency, because the accuracy and correctness of the model depends naturally on the context of the software development environment. Against them, our aim in this paper is to reflect and summarise the various steps and aspects of the construction of this quality model and to provide you with an applicable, systematic proceeding for structuring goal-oriented software quality models. The described proceeding is not limited to the regarded quality criterion efficiency nor to the early phases of the software life cycle, but it will be suitable for applying to different problems in the context of software quality.

Based on our experiences there are two success factors in developing and using a software quality model in practice which are considered in this presented guideline:

- The quality model has to reflect the context and must be suitable for adaptation to the environment of the software development.
- Persons who are affected with the software development have to participate in structuring the quality model and must consent to the contents of the quality model.

Proceeding for structuring

The course of developing the goal-oriented software quality model is defined by the following process model which is organised in three different abstraction levels, named AL-1, AL-2 and AL-3. Each abstraction level defines a milestone with a corresponding result:

![Process model for structuring the quality model](image)

**Figure 2: Process model for structuring the quality model**

**AL-1: Structuring the context-oriented Goal-tree**

Starting point in the structuring process is a vague problem description about the interesting quality aspect which is derived for example from an assessment
or directly from the system requirements. In the first step, the intention is to clarify the interesting quality aspect in the regarded problem context. This means, on the one hand you have to formulate the quality goal you want to achieve and on the other hand you have to specify the reflection of this quality goal in the context of the development environment.

From our experience, the GQM-method (Basili [6]) provides a suitable method to formulate the abstract and vague quality aspect in form of a measurable quality goal. The result of applying GQM is a GQM-plan which defines the interesting quality goal operationally and involves a specific definition of the quality goal, several questions, and for each question several metrics for quantification of the regarded quality goal. The definition of the quality goal, named GQM-goal, is composed of five attributes: Object, purpose, quality focus, viewpoint and environment.

In this level we use GQM exclusively to formulate and define the quality goal and do not derive questions and metrics yet. Afterwards we apply the Goal-tree method to refine and clarify the abstract GQM-goal in the project context to more comprehensible subgoals. The refinement stops at the level of process or product specific aspects.

The context-oriented Goal-tree as the result of this first step is built of two components. First, it is gained by a brainstorming and structuring process, e.g. as part of an organised and moderated workshop, and second, by input from similar problem contexts or from literature. The main topics are gained by a workshop where experts and those persons who are involved in the software development participate. The task of this workshop is to get a common comprehension of the problem context with regard to the defined quality goal.

![Figure 3: How-Why-Principle for structuring the Goal-tree](image)

The Goal-tree reflects, summarises, and specifies the set of interesting quality goals as well as the hierarchical relations between the GQM-goal as reference goal and the derived subgoals. The structure of a Goal-tree and the clarification process to derive the set of useful subgoals is mainly characterised by the How-Why-Principle (see Figure 3) between different hierarchical levels in the Goal-tree, exactly between an uppergoal and the refined subgoal.
The structuring process of the context-oriented Goal-tree in abstraction level AL-1 is arranged into four steps which are based on each other:

(1) Clarifying
Understanding the problem related to the defined GQM-goal and clarifying the context with concrete problem situations and examples from practice.

(2) Collecting
Collecting, precising and matching the corresponding quality goals by brainstorming.

(3) Classifying
Classifying the quality goals according to affinities and defining main subjects for the affinities.

(4) Structuring
Structuring the goal classes in a hierarchical manner by using the goal-tree specific How-Why-Principle (see Figure 3).

AL-2: Projection to the process and product model
In the previous abstraction level we specified quality goals corresponding to the existing problem context and their relationships, represented in form of a context-oriented Goal-tree. Now, the task is to project and refine these quality goals to the components of the available development process and their belonging products. For example, when we focus on the early phases, especially to systems analysis, we have to project and reflect the subjects of the Goal-tree to the systems analysis process and the belonging requirements document in expression of SA/RT.

A contribution of fundamental importance for this step is the specification of the existing processes and the intermediate products which result of the developing process. Derived from the contents of the Goal-tree it is optional necessary to define new process elements in addition to the existing ones, and to integrate it into the process model, e.g. related to this project, to add a specification method for efficiency requirements to the SA/RT-method in the systems analysis process.

AL-3: Derivation of metrics
After refining the interesting quality goal into subgoals and their projection to the existing process and product models of the project context our intention at this abstraction level is to derive metrics from the subgoals, actual from the leafs of the Goal-tree. In this situation it is recommended to apply the GQM-method to derive the metrics from the subgoals. GQM provides not only the traditional information of a derivation method, i.e. what metrics to use, but also supports the user in how to interpret the measurements relating to the
goal. This will ensure that all measurements are done with a clear objective and that the measurements are traceable to this objective.

Adjoining to the GQM-approach for each subgoal corresponding questions are derived. Each question is related to this subgoal and describes an aspect, orientated towards the questions categories defined by GQM, and characterises the object of measurement of this subgoal in detail. In the next step metrics, at least one for each question, are derived. The derived metrics, atomic metrics or composed metrics, define the questions operationally.

Exemplary results

In this section our aim is to present exemplary results we obtained in each abstraction level by applying our proceeding of structuring a software quality model in the context of software development for robotics systems. The presented results reflect only exemplary facets, but not the complete project result.

AL-1: Structuring the context-oriented Goal-tree

For structuring the Goal-tree we organised a one day workshop where persons who are affected with the development in this software project and experts from the research department participated. In this workshop we had first to define the quality goal related to the attributes of GQM, and second, based on this GQM-goal, to structure a context-oriented Goal-tree related to the software project.

The result from the first activity includes following basic sources of information about the quality goal, formulated by GQM:

<table>
<thead>
<tr>
<th>GQM-goal</th>
<th>Attributes of GQM-goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improve the appraisal of fulfilment efficiency requirements from the software developer's viewpoint in the early phases of software development of robotics systems</td>
<td>Purpose Quality Focus Object Viewpoint Environment</td>
</tr>
</tbody>
</table>

Figure 4: GQM-goal from project context

As we see, this is a very abstract quality goal and therefore it is very difficult to derive directly questions and metrics for quantification. In this situation it is helpful to construct a Goal-tree to refine this GQM-goal into more comprehensible subgoals.

The Goal-tree we got from a brainstorming and structuring process in this workshop contains in summary more than 30 subgoals which are grouped in 4
categories. An extract of this context-oriented Goal-tree, without product or process specific subgoals, is now presented (Note: *efficiency requirements* is abbreviated by *ER*):

![Image]

**Figure 5: Extract of the context-oriented Goal-tree from project context**

In summary, the Goal-tree contains four categories of aspects that contribute to appraise the fulfilment of efficiency requirements in this project context:

- In the first category there are subgoals that come to terms with the problem of acquisition and specification of efficiency requirements as well as with proving the logical correctness of the specified efficiency requirements.
- Main part in the second category are aspects about factors from the environment which influence the efficiency of the software product.
- The third category combines aspects about the methodical ability in specification and implementation of efficiency requirements in several phases of development.
- The fourth category describes goals related to necessary activities to implement and test the efficiency requirements in all phases of software development.

**AL-2: Projection to the product and process model**

Derived from the problem context, we concentrate on the systems analysis based on the SA/RT-model with E-Spec extensions. Therefore we have to project the subgoals of the Goal-tree to the extended SA/RT-model in the environment of the software development in robotics systems, especially under employment of the CASE-Tool Teamwork (Cadre [2], [3]).
For example, the projection of the following selected subgoal from the first category results in:

**Figure 6: Example for projection a subgoal to the extended SA/RT-model**

After the projection to process and product specific subgoals we must trace those subgoals to the data that are intended to define those subgoals operationally. At this time it is recommended to classify the subgoals, because on the one hand not all subgoals must be quantified, on the other hand the regarded product and process models in the project context have to be extended for a quantification. In our context there are three classes of subgoals:

- **Subgoals which are fulfilled by application of the CASE-Tool**, e.g. proving the logical correctness of the SA/RT-model is settled by the model editor in Teamwork. This kind of subgoal must not be refined by questions and metrics.

- **Subgoals that cause a methodical extension of the existing process and product model for their fulfilment**, e.g. a formal method to specify efficiency requirements in the context of systems analysis by SA/RT. The application of this method impacts the fulfilment of the related subgoals.

- **Normal subgoals which had to be refined by questions and metrics for measurement and evaluation**. We call them in future quantifiable subgoals.

**AL-3: Derivation of metrics**

In the abstraction level AL-3 a set of questions is derived from each quantifiable subgoal to characterise the way to perform the achievement of this specific subgoal. Questions try to characterise the object of measurement with respect to a selected quality issue and to determine its quality from the selected viewpoint. Based on the derived questions a set of data is associated with every question in order to answer it in a quantitative way.

In our context there are the following objects, documents, and models that are able to be used for the measurement:
SA/RT-model, edited by the CASE-Tool Teamwork (Data Dictionary, Data Flow Diagrams, P-Specs, C-Specs, Information Model, ...)  
E-Specs (efficiency requirements represented by formal annotations)  
Components of the CASE-Tool Teamwork, especially elements of the simulation model Teamwork/SIM™ (Cadre [2])

For example, we got in this abstraction level for a chosen subgoal the following questions and belonging metrics:

<table>
<thead>
<tr>
<th>Subgoal</th>
<th>ER from different Data Flow Diagrams (DFD) have no conflicts</th>
</tr>
</thead>
</table>
| Question Q-1  
Are timing constraints in top-level DFD in conflict with timing constraints in lower-level DFD? | Metric M-1.1  
Value = 1, if Time(Child-Process) > Time(Parent-Process)  
Value = 0, otherwise |
| Metric M-1.2  
Value = 1, if \( \sum \) Time(Child-Process in linear path) > Time(Parent-Process)  
Value = 0, otherwise |

**Figure 7: Questions and metrics for a subgoal**

**SPECIFICATION OF EFFICIENCY REQUIREMENTS**

**Motivation**

In the systems analysis a methodical support to specify the efficiency requirements is very profitable, especially for a measurement based evaluation of quality. A formalised specification of requirements offers an important contribution for quality improvement in this context:

- First, an automatic evaluation and determination of metrics for interesting quality factors is supported, and
- second, you are restrained yourself to acquire the essential information about efficiency and specify it in the analysis model.

Such a specification method must be accepted by the system analysts. So, you have to take several constraints into consideration, e.g. applied engineering methods, tools and also human factors. In this problem context the following influences are significant and have to be considered:

- methodical consistency to the formal analysis method SA/RT  
- useability in connection with the CASE-Tool Teamwork  
- consistency with the simulation facility of Teamwork  
- "keep it simple" strategy in application
Additional to these constraints it has to be considered that typical efficiency requirements of practice can be directly represented in the analysis model. When we look after practice, there are two classes of efficiency requirements: Timing constraints and storage constraints. Timing constraints can once again be classified into sporadic and periodic timing constraints. In the future expositions of this subject in this paper we will limit the efficiency requirements to sporadic and periodic timing constraints.

**Basic concepts of the specification method**

Related to the basic components of the systems analysis model, timing constraints refer to the execution of processes, represented by bubbles in Data Flow Diagrams (DFD), and refer to time spent in definite system states. If we regard the execution of a process as a transition from an initial state to a final state, we can use state transitions as a common basis for modelling timing constraints. Therefore, *time-evaluated state transitions* are the basic components for modelling timing constraints belonging to the execution of processes as well as for time spent in system states.

For modelling the course of events, also the dynamic behaviour, within the executing of processes we use the internal processing states from the simulation model of Teamwork (Cadre [2]). The processing of each process is modelled by a sequence of state transitions. The semantic meaning of this states and their transitions are defined in the simulation model. The use of the internal processing states from the simulation model is advantageous in different aspects. First, it can be transformed to the SA/RT-model, independent of the use of Teamwork, and it supports the modelling of dynamic behaviour of processes in the SA/RT-model. Second, it offers the possibility to validate the efficiency requirements by simulation of the SA/RT-model with Teamwork/SIM™.

**Specification of timing constraints for processes**

The purpose of this section is to explain, how to specify timing constraints for processes within Data Flow Diagrams (DFD) of the SA/RT-model. First, we have to characterise the process element of the DFD and then we have to transform the basic concept of time-evaluated state transitions to this context.

A process element of DFD is characterised by following concepts:

- **Static perspective** - Each process, represented as a bubble in the DFD, possesses a set of input as well as a set of output of data flows and control flows. The data flow and control flow influence the execution of the process, and therefore they have to be considered in the specification.

- **Dynamic perspective** - The processing of input to output is characterised by internal processing states which are derived from the simulation model of Teamwork. The state transition as well as time spent in the states depend on the values of data and control input. Furthermore the processing
depends on the actual system state, defined by the State Transition Diagram (STD) of SA/RT (Hatley [1], Cadre [3]).

Now, a graphical notation is presented which exemplary describes how a timing constraint for a process can be specified with inclusion of its static and dynamic perspective:

![Diagram showing specification of timing constraints at processes in SA/RT](image)

**Figure 8: Specification of timing constraints at processes in SA/RT**

The graphical notation of E-Specs for processes contains different elements:

- **Initial state** (*ready*) and final state (*firing*)
  The initial state, e.g. *ready*, and final state, e.g. *firing*, define the internal state transition of processing to which the specified timing constraint belongs.

- **Precondition** (*Pre*)
  The specified timing constraint is only valid under a certain constellation of data and control input. *Pre* defines this constellation.

- **Postcondition** (*Post*)
  In analogy to precondition, the specified timing constraint is only valid under a certain constellation of data and control output. *Post* defines this constellation.

- **Time-Relation** (*Time_Relation*)
  *Time_Relation* defines the available time units for the defined state transition from initial to final state, under the condition that *Pre* and *Post* are valid.

In this section only an extract of the specification method, how to specify graphically sporadic timing constraints to processes in the DFD of the SA/RT-
model, is shown. In addition there exists also concepts in the specification method for the definition of periodic timing constraints as well as timing constraints for data stores in DFD and for time spent in certain system states which are defined in State Transition Diagrams (STD) of the SA/RT-model.

**A notation for textual specification**

Further processing and accessibility of the E-Specs in the extended SA/RT-model is important, especially in the context of quality evaluation. Therefore it is more advantageous to represent this graphical representation in a textual description, if possible in a formal language. Based on this consideration we developed a formal language, syntactically defined by Extended Backus Naur-Formalism (EBNF), which directly allows to specify various concepts of efficiency requirements in textual notation.

A sentence of this language is composed in the following form

\[ \langle E-Spec-Id \rangle \langle element \rangle \langle context \rangle \langle requirement \rangle \]

and consists of the four components:

- **E-Spec-Id**
  
  identification key for this E-Spec; derived from the SA/RT-model name of the regarded object *element*

- **element**
  
  model object, to which the E-Spec belongs to; e.g. a process or a data store

- **context**
  
  defines the context, in which the specified E-Spec is valid; e.g. the E-Spec for a selected process is valid only within a certain system state

- **requirement**
  
  defines for the model object *element* the timing constraint and the state transition to which it belongs

Projected to the specification of timing constraints in the context of processes within Data Flow Diagrams, a sentence of the language has the following syntactical expression:

<table>
<thead>
<tr>
<th>syntactical expression</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object_Type.Object_Id&quot;:&quot;</td>
<td>model object within the process is defined</td>
</tr>
<tr>
<td>ESpec.No&quot;#&quot;</td>
<td>serial number for the E-Spec within this model</td>
</tr>
<tr>
<td>P&quot;%&quot;</td>
<td>regarded process</td>
</tr>
<tr>
<td>C&quot;:&quot;</td>
<td>valid context; here system states</td>
</tr>
<tr>
<td>I, F&quot;%&quot;</td>
<td>initial and final state of internal processing</td>
</tr>
<tr>
<td>Pre &quot;/&quot; Time_Relation &quot;/&quot; Post</td>
<td>timing constraint for transition from state I to F</td>
</tr>
</tbody>
</table>

*Figure 9: Syntax for E-Spec notation of processes*
For example, the textual representation of the previous, in a graphical notation specified efficiency requirement for process A looks like this:

DFD.0 : ESpec.7 # Process A % all: ready, firing % / t <= 4 /

CONCLUSION

In conclusion, this approach is a contribution to improve the software quality for real-time systems by providing substantial support for specification and evaluation of efficiency requirements in the systems analysis phase of software development. Failure costs related to the violation of efficiency constraints that are caused in early phases of the software life cycle are reduced by increasing appraisal and prevention activities in the systems analysis.

REFERENCES


