Integrating Goal-Oriented Measurement in Industrial Software Engineering: Industrial Experiences with and Additions to the Goal/Question/Metric Method (GQM)

Rini van Solingen[1]
CMG Trade, Transport & Industry
Advanced Technologies, Software Engineering
P.O. Box 8566, 3009 AN Rotterdam, Netherlands
E-mail: rini.van.solingen@cmg.nl

Egon Berghout
Department of Information Systems and Software Engineering, Delft University of Technology,
P.O. Box 356, 2600 AJ Delft, The Netherlands,
E-mail: e.w.berghout@its.tudelft.nl

Abstract

In this paper two major additions to the Goal/Question/Metric (GQM) method are presented based on seven years of experience with goal-oriented measurement programs. GQM, is a method to organize software measurement programs. The initial ideas of GQM were first published in 1984 and, since then, much industrial experience has been gained and theory has been developed to underpin the approach.

The two additions concern a thorough elaboration of the feedback of measurement data and a conjunct cost/benefit analysis of the GQM project. The first addition refers to establishing conditions that are necessary to facilitate learning in software measurement programs. Learning about software quality is identified as the most important objective of a measurement program. The second addition refers to identifying whether the measurement program was actually worthwhile. Through a cost/benefit analysis the software measurement activity is brought in perspective with the other business objectives.

Case study research illustrates the importance of both additions. In the past seven years we have been involved in fifteen measurement programs in five industrial organizations. Successes and failures in these organizations support the importance of both additions. The two additions: feedback sessions and cost/benefit analysis are described in detail in this article.

1. Introduction

The subject of quality improvement has received much attention in the past years. Initiatives for Software Process Improvement (SPI) such as benchmarking, assessment, measurement, product evaluation, process modeling etc. have gained ground in both the industrial and the academic worlds. This paper is focused on one of these SPI approaches: measurement.

Measurement is used for objective and quantitative evaluation of software processes, products and resources. Software measurement is an essential prerequisite for continuous improvement and learning in industrial development projects. Measurement is especially applicable during the implementation of process changes, since measurement includes a feedback mechanism to the software development team and enables process optimization through integrated group learning. Ideally, measurement should be driven by goals in order to focus and to limit cost. In the past years of industrial experience it has become clear that two factors have a critical impact on measurement success and continuation: feedback to the software development team, and capturing cost and benefits of measurement.

In this paper we provide an overview of practical lessons learned and guidelines for these two factors, all based on multiple measurement case studies in several industrial projects.

1.1. Measurement of products, processes and resources

'Measurement' is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules [29]. Measurement can be used to characterize and learn about processes, products and resources. It can be used to measure for example quality characteristics of a certain software product, to measure effectiveness of a certain software process, or to measure the amount of resources spent on a specific product with a specific size. Measurement has an important role in other engineering disciplines, as such disciplines are based on measuring and codifying knowledge in models and guidelines in order to transfer knowledge and to package learning from practice [66], [9], [32] and [46].

[1] At the time of writing employed by the Fraunhofer IESE in Kaiserslautern, Germany.

1530-1435/01 $10.00 © 2001 IEEE

An empirical factor is added to the research on software engineering by using measurement: new methods, techniques and tools are measured to understand how good they are, which problems they solve, which they do not solve, and the basic needs to apply them. In other words: measurement provides an excellent mechanism to learn what works and what does not work. [13], [32]. Examples of measurement results are [39], [58], [10], [12], [49], [19], [59], [29], [53], [37], [36], [35]:

- increased understanding of software development processes;
- increased control of the software development process;
- increased capacity to improve the software development process;
- more accurate estimates of software project costs and schedule;
- more objective evaluations of changes in technique, tool, or methods;
- more accurate estimates of the effects of changes on project cost and schedule;
- decreased development costs due to increased productivity and efficiency;
- decrease of project cycle time due to increased productivity and efficiency;
- improved customer satisfaction and confidence due to higher product quality.

The number of metrics that can be collected is enormous, see for example [67], [33], [68], [25], [81] and [2]. An important question is, therefore, which metrics to collect and which not? The key answer to that question is that only those metrics should be collected which data will actually be used [72]. Therefore, organizations need to identify which urgent information needs they have and base their metrics programs on those needs. It has been proposed to base metrics on goals: project goals, business goals, people goals, or improvement goals [14]. Refining goals to metrics has proven to be a successful method to assure that data that is collected is actually relevant and used. A framework to support this refinement is the Goal/Question/Metric approach [14], [10] and [71].

1.2. Goal/Question/Metric measurement

In the GQM method a systematic approach is represented for tailoring and integrating goals to particular models of software processes, products and quality perspectives, based upon the specific needs of the project and the organization [10]. The result of the application of the GQM method is the specification of a measurement system targeting a particular set of issues and a set of rules for interpreting measurement data.

By using GQM a certain goal is defined, this goal is refined into questions, and, subsequently, metrics are defined that should provide the information to answer these questions. By answering the questions, the measured data can be analyzed to identify if the goals are attained. Thus, by using GQM, metrics are defined from a top-down perspective and analyzed and interpreted bottom-up.

GQM trees of goals, questions and metrics are built on knowledge of the experts in the organization: the software engineers [12] and [50]. Knowledge acquisition techniques are used to capture the implicit models of the engineers built during years of experience. Those implicit models give valuable input to the measurement program and will often be more important than the available explicit process models. Also the anticipated outcome of the measurements, the so-called 'hypotheses' of the engineers are captured. Actual data can then be used to compare to the initial hypotheses and to identify causes for flaws and to learn causal relationships. Even though the GQM approach is often considered to be a process measurement approach, it is also effective to measure products and resources [19], [71] and [73].

1.3. Industrial experiences

During the past years many companies have gained and published their measurement experiences, see for example: [11], [58], [31], [36], [16], [37], [74], [52], [50], [29], [41] and [71]. Lessons from these experience reports are:

- Successful software measurement is possible, although difficult. Furthermore, what 'successful' actually means is context specific: the purpose/objective of measurement differs.
- Software measurement should focus on a particular improvement area to limit cost of measurement and to prevent measuring everything; aligning measurement with goals is frequently proposed.
- Most experiences focus on the definition process of a measurement project. Guidelines and experiences on how measurements should be analyzed and interpreted are mostly assumed to be common knowledge.
- Commitment for a measurement program is crucial, both from the software development team as well as from management viewpoint.
- Return of investment calculations of measurement programs are seldomly reported.

Through our research we have been involved in five industrial organizations and fifteen software measurement projects. A number of these case-study experiences have been published [50], [72], [73], [71], [16] and [70], others were company confidential. We

have learned many aspects of software measurement during these industrial studies. Our observation is that there are two factors that are often not sufficiently addressed in industry and according to us are probably the most important factors that facilitate software measurement success:

1. Feedback sessions. In these sessions the software development team interprets the measurement data, draws conclusions, takes decisions and defines action. Measurement is started to support the attainment of certain goals. This is achieved through creating a certain understanding and enabling the possibility for corrective action. Many metrics programs in industry did collect ample metrics, however, did not provide the necessary feedback to the development teams. Software measurement should be regarded as a group learning process. Without feedback sessions in which these groups analyze their data and analyze what they can improve with the measurements, a program will die a silent death. Frequent (every 6-10 weeks) feedback sessions are mandatory. In these sessions the software development team that collected the data is confronted with charts and tables of aggregated measurement data, which they are requested to interpret, conclude, decide and define action. Furthermore, these feedback sessions should include a number of important elements. Requirements to feedback sessions will be addressed in section 2 of this paper.
2. Cost/benefit analysis. Cost and benefits of software measurement projects are in most organizations not made explicit [69], [24], [6], [7] and [54]. There are few publications on cost/benefit analysis of improvement projects [69], however, from our experience in industrial cases we assume that a negative perception of the cost and benefits, particularly by management, is one of the most important reason to stop a quality improvement program. This assumption is supported by [15], [40], [64] and [63]. To prevent individual misjudgments regarding the profitability of a measurement project an explicit analysis is recommended [69], [24] and [6]. In the industrial cases we examined, it was always possible to make such an analysis and in these cases also show a positive return on investment. Making costs and especially benefits explicit and translating them to financial values will be addressed in section 3 of this paper.

1.4. Objective and outline

This article focuses on describing the additions to the GQM method in such a way that software engineers and researchers should be encouraged to apply the additions in their own measurement projects. This implies that most attention will be given to describing the proposed methodology and the impact we observed in the various industrial cases. In this article we will not focus on the detailed validation of every statement in the various cases. Whenever such validations have been published in earlier, more focussed papers, this will be referred to. In section 2 requirements to the feedback process are described. In section 3 requirements to cost/benefit analysis are described. In section 4, the findings of sections 2 and 3 are associated to the objectives of the various stakeholders in a quality improvement program in order to provide a theoretical explanation of the importance of both additions.

2. Feedback

In this section first an overview of industrial experiences regarding feedback of measurement data is given. Secondly, a model to understand how learning in feedback sessions takes place is presented. The section ends with stating the most important learning enablers in feedback and the conclusions regarding this section.

2.1. Observations from industrial measurement programs

Feedback of measurement data and the associated analysis of this data by the software engineers is done in so-called 'feedback session'. Feedback sessions are meetings of all software development team and GQM team members in which the measurement results are discussed. Outcomes of a feedback session are interpretations, conclusions, decisions and actions. The typical way in which feedback sessions are carried out in industrial measurement programs is according to the following steps:

- Preparing a feedback session: Preparing feedback sessions concerns processing the collected data into presentable and interpretable material. The GQM plan provides the basis for preparing feedback sessions: feedback material should support answering the questions as defined in the GQM plan, and based on these answers, one should be able to conclude whether the defined measurement goals are attained. The GQM team primarily carries out the preparation of feedback sessions.
- Holding a feedback session: Feedback sessions are held approximately every six to eight weeks. They typically last about 1.5 to 2 hours, with a maximum of 3 hours. Any longer is normally experienced as counter-productive. This time is sufficient to discuss some 15 to 20 slides (containing graphs and tables) [71]. In principal, a software development team should run a feedback session alone. They analyze,

interpret and draw conclusions regarding the measurements, and translate their conclusions into action points. After all, they are the experts with respect to the object under measurement. The software development team should focus on: evaluating action points from earlier sessions, interpreting measurement data with respect to the questions and goals as defined in the GQM plan, and translating interpretations into conclusions and action points. The GQM team should avoid interpreting the data themselves. Their role is to challenge a project team, for example, by offering alternative interpretations [70]. Furthermore, the GQM team provides support and may, for example, provide meeting reports. Feedback sessions are a delicate phase in a measurement program as mutual trust among all participants is, for example, an essential element of a feedback session. Through focusing on identified goals, questions and metrics, the discussion will start on the basis of facts.

- Documenting results of a feedback session: After the feedback session, the GQM team writes a meeting report containing all relevant observations, interpretations, conclusions and action points that were formulated during the session. Advised is to stay with the rule that the software development team 'owns' the measurement data and therefore decides on distribution of both data and reports to, for example, management. When the GQM team wants to inform higher management, the GQM team only uses particular, often aggregated results and asks permission to do so. In order to reuse measurement results and experiences in future measurement programmes, the results of a measurement programme should be documented in such a way that they are easily accessible and understandable.

The concept for organizational support for GQM measurement programs distinguishes between a software development team and a GQM team [10]. This GQM team supports a software development team by carrying out all tasks in a measurement program that do not need to be performed by the software engineers. As such, the engineers only provide their input and participation when necessary, leaving the workload of measurement relatively low. Our research indicated that the amount of time spent by a software development team on a measurement program is limited to 2% of their time, reducing even to less than 1% for teams with GQM measurement experience [16] and [71]. Beside the benefit that measurement does limit the participation effort of the software engineers, the other benefit is that this 'two team structure' facilitates continuation. Both teams depend on and therefore trigger each other, which results in a continuous process. The software development team triggers the GQM team with data and requests for aggregated results. The GQM team triggers the software development team with requests for measurement data and interpretations of feedback material.

The second point is the observation that the main purpose and outcome of measurement programs is *learning* [70]. Measurement programs are performed to increase understanding, to control and to optimize practices [7] and [71]. This is all centered around collecting information to increase knowledge, which is nothing more than learning. Considering that improvement and measurement should be learning processes, immediately leads to the recommendation to explore learning theory to identify how measurement can be performed better by increasing the learning effects. During the past years we performed such research and explored learning theory and formulated practical guidelines on increasing measurement learning effectiveness [72] and [71]. Measurement programs can only be successful when the participants actually learn. Realizing that learning is an important objective of measurement, learning theory is explored in order to understand how learning takes place within industrial measurement programs and also how this learning can be encouraged. Learning deals with expanding knowledge and changing behavior [76] and [29]. Elements of learning theories were again included in feedback sessions and led to two particular findings:

- A model of the learning process between software development team and GQM team.
- A list of learning enablers that stimulate group learning within measurement programs.

These two products are elaborated upon in the following sections.

2.2. Learning model for software development team - GQM team interaction

In this section we will position the guidelines from learning theory into a model of feedback sessions. Based on learning theory of student-teacher interaction [27], a model is proposed regarding development-GQM team interaction. This model is depicted in Figure 1 and describes the learning process that the software development team and the GQM team go through. The model illustrates that the results of a feedback session are significantly influenced by the organization of the feedback process.

The model of feedback sessions contains two loops. One loop represents the learning process of the software development team follows, the other the learning process the GQM team. Both loops contain two types of impact: short-term and long-term changes. A GQM team

possesses specific characteristics, and defines the feedback. The software development team (with also specific characteristics) has a particular perception of this feedback. The results of a feedback session are improvements that are made explicit in: interpretations, conclusions and action points. These improvements influence both software development team and GQM team, as well as the short-term and long-term changes. A more detailed description of this model is given in [72]. Multiple case-studies have been carried out to validate the model.

is argued that a GQM team should be independent of the software development team and have no interest in the data that a software development team gathers. To be able to guide and support the measurement program, the GQM team needs to have an adequate level of background knowledge of the processes and products that are being measured. This is an important prerequisite if this team is to question and challenge the interpretations made by a project team. It is also a prerequisite in the sense of respect from a project team. The GQM team should regard themselves as facilitators

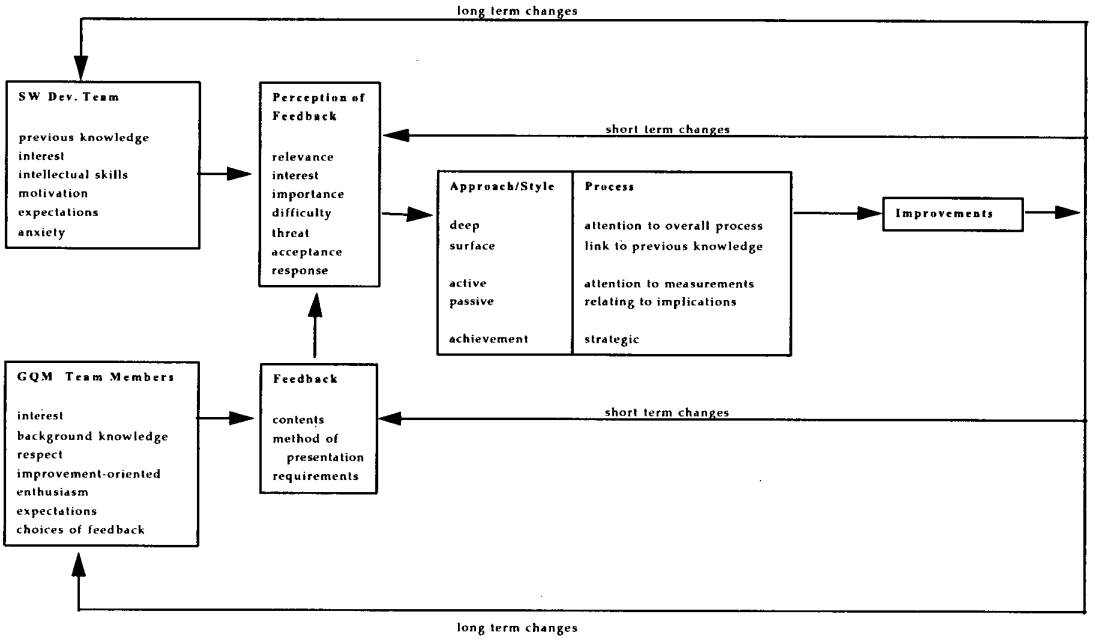

Figure 1: Conceptual model of learning in a feedback session [72]

The previous knowledge of the software development team with regard to the measured process, is captured by building the measurement program on their knowledge of the process, through interviews for the definition of the Goals, Questions and Metrics. Measurement data are normally of great concern to the software engineers, representing their performance and often accessible to other departments in the organization.

The GQM team members are the people who provide the feedback: they process the data and prepare it for analysis and interpretation. Also, they guide the feedback process and assist software engineers in their analysis and interpretation of the data. An important requirement that needs to be fulfilled to succeed in the measurement program, is that a high level of mutual trust and cooperation between the two teams exists. Therefore, it

of learning and, in guiding the measurement programs, be improvement-oriented. If they are, they not only assist in improving the processes of the project team, but also learn how to improve their own work.. Enthusiasm for the measurement programs is required to create a good atmosphere during the definition of the GQM plan as well as during the feedback process.

Regarding the feedback process the following elements are pertinent. The contents of the feedback sessions should be based on the measurement results and the GQM plan. Presented material should be limited in quantity and provided regularly. The concepts of relating new knowledge to available knowledge are an integrated part of GQM measurement. The emphasis on relating new and existing knowledge, is a fundamental prerequisite for the interpretation of measurement data.

During the presentation, much attention needs to be given to these aspects. In presenting results, they should be related to the goals and questions and also to their recorded expectations (hypotheses). Learning in feedback is primarily achieved through discussion of the data presented. This way, it is an explicit conversation of people [18]. Although it will be clear that this statement is primarily aimed at educational environments, it applies equally well to the material that is presented in the feedback process. Relevance is primarily determined by a measurement process and a correct refinement of goals into questions and metrics. When the right processes are measured (the ones that require improvement according to a project team), and the right data are gathered that indicate possible improvements, the relevance of the measurement program is normally high. Carrying-out a measurement program with a central role for the project team, also implies that the relevance is guarded by the project team. The relevance of the feedback is primarily the GQM team's responsibility as the GQM team is expected to process the data and prepare them for interpretation. This relevance can be achieved by relating the processed data back to the GQM plan, in which the objects, that are considered relevant by the Project team, are stated. The perception of the interest of the feedback itself is considered equally important: it is important for the GQM team to convince the software development team of the importance of feedback and of the project team's contributions to the feedback process. The importance of the feedback process, lies in the fact that the knowledge of the software engineers is shared and used to improve the processes. To the members themselves, this gives an opportunity to improve the processes in a way they consider effective.

2.3. Learning enablers for feedback sessions

Based on learning theory, the most prominent learning enabling factors for feedback have been identified. This paper does not leave sufficient space to describe the complete analysis of these enablers described in [30], [65] and [55]. For this analysis we refer to [70]. The learning enablers will be subsequently described, together with what the enabler means within the context of software development. The enablers are:

- *Climate of openness.* A climate of openness addresses the establishment of an environment in which free flow of information, open communication, sharing problems and lessons learned, and open debate of ways to solve problems, is available. Such a climate or 'learning culture' could seem a simple concept, however, is difficult to establish in practice. Research has indicated that current structures for control and management in organizations tend to disable such climates of openness and with that decrease the commitment of their people [3] and [75]. The intrinsic motivation of people especially is crucial for establishing a creative and learning oriented environment. Practical actions that managers can take to increase the intrinsic motivation of people are grouped over: challenge, freedom, resources, work group feature, supervisory encouragement, and organizational support [3]. A climate of openness appears to be one of the most crucial prerequisites for organizational learning. This requires a context in which people are willing to learn from their mistakes and willing to discuss underlying causes and models for these mistakes.
- *Scanning for knowledge.* In the broadest sense this means that there should be a continuous search for knowledge that could be relevant or applicable in the specific learning situation. Scanning for knowledge from previous products, competitor products, similar products, or new methods is an important input to the requirements phase of a software project. Software product requirements should preferably not be built from scratch. Carrying out post-mortem analysis to find out whether a certain used process model was adequate, is also a good source of knowledge to increase learning effects. Reading publications on achievements in software engineering by software engineers, is another way to scan knowledge.
- *Information on context and current state of the system.* Learning adds knowledge to an existing situation and is influenced by the state of external influences. Information is needed on the context and current state to learn appropriately, and select the best-suited additions. The retrieval of information on the context and the current state of the product and the project is essential here. Making processes explicit, measuring the performance of processes, or the current state of the product and its quality is a useful source of information for this learning enabler.
- *Team learning.* Team learning is an important part of an organizational learning process. It means that learning is established within groups that work together towards a shared vision and mutual objectives. Joint formulation of learning objectives, information sharing, discussion, and drawing conclusions together take place within team learning. Team learning can be used to find out a good way in which product requirements need to be specified to let the final product comply to them. It is also important that development teams learn the behavior of different development processes. A specific process may not always give the same effect within different projects, for different products, with

different team members. These differences and the causes for them should be determined. Measurement is a powerful mechanism to enable this group learning. Discussing measurement results within a software development team and challenging a team's interpretations, is already a means to establish team learning.

- *Modeling of the system under control.* In order to control a system, a model needs to be created from this system and its influencing factors. This can be done through process modeling, and modeling of the relationship between the product requirements and this process.
- *Possibilities for control.* In order to steer a process towards the required outcomes; possibilities for control should be available. This means that during a software project (corrective) action can be taken whenever necessary.
- *Involved leadership.* Managers should articulate vision, take part in the implementation of ideas, and be actively involved in the learning processes. The role of a manager for the establishment of organizational learning, and motivating the people in the organization is crucial. In a learning organization managers and the role of the manager is changed largely compared to traditional management styles. The largest differences are that [65]: the manager is a designer of the learning organization, a teacher of the view on reality, and a steward for the people they manage.
- *Explicit goal definition.* In order to have clear targets towards learning, particular goals should be defined and made explicit. Learning processes benefit if it is clear what the goals are and in which area learning is required to attain such goals. Expectations (hypothesis) must be explicitly specified regarding attainment of these learning goals, because expectations can be compared to actual values and reasons for differences can be identified.
- *Monitoring performance gap.* Monitoring the differences between target and actual situations is an important prerequisite for learning. It supports in identifying what is going well, and what needs improvement. Through this performance monitoring, people get feedback on their way of working and learn where to improve. Monitoring a possible performance gap is not only done for the product, but also for the development process. The performance of process actions should be monitored and if differences exist between expected and real effects of process actions, corrective action can be taken.

2.4. Conclusion regarding feedback

Although rarely considered in industry, probably the main objective of software measurement is learning. Without learning there will be no increased understanding and therefore no improvement. In this section, learning theory was used to identify possibilities to increase the learning effectiveness of feedback sessions. The outcome of this investigation was presented: a model of software development team - GQM team interaction that describes both the learning processes of a software development team and a GQM team. Furthermore, several enablers were identified that support learning in measurement programs.

3. Management of cost/benefit

3.1. Cost/benefit analysis

As referred to in the introduction, the second addition to the GQM-method is a cost/benefit analysis of each GQM-project. Through such an analysis, it becomes apparent whether a measurement program was worthwhile for the company as a whole and to what extent. Several researchers assume that in practice many measurement projects are abandoned, because management implicitly evaluates them as unbeneficial [15], [40], [64] and [63]. In such a case, the perceived cost of particularly time of engineers is expected to be higher than the benefits of the improved quality. However, when a cost/benefit analysis is explicitly included in a measurement project, these implicit ideas are made explicit and open to discussion. Including a cost/benefit analysis appears to have a significant effect on quality improvement programs.

Many engineers still claim that cost/benefit analyses are impossible to make in practice due to the indirect relation of cost and benefits of an improvement program. In practice this statement seems to be both true and untrue. In the cases we participated in, it was always relatively simple to estimate a return of the project that was agreeable to all participants. These calculations typically included all measurement effort, efficiency benefits and additional revenues. However, gathering data became more complex when the calculation was made more comprehensive. Particularly, if the calculation included benefits outside the software development department (and this is always the case). For example, valuing the contribution to the corporate identity. These valuations require studies of customer behavior and if such information is not already available, it will certainly not be worthwhile to acquire them just to calculate the return of a measurement project. There are also several examples of cost/benefit calculations available, see for example [16], [69] and [26]. Insight in

the extent to which the measurement programme was worthwhile provides important information about the resourcing of measurement project and the appropriateness of the measurement area.

In this section an overview of a cost/benefit framework is presented, together with our industrial experiences. First, the cost analysis is presented. Subsequently, the benefit analysis is given.

3.2. Analysis of cost

A plethora of literature regarding software development cost analysis exists [29], [48], [42], [69] and [47]. Most literature focuses on planning the programming effort. Little research is available on the particular costs of measurement projects. In this paper the cost model of Birk [16] is preferred. Based on the phases of the GQM method, this model provides a framework for addressing costs. A typical cost model for the application of GQM is given in Figure 2. The project phases include the following activities:

1. *GQM programme planning.* This includes the identification of available input, preconditions and constraints, the set up of an infrastructure, the selection of an improvement area, the selection of a project, initial planning of the measurement programme, and the preparation and training of the Project team.
2. *Identify and define GQM goals.* This includes the characterisation of the project and organisation, identification and selection of improvement goals, definition of the measurement and GQM goals, modelling of the relevant software processes involved in the measurement programme, and the identification of artefacts to be reused.
3. *Conduct GQM interviews.* This includes studying documentation, defining, scheduling and inviting interviewees, briefing of a project team, conducting GQM interviews and reporting them.
4. *Develop GQM deliverables.* This includes definition, reviewing, and refining of a GQM plan, the definition of a measurement plan, identification and definition of data collection procedures, reviewing and refinement of a measurement plan, and development of an analysis plan.
5. *Data collection.* This includes a data collection trial to test data collection procedures and forms, briefing the software development team and kick-off of the measurement programme, collection, validation, coding and storage of measurement data.
6. *Data analysis and interpretation.* This includes analyses of the measurement data, preparation of the presentation material, planning, conducting and reporting of the feedback sessions.

Both [16] and [71] make distinct costs models for the first GQM-project and subsequent, more routine, projects. The following cost structure appeared to be typical for routine application of GQM [70], [71] and [16]:

- Roughly 30% of the effort is spent on defining the measurement programme, while 70% is spent on continuation of the measurement programme, which is almost completely spent on the feedback sessions.
- 70% of the effort on the measurement programme is spent by the GQM team, and only 30% of the total effort is spent by the project team.
- The effort spent by the software development team is less than 1% of their total working time.
- A total three person-months of effort is required for a typical GQM measurement programme, distributed over one calendar year.

The following cost structure appeared to be typical for first-time use of GQM:

- An initial GQM measurement programme needs approximately eight person-months of effort. This is significantly more than the three months mentioned for routine application. A first project requires a lot of initial work. An example is the initial training of the GQM team.
- Approximately 50% of the total effort is spent on defining the measurement programme. This differs significantly from the 30% in routine application. Especially the definition of the GQM and measurement plan requires more effort.
- 70% of the total effort on the measurement programme is spent by the GQM team (including expert), and 30% by the project team, which is identical to routine application.
- The effort spent by the software development team is less than 2% of their total working time, which still is double the amount compared to routine application of GQM measurement.

3.3. Analysis of benefits

GQM projects may have various benefits. Goal attainment is obviously one of the most desired ones, however, substantial additional benefits will normally also occur. The following benefit categories are discerned and will subsequently be described:

- Goal attainment
- Understanding of software products and processes
- Improved communication
- Attention for software quality
- Corporate identity.

The gains of the perceived *goal attainment* are one of the most obvious benefit categories of a GQM project. For example, in one of the case studies a project was started to increase understanding of the efficiency of reuse of software components, comprehending the effects of reuse and the difference to newly developed code is than an evident benefit. In this case the outcome was in favour of reuse and consequently an estimate could be made of the efficiency gains through the anticipated increase of reuse. Furthermore, there were additional benefits. In order to make a difference between the re-used code and newly developed code, the concept of 'fault severity' had to be redefined. Otherwise, the faults in both types of code could not be compared. This, consequently, led to an increased *understanding of both software products and processes*. Something that was more difficult to express in a financial equivalent, however, could be compared to the cost of acquiring this knowledge. In the particular case this information was for free, because the efficiency improvements already outweighed the cost of the GQM project.

Besides the above benefits, there were also the advantages of *improved communication* between the engineers that discussed more issues during the various GQM meetings than the measurement results alone. Having a GQM project also implies more *attention to software quality*. In the case studies, the financial equivalents of both these categories were again difficult to determine. The same is true for the effect of the quality focus on the *corporate identity*. The emphasis on quality was, for example, also communicated to potential buyers and employers. The above benefit categories, however, did provide sufficient information to determine whether the benefits exceeded the costs in the observed case studies. In the case of the software reuse, for example, the productivity increase due to the increased understanding of the features of reused software, largely exceeded the cost.

In association with information on potential customer and employer perception of the quality programs, management appeared to be very well capable to form an opinion of the value of a GQM project. Primarily, in terms of 'highly valuable' to 'fine'. However, this does not make such an evaluation less important or incorrect. Through these explicit evaluations a better argumentation regarding the resources of the GQM projects was also possible. GQM programs are intended to focus on the most valuable area for quality improvement, because identifying this area is part of GQM's planning phase. Determining the value of this anticipated most valuable area is an important evaluation of the correctness of this choice and, therefore, regarding the feasibility of the GQM projects and quality improvement in general.

4. Primary goals of GQM projects

Task	GQM Team	Manager	*Single engineer*	For 10 engineers	Total
GQM programme planning	4	2	-	-	6
Identify and define GQM goals	8	1	*1*	10	19
Conduct GQM interviews	40	2	*2*	8	50
Develop GQM plan	40	1	*1*	6	47
Data collection	24	1	*1.5*	15	40
Data analysis and interpretation per feedback session	*48*	*2*	*2*	*20*	*70*
Data analysis and interpretation (5 feedback sessions)	240	10	*10*	100	350
Total	356	17	*15.5*	139	512

Figure 2: Effort model for routine application of GQM (effort in person hours)

4.1. Stakeholder objectives for GQM projects.

As stated in the introduction two concepts and their practical implications were added to the GQM measurement methodology in this paper, being, *feedback* and *cost benefit/benefit analysis*. In this paragraph is argued that these additions can be associated to distinct stakeholders in quality improvement projects and, consequently, that meeting stakeholder objectives will be of major important to the success of a GQM project. Consequently, stakeholder theory is identified to underpin the GQM methodology and an interesting area for further research.

In this section, first, the various stakeholders will be described. Second, their typical objectives are stated.

4.2. Stakeholder categories

In this article it is argued that both feedback and cost/benefit control are associated to typical objectives of typical stakeholder of GQM projects. The following stakeholder categories are discerned (10):

- Corporate management
- Project team
- GQM team

Corporate management provides project objectives and business-driven goals to both software development team and GQM team. They indicate what the market will be sensitive for regarding the software development activity. In the subsequent stages corporate management primarily has a facilitating role: providing adequate resources for the GQM project. A strong involvement of management in, for example, feedback sessions implies an enormous risk that individual engineers are appraised. Engineers will generally not support such a measurement project.

The *GQM team* and the *software development team* together translate the management goals into measurement goals and refine these goals into questions and metrics. This will primarily be based on context specific characteristics provided by the software development team and previous experiences. Consequently, the software development team provides the GQM team with measurement data. The GQM team processes these data and presents the results to software development team and corporate management.

4.3. Stakeholder objectives regarding feedback

As stated before, particular stakeholders have dissimilar objectives. In this section these objectives are described and linked to the two additions to the GQM method starting with requirements to feedback.

Feedback typically corresponds to the objectives of a software development team and a GQM team. Their emphasis is on understanding the problem domain, software and development process. Consequently, a quality improvement project that does not contribute to these objectives will not be supported by these stakeholders. Early feedback of measurement results is, therefore, also essential. This provides a first idea of whether the improvement project is indeed worthwhile for these stakeholders. Also, the explicit hypotheses formulation of anticipated measurement outcomes is essential here. As normally hypotheses and reality will differ, this illustrates that the measurement project actually contributed to the knowledge of the engineers. Because engineers learn throughout the entire measurement project, for example, through discussions with each other and through filling in measurement data, nobody will remember what they did not know at the beginning of the project. At the end of the project the engineers will react in a sense of 'we knew this all along' and might perceive the improvement project as a waste of time.

Early feedback and the formulation of hypotheses are elements of a quality improvement project that are typically associated to the objectives of the stakeholders: software development team and GQM team. Although these elements might appear to be trivial, they are often missing in practice. In a number of industrial cases we encountered, the absence of these elements could very well explain project failure.

Given the objectives of these stakeholders there will also be a risk of making the quality improvement project a goal itself, because in practice it will always be possible to learn more and further improve your working methods. However, the cost of acquiring this knowledge might exceed the benefits. Cost reduction will, for example, seldom be an objective of a software development team or a GQM team. If only the objectives of development team and GQM team are taken into consideration the chance of success of a quality improvement program are considered low, specially on the longer term.

4.4. Stakeholder objectives regarding cost/benefit analysis

Corporate management is the third stakeholder in a quality improvement project. For corporate management software quality is generally not a goal as such, however, should always be compared with the willingness of customers to pay for quality [29]. The addition of cost/benefit analysis, therefore, typically meets the objectives of corporate management. The link between a quality improvement project and yielding of customers paying for final products can, of course, be indirect and often requires many subjective judgements. However, in all the industrial cases we participated, making such a calculation was possible.

For example, in the industrial case of Section 3.1, the following cost and benefits were realized. In this project was investigated what the difference was in quality between re-used and newly developed code. In the GQM project was determined that re-used code was of substantial better quality newly developed code. On basis of this knowledge, the software development department decided to put more effort in the re-use of software and this effort would be more than regained during testing. The final quality of the software using the re-use oriented approach would be similar to the situation before, because previously one was able to repair faults in the later development stages. In this case the productivity gains were sufficient to justify the project. Additional

benefits were improved planning of projects (less repair) and more attention for quality. There was no need to quantify these benefits, because the project was already perceived as being beneficial based on the productivity gains. Through explicitly stating the additional benefits a better evaluation could be performed whether the quality improved project was experienced as valuable. An order of magnitude could be determined regarding the order of magnitude of the value of the project.

Such an analysis provides a common understanding of whether a measurement project is perceived as worthwhile, because money is a language that all stakeholders understand. As GQM projects always focus on an improvement area that is experienced as most valuable to investigate, a negative outcome of a cost/benefit analysis should be regarded as a serious warning regarding the usefulness of quality improvement. There is no reason to assume that the laws of economics do not apply to software quality improvement and that software improvement does not have some form of diminishing return [69].

In the beginning of this paper we referred to the importance of adding feedback requirements and cost/benefit analysis. The stakeholder analysis presented in this section supports this proposition. In this perspective, meeting the measurement goals only is regarded as inadequate result for measurement projects. *Learning* and *maintaining profitability* are too important to be left out. Stakeholder analysis also indicates that a software development team and GQM team are tended to exaggerate measuring.

5. Summary and conclusions

In the GQM method a systematic approach is represented for tailoring and integrating goals to models of software processes, products and quality perspectives of interest, based upon the specific needs of the project and the organization [10]. The result of the application of the GQM method is the specification of a measurement system targeting a particular set of issues and a set of rules for interpreting measurement data.

In this paper two additions to the GQM methodology were presented, being:

1. Feedback sessions. Frequent feedback sessions are identified as a mandatory element of any measurement program. Essentially, feedback sessions are a group learning process. On basis of learning theory, therefore, a conceptual model was defined of software development team and GQM team interaction in feedback sessions. Furthermore, nine learning enablers were defined to provide instruments to improve the learning conditions during feedback sessions.
2. Cost/benefit analysis. Cost and benefits of software measurement projects are in most organizations not made explicit [69], [24], [6], [7] and [54]. There are also few publications on cost/benefit analysis of improvement projects [69], however, from industrial experience we assume that a negative perception of the cost and benefits of measurement, particularly by management, is one of the most important reason to stop a quality improvement program [15], [40], [64] and [63]. To prevent individual misjudgments regarding the profitability of a measurement project an explicit analysis is recommended [69], [24] and [6]. In the industrial cases we examined, making such an analysis was always possible.

Stakeholder theory was used to illustrate the importance of both additions. The following stakeholder categories were discerned [9]:

- Corporate management
- Project team
- GQM team

The two additions to the GQM methodology could be associated to typical objectives of particular stakeholders. Software development team and GQM team are typically associated to the learning objective contained from feedback. Corporate management is typically associated to a profitability objective contained from cost/benefit analysis.

The analysis also illustrates that dominance of either stakeholder is most likely conflicting with the interests of the other stakeholders and presumably fatal for the measurement programs. An often heard reasons is that measurement programs are too costly, that their actual benefits are unclear and that the actual usability of the data for software project management is minimal [43]. Consequently, we assume that the absence of either one addition is an important failure factor of measurement projects. We will investigate this in further research, together with other insights stakeholder theory can provide to software measurement.

6. References

[1] Agarwal, R., Krudys, G., Tanniru, M., Infusing learning into the information system organisation, *European Journal of Information Systems*, No. 6, pp. 25-40, 1997.

[2] Albrecht, A.J., Gaffney, J.E., Software function, source lines of code, and development effort prediction: A software science validation, *IEEE Transactions on Software Engineering*, Vol. 9, No. 6, pp. 639-648, November 1983.

[3] Amabile, T.M., 'How to kill creativity', Harvard Business Review, September/October, pp.77-87, 1998

[4] Argyris, C, Schön, D.A., Organizational learning: a theory of action perspective, Addison-Wesley, 1978.

[5] Bach, J., The immaturity of the CMM, *The American Programmer*, September 1994.
[6] Banker, R, R. Kauffman, C. Wright and D. Zweig, Automating output size and reuse metrics in a repository-based computer-aided software engineering (CASE) environment, *IEEE Transactions on Software Engineering*, Vol. 20, No. 3, 1994, 169-187.
[7] Banker, R. and S. Slaughter, A field study of scale economies in software maintenance, *Management Science*, Vol. 43, No. 12, 1997, 1709-1725.
[8] Basili, V.R., The experimental paradigm in software engineering, In: *Experimental Software Engineering Issues*, Springer Verlag, LNCS#706, 1993.
[9] Basili, V.R., Caldiera, C., Rombach, H.D., Experience Factory, *Encyclopaedia of Software Engineering* (Marciniak, J.J., ed.), Vol. 1, John Wiley, 1994, 469 - 476.
[10] Basili, V.R., Caldiera, C., Rombach, H.D., Goal/Question/Metric Paradigm, *Encyclopaedia of Software Engineering* (Marciniak, J.J., ed.), Vol. 1, John Wiley, 1994, 528-532.
[11] Basili, V., Green, S., Software Process Evolution at the SEL, *IEEE Software*, July 1994
[12] Basili, V.R., Rombach, H.D., The TAME project: Towards Improvement Oriented Software Environments, *IEEE Transactions on Software Engineering*, Vol. 14, No. 6, June 1988, 758-773.
[13] Basili, V.R., Selby, R.W., Huthchens, D.H., 'Experimentation in Software Engineering', IEEE Transactions on Software Engineering, Vol. 12, No. 7, pp. 733-743, July 1986.
[14] Basili, V.R., Weiss, D.M., A methodology for collecting valid software engineering data, *IEEE Transactions on Software Engineering*, SE-10 (6), 1984, 728-738.
[15] Beach, L.R., Image theory: decision making in personal and organizational contexts, John Wiley & Sons, 1990.
[16] Birk, A., Solingen, R. van, Järvinen, J, Business Impact, Benefit, and Cost of Applying GQM in Industry: An In-depth, Long-term Investigation at Schlumberger RPS, *Proceedings of the 5th International Symposium on Software Metrics* (Metrics'98), Bethesda Maryland, November 19-21, 1998.
[17] Bøegh, J., Depanfilis, S., Kitchenham, B. and Pasquini, A. (1999), A Method for Software Quality Planning, Control, and Evaluation, *IEEE Software* 16(2), 69–77.
[18] Boehm, B.W. *Software engineering economics*, Prentice-Hall International, New Jersey, 1981.
[19] Briand, L.C., Differding, C.M., Rombach, H.D., *Practical guidelines for measurement based process improvement*, ISERN Report 96-05, 1996.
[20] Brooks, F.P., The mythical man month: essays on software engineering, Addison-Wesley, 1975.
[21] Bush, M.E., Fenton, N.E., Software Measurement: A conceptual framework, *Journal of Systems and Software*, Vol. 12, pp. 223-231, 1990.
[22] Card, D., Glass, R., *Measuring software design quality*, Prentice-Hall, 1990.
[23] Cavano, J.P., McCall, J.A., A framework for the measurement of software quality, *Proceedings of the Software Quality and Assurance Workshop*, 1978.
[24] Chidamber, S. and C. Kemerer, A metrics suite for object-oriented design, *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, 1994, 476-493.
[25] Cook, C.R., Roesch, A., Real-time software metrics, *Journal of Systems and Software*, Vol. 24, 1994, 223-237.
[26] Debau, C., J. Liptak and H. Schippers, Decision making for software process improvement: a quantitative approach, *J. of Systems Software*, Elsevier (new York), Vol.7, No. 26, 1994, 43-52.
[27] DeMarco, T., *Controlling software projects*, Yourdon Press, New York, 1982.
[28] Entwistle, N., *Styles of Learning and Teaching*, John Wiley & Sons, 1981.
[29] Fenton, N.E., Pfleeger, S.L., *Software Metrics: A rigorous and practical approach*, Thomson Computer Press, 1996.
[30] Garvin, D.A., Building a learning organization, *Harvard Business Review*, July-August, pp. 81-91, 1993.
[31] Genuchten, M. van, Why is software late? An empirical study of reasons for delay in software development, *IEEE Transactions on software engineering*, Vol. 17, No. 6, 1991.
[32] Gibbs, W.W., Software's Chronic Crisis, *Scientific American*, September, 86-95, 1994.
[33] Gilb, T., Principles of software engineering management, Addison-Wesley, 1994.
[34] Gilb, T., *Software Metrics*, Winthrop Publishers, 1977
[35] Goodman, P., *Practical implementation of software metrics*, McGraw-Hill Publishers, London, 1993.
[36] Grady, R.B., Practical software metrics for project management and process improvement, Prentice-Hall, 1992.
[37] Grady, R.B., Successfully applying software metrics, *IEEE Computer*, September, 1994, 18-25.
[38] Grady, R.B., Caswell, D.L., Software Metrics: Establishing a company-wide program, Prentice-Hall, 1987.
[39] Hall, T., Fenton, N., Implementing software metrics - the critical success factors, *Software Quality Journal*, No. 3, 1994, 195-208.
[40] Harrison, E.F., *The managerial decision-making process*, Houghton Mifflin Company, Boston (MA), Third Edition, 1987.
[41] Hatton, L., Automated incremental improvement of software product quality: a case history, In: *Software Quality: Assurance and Measurement A Worldwide perspective*, editors: Fenton, N., Whitty, R., Iizuka, Y., Thomson Computer Press, 1995.
[42] Heemstra, How expensive is software? (in Dutch: Hoe duur is programmatuur?), Kluwer (Deventer), 1989.
[43] Hetzel, B., The sorry state of software practice measurement and evaluation, In: *Software Quality: Assurance and Measurement A Worldwide perspective*, editors: Fenton, N., Whitty, R., Iizuka, Y., Thomson Computer Press, 1995.
[44] Humphrey, W. S., Managing the software process, SEI series in software engineering, Addison-Wesley, 1989.
[45] Jelinek, M., *Institutionalizing Innovation*, Praeger, 1979.
[46] Kan, S.H., Basili, V.R., Shapiro, L.N., Software Quality: An overview from the perspective of total quality management, *IBM Systems Journal*, Vol. 33, No. 1, 1994, 4-19.

[47] Kemerer, C.F., Progress, obstacles, and opportunities in software engineering economics, *Communications of the ACM*, Vol. 41, No. 8, August, 1998, 63-66.

[48] Kumar, K., Post implementation evaluation of computer-based information systems: current practices, *Communications of the ACM*, Vol. 33, No. 2, February, 1990, p. 203-212.

[49] Latum, F. van, Oivo, M., Hoisl, B., Ruhe, G., No improvement without feedback: experiences from goal oriented measurement at Schlumberger, *Proceedings of the 5th European Workshop on Software Process Technology* (EWSPT96), Nancy, France, Lecture Notes in Computer Science #1149, Springer Verlag, October 1996, 167-182.

[50] Latum, Solingen, Oivo, Rombach, Hoisl and Ruhe, Adopting GQM-based measurement in an industrial environment, *IEEE Software*, Jan/Feb 1998.

[51] March, J.G., Exploration and exploitation in organizational learning, *Organization Science*, Vol. 2, No. 1, February, 1991, 71-87.

[52] McGarry, F., Pajerski, R., Page, G., Waligora, S., Basili, V., Zelkowitz, M., *Software Process Improvement in the NASA Software Engineering Laboratory*, Technical Report, CMU/SEI-94-TR-22, Software Engineering Institute, 1994.

[53] Möller, K.H., Paulisch, D.J., Software Metrics: A practitioner's guide to improved product development, Prentice-Hall, 1993.

[54] Mukhopadhyay, T. and S. Kekre, Software effort models for early estimation of process control applications, *IEEE Transactions on Software Engineering*, Vol. 18, No. 10, 1992, 915-924.

[55] Nevis, E., DiBella, A., Gould, J., Understanding organizations as learning systems, *Sloan Management Review*, Winter 1995.

[56] Nonaka, I., Takeuchi, H., *The knowledge-Creating Company*, Oxford University Press, New York, 1995.

[57] Oivo, M., Bicego, A., Kuvaja, P., Pfahl, D., Solingen, R. van, *The PROFES methodology book and User Manual*, Http://www.ele.vtt.fi/profes/, 1999.

[58] Paulisch, D.J., Carleton, A.D., Case Studies of software process improvement measurement, *IEEE Computer*, September, 1994, 50-57.

[59] Pfleeger, S.F., Software Engineering: the production of quality software, McMillan Publishing, New York, 1991.

[60] Pfleeger, S.L., Lessons learned in building a corporate metrics program. *IEEE Software*, 10, 3, 1993, 67-74.

[61] Pfleeger, S.L., Rombach, H.D., Measurement based process improvement, *IEEE Software*, July 1994, 9-11.

[62] Putnam, L.H., Myers, W., *Measures for Excellence: Reliable software on time, within budget*, Yourdon press - Prentice Hall, New Jersey, 1992.

[63] Remenyi, D. and M. Sherwood-Smith, *Achieving maximum value from information systems*, John-Wiley (Chichester), 1997.

[64] Renkema, Th.J.W., *The IT value quest*, John-Wiley (Chichester), 2000.

[65] Senge, P.M., The fifth discipline: The art and practice of the learning organization, Doubleday, New York, 1990.

[66] Shaw, M., Prospects for an engineering discipline of software, *IEEE Software*, November 1990.

[67] Shepperd, M.J., Software Engineering Metrics, Volume 1: Measures and Validations, McGraw-Hill, 1993.

[68] Shepperd, M.J., Ince, D., *Derivation and Validation of Software Metrics*, Clarendon Press, 1993.

[69] Slaughter, S.A., D.E. Harter and M.S. Krishnan, Evaluating the cost of software quality, *Communications of the ACM*, Vol. 41, No. 8, August, 1998, 67-71.

[70] Solingen, R. van, *Product Focused Software Process Improvement: SPI in the embedded software domain*, BETA Research Series, Nr. 32, Downloadable from http://www.gqm.nl/, Eindhoven University of Technology, February 2000.

[71] Solingen, R. van, Berghout, E.W., The Goal/Question/Metric Method: A practical guide for quality improvement of software development, http://www.gqm.nl/, McGraw-Hill Publishers, 1999.

[72] Solingen, R. van , Berghout, E., Kooiman, E., Assessing feedback of measurement data: Relating Schlumberger RPS practice to learning theory, *Proceedings of the 4th International Software Metrics Symposium* (Metrics'97), Albuquerque, November 5-7, IEEE CS, 1997, 152-164.

[73] Solingen, R. van, Berghout, E.W., Latum, F. van, Interrupts: Just a minute never is, *IEEE Software*, September/October 1998.

[74] Stark, G., Durst, R.C., Vowell, C.W., Using Metrics in Management Decision Making, *IEEE Computer*, September, 1994, 42-48.

[75] Ulrich, D., 'Intellectual capital = competence x commitment', Sloan Management Review, pp. 15-26, Winter 1998.

[76] Weggeman, M., *Knowledge Management* (In Dutch), Scriptum Management, 1997.

[77] Weinberg, G.M., *Quality Software Management: Vol. 1: Systems Thinking*, Dorset House Publishing, New York, 1992.

[78] Weinberg, G.M., Quality Software Management: Vol. 2: First-Order Measurement, Dorset House Publishing, New York, 1993.

[79] Wijnhoven, F. van, Organizational Learning and Information Systems, 1995.

[80] Wohlwend, H., Rosenbaum, S., Schlumberger's software improvement program, *IEEE Transactions on Software Engineering*, Vol. 20, No. 11, 1994, 833-839.

[81] Zuse, H., Software Complexity: Measures and methods, De Gruyter, Berlin, 1991.