DECISIONS AND JUSTIFICATIONS IN THE CONTEXT OF INDUSTRIAL-LEVEL SOFTWARE ENGINEERING

FRANÇIS DION
ESI Software inc.
E-mail: FDion@esisoft.com
Phone: (514) 745-3311 ext. 283
Fax: (514) 745-3312

ALAIN ABRAN
Software Engineering Management Research Laboratory
Université du Québec à Montréal
C.P. 8888, Succ. Centre-Ville
Montréal, Québec, Canada
E-mail: abran.alain@uqam.ca
Phone: +1 (514) 987-3000 (8900)
Fax: +1 (514) 987-8477

ABSTRACT

Decision-making is a difficult task per se. This inherent difficulty is exacerbated by the complexity and fast pace of the changes that characterize software engineering. Critical decisions impacting the success of a project or even an entire organization must be made quickly based on information that is either limited to the point of being insufficient or so abundant that it is virtually unmanageable. Either way, the information is more often than not of questionable quality.

This paper proposes an evolutionary framework to support efficient and justifiable decision-making throughout the implementation phase. This approach covers the necessity to make decisions quickly without complete, reliable information, as well as integrate new data as it becomes available.

INTRODUCTION

This integration paper proposes an evolutionary framework to effectively support decision-making for software management. The proposed approach takes into consideration the quality of information, a typology of information sources, a structured hierarchy for the organization of available data, and an iterative design and validation process for data acquisition and use. An application of this approach is presented in an annex at the end of this document.

DECISION-MAKING IN THE CONTEXT OF SOFTWARE ENGINEERING

Making decisions is no easy task. The consequences can sometimes be enormous and are often poorly defined, misunderstood or even unknown. Nothing guarantees that past experiences will be of any use in the future, or that the latest fad is actually effective.

Decision-making in the field of software engineering is probably even more challenging. Contrary to other disciplines such as engineering or medicine, this industry has yet to develop or accept reliable validation and control practices. Novelty is perceived as an intrinsic indicator of value by suppliers (naturally), but also by practitioners and consumers. Consequently, new tools, techniques and methodologies are constantly being introduced. These ceaseless upheavals hamper all attempts to consolidate and tend to discredit lessons learned from past experience because it is generally believed that they would not apply to the "new" context.

Managers of software development initiatives must nonetheless routinely make crucial decisions in a context where there are many unknowns regarding the expected consequences. A few examples are given below:

- Would an object-oriented methodology be appropriate?
- Which language should be used?
- Which modelization tool?
- Add more resources, streamline the list of features or postpone delivery?
- Would increased code inspection have a positive or negative impact on the schedule?
- Would correcting a particular defect create more problems than it would solve?

Information – The Cornerstone of Every Decision

The quality of any decision depends on the available information. However, this information is more often composed of individual opinions than verified and verifiable facts. Taking into consideration the reliability of the information being used is an essential part of decision-making.

Zvegintzov (1998) proposed a generalized classification of information sources. In this classification, information sources are divided into six categories ranging from most credible and reliable ("A" rating) to rumors and folklore ("F" rating).

A. The systematic collection of real data. Systematic measures with precise procedures that are applied for a period of time long enough to allow a sufficient number of observations.

B. Questionnaire studies. They depend on individual opinions. Furthermore, the representativeness of the respondents must be assessed. Such studies mostly reflect recent events (2 to 12 months).

C. Isolated data points from observations.

D. Isolated data points from advocacy.

E. Statement by information providers, such as Gartner, Meta, Giga, CaperJones, H.Rubins. These groups rarely disclose their sources or methodologies.

F. Folklore or commonly-held ideas.

Few decisions in software engineering are based on "A"-rated sources. This is mostly due to the fact that this type of information is often simply unavailable. The systematic gathering of factual data involves significant costs and delays that not all organizations are able to or are prepared to incur. The fact that this information is available does not necessarily guarantee that it will be used or even considered, however. Responsible parties are often unaware of how to use it or believe that their project or situation is too unique for the information to apply.

Metrics and Measures

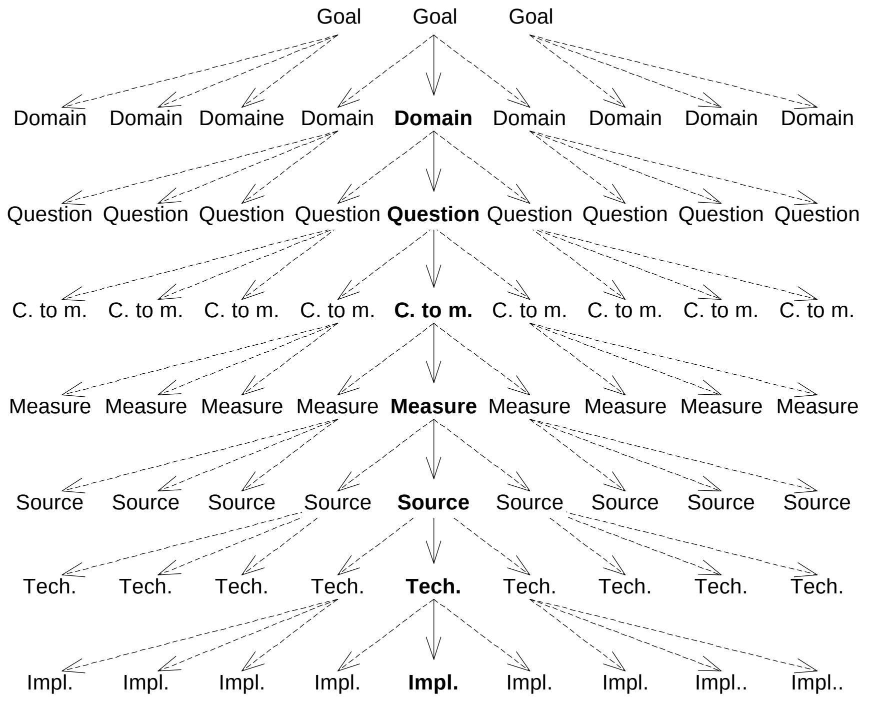
Numerous "metrics" have been proposed to evaluate various aspects of software and the software engineering process. Specific to the field of software engineering, this 'metrics' concept is neither clearly defined nor associated with a universally-recognized definition nor validation process (Jacquet & Abran, 1997). Choosing a particular measure or measurement model and interpreting its results can be rather problematic in this context. The more precise terms of measures, measurement methods and measurement models will therefore be preferred (Jacquet & Abran, 1997).

Gray and McDonnell (1997) proposed a measurement analysis framework that takes many related characteristics into consideration. The resulting model, known as the GQM++, contains up to 11 levels and provides the necessary support for a highly-detailed analysis of the measurement process. The authors' primary objective was to facilitate the timely detection of the cost effectiveness or applicability concerns encountered by many measurement programs.

The main characteristics of this model are given below.

- **Goals**. It is crucial to clearly identify the goal sought by a measurement program. This characteristic can be further detailed through sub-goals.
- **Domains**. It is wishful thinking to believe that a measure defined for a given domain could be applied to another domain without at least some level of adaptation. This characteristic can be further detailed in sub-domains in order to refine the context.
- **Questions**. Questions make it possible to identify exactly which questions the measurement programs should address. This characteristic can be further detailed in sub-questions.
- **Characteristic to be measured**. Identification of the concept to be measured (size, complexity, risk, etc.).
- **Measurement**. Size, for example, can be measured in terms of the number of lines of code or the number of objects — two vastly different methods for measuring the same concept.
- **Sources**. Where did the data used for the measurement come from?
- **Analysis techniques**. This characteristic refers to the methods (statistical or other) used to analyze the measurements results.

- **Implementation**. How each method will be used and applied, as well as what the expected benefits are.



The GQM++ model proposed by Gray and MacDonnell (Adapted)

The vertical and horizontal views provided by this model make it highly interesting and pertinent.

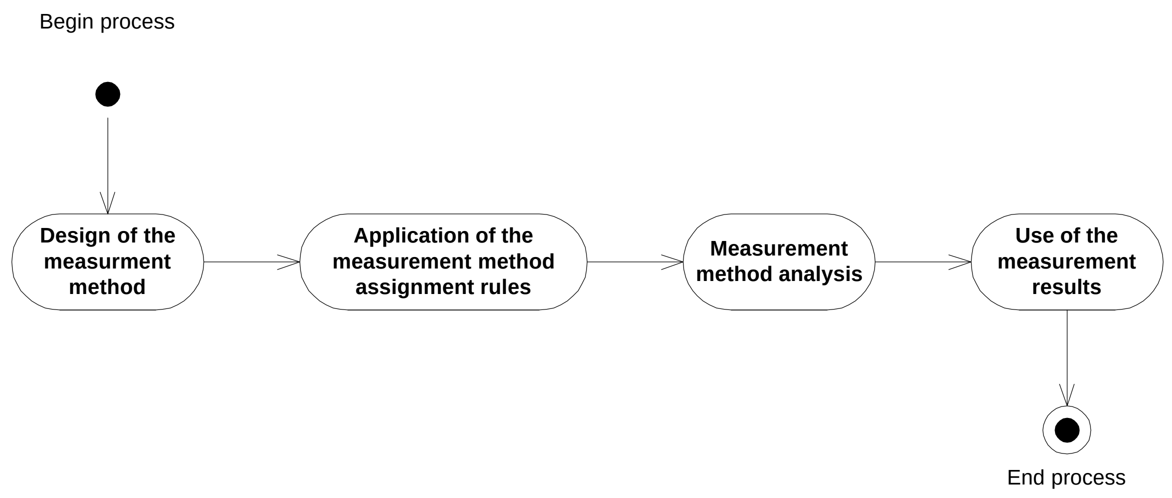
It can be read vertically to assess the completeness of the process. If every step of the ladder has been completed for a specific measure, its application can be considered thorough.

Horizontally, this model puts into perspective what has been accomplished... and what has not! A specific measure can only present a narrow aspect of any given situation. Typically-speaking, only a tiny fraction of the model's "branches" will be covered by a measurement program. The visualization of residual space should serve as a warning call and help convince involved parties that such partial results cannot be used as a basis for an immediate generalization.

This model can also facilitate cross-checking information and clarify the reuse of measures, analysis techniques and more.

A PROCESS FOR THE DESIGN, VALIDATION AND APPLICATION OF MEASURES

The GQM++ is a structure or framework for organizing and documenting measurement programs. Jacquet and Abran, cognizant that software measurements are regularly based on an informal and unverified methodology, present an integrated process for the design, validation and application of measures.


Begin process
Design of the measurment method
Application of the measurement method assignment rules
Measurement method analysis
Use of the measurement results
End process

The measurement process: A high-level model adapted from Jacquet and Abran

Even though this process was developed independently, a parallel can be established between it and the structure proposed by Gray and MacDonnell in the GQM++.

- **Design of the method of measurement**. This step covers the levels of the questions and the characteristics to be measured, as well as the measurements themselves.
- **Application of the assignment rules for the method of measurement**. This covers the levels of the sources and the application of the measure.
- **Analysis of the results**. Maps to the analysis techniques.
- **Application of the measurement results**. The implementation level.

THE ITERATIVE PROCESS AND DECISION SUPPORT

Jacquet and Abran's primary objective is to formalize the process of designing, validating and applying measurement methods; whereas Gray and MacDonnell strive to organize the relevant pieces of information. These complementary approaches are in fact attempts to bring software development to the level of a full-fledged engineering discipline where performance is quantifiable and results predictable. Although the possible benefits of these techniques are indisputable, their practical implementation could be problematic. Establishing a measurement program is a project in itself, with costs and implementation delays that can be substantial.

If a decision based on incomplete or unconfirmed information can have serious consequences, the same can be said about indecision. Not applying corrective action to an already-problematic situation simply because all the relevant information is not available or would be difficult to obtain can be as detrimental if not worse, especially if product quality or team productivity is affected.

The adaptations to the context of decision-making given below appear necessary:

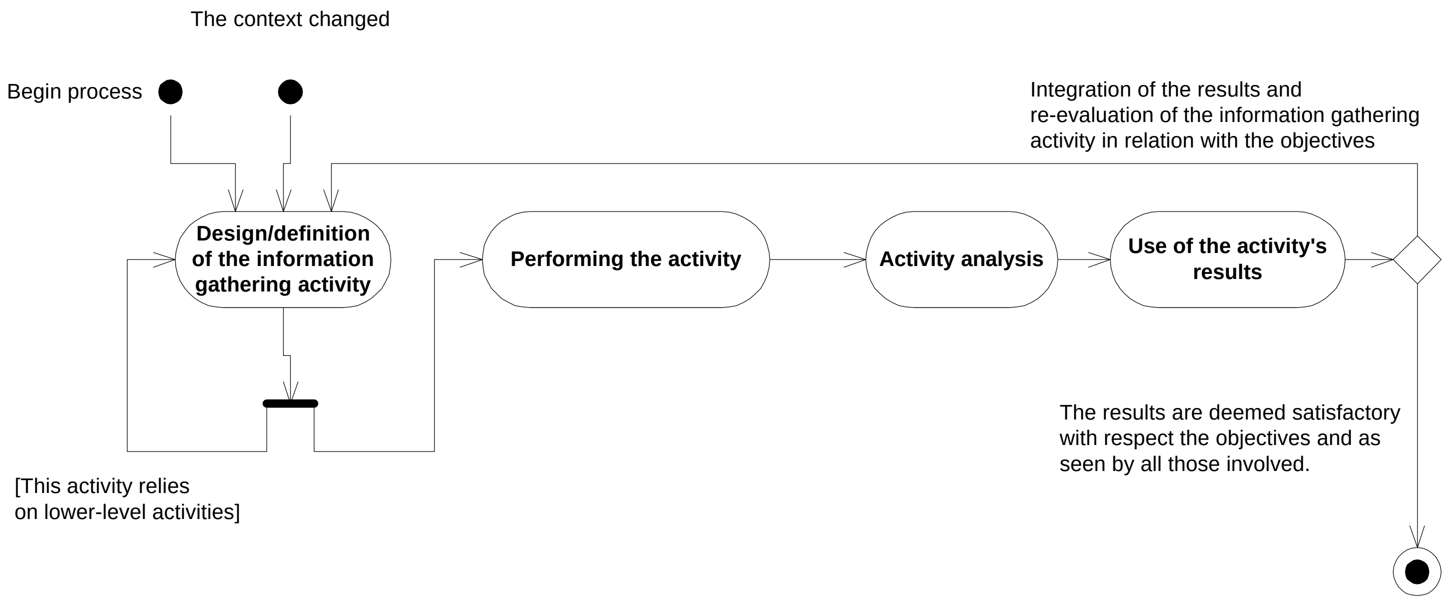
- Refine how the GQM++ model is to be used in an iterative context.
- Extend Jacquet and Abran's process to explicitly support composite measures.
- Extend the scope of these tools to cover other sources of information, such as expert opinions, in addition to measurement methods.

Sound decision-making requires having access as quickly as possible to a minimal amount of information – even when incomplete – as well as being able to integrate new data when it becomes available. As suggested by Gray and MacDonnell, this viewpoint would support regarding the GQM++ model as a dynamic sequence of approximations, each one built upon what was previously learned, rather than a static, definitive representation.

The process proposed by Jacquet and Abran can be embedded in another process where iterative and recursive aspects are explicitly integrated.

The evaluation of an attribute or situation often requires applying lower-level measurements for which the acquisition and integration process also needs to be precisely and unambiguously defined. The underlying hypothesis behind this "meta-process" is that the specifications of a higher-level measure should determine the choice of intermediary or atomic-level measures. Furthermore, the definition and application of "secondary" measures can and should trigger a re-evaluation of the initial measures. The result is a feed-back mechanism that could require several iterations before the network of measures stabilizes and final conclusions can be made.

An interesting aspect of this expanded process is that it promotes a gradual approach to the measurements and decision-making of complex phenomena. As such, a partial conclusion can realistically be drawn from partial results. This first observation can later be refined as more complete or reliable results become available.


The context changed
Begin process
Integration of the results and re-evaluation of the information gathering activity in relation with the objectives
Design/definition of the information gathering activity
Performing the activity
Activity analysis
Use of the activity's results
The results are deemed satisfactory with respect the objectives and as seen by all those involved.
[This activity relies on lower-level activities]

Information Meta-Process

Note that a second entry point has been added to the model to account for any condition (unforeseen difficulties, resource reallocation, a new offensive by the competition, etc.) that could lead to a change in the context. The link between the first two steps now has two branches: direct application of the measurement method, and recursive design of secondary measurement methods. The process is considered completed when the elements required to make the decision have been fully acquired, and not before. This decision is revised at the end of each iteration to take into account that interim results might be sufficiently convincing (one way or another) to warrant an immediate decision.

Jacquet and Abran's original process and Gray and MacDonnell's GQM++ model focus exclusively on measurements. In a decision-making context, however, measurements are only one of many available information sources. Because any one of these sources can have an impact on the usefulness and even the interpretation of the others, it only seems logical to integrate them in a unified model and process.

DEMONSTRATION OF THE PROCESS

The following example is based on actual events, but has been modified for the purpose of this demonstration.

The Scenario

A software engineering specialist is asked to find a solution to the poor quality plaguing the deliverables of a large corporation's IS Department. The problem has already been linked to the number of bugs still present in the modules when released by the Development Team to Quality Assurance.

While attending a conference, our engineer receives the same piece of advice from two different experts: set up an environment to automate unit testing. Although the idea seems reasonable, its development and application would require significant investment and represent a large-scale change in the department's operations.

At this stage of the analysis, our engineer completes the following GQM++ grid:

Goal: Improve the quality of the deliverables.

Sub-goal: Reduce the number of bugs detected by QA in order to lower the number of cycles before a system goes to production from 5 to 2.

Domain: Software development in the IS Department of a large corporation.

Sub-domain: A strategic software development project employing 30 engineers for a period of 3 years.

Questions	Sub-Questions	Characteristics to be Measured	Measures	Sources and Rating	Analysis	Application
Would automating unit testing help the Department achieve its objectives?	How much would automating unit testing cost?	Cost of developing the unit testing automation process and tools.				This measurement corresponds to the "fixed costs" of the project. Are they justified?
		Cost of original deployment and maintenance.	Experts' opinion: Not much when compared to all the benefits.	Martin Fowler (C) Bruce Eckel (C)		This measurement corresponds to the "variable costs" of the project. Crucial to the overall profitability.
	What would be the impact on the quality of the deliverables?	Quality enhancement ratio.	Experts' opinion: No single action would have a greater positive impact.	Martin Fowler (C) Bruce Eckel (C)		Expected "benefits."
	Will the team members accept this new way of operating?					Never underestimate the human factor!

Our engineer classified these sources as type "C" (isolated observations by independent observers) in part because he realizes that they personally have nothing to gain by advocating this technique. He is satisfied with the information he has collected thus far and decides to proceed with the feasibility analysis.

The next step is to assess as accurately as possible the real costs of developing this solution. The evaluation is carried out by inspecting similar environments found on the Web and analyzing historical data gathered locally from projects of comparable size and complexity. The results obtained are compatible with the cost parameters allocated for R&D.

After developing a prototype for the unit testing automation framework, the concept must be validated through a pilot project where deployment costs and the actual quality enhancement ratio may be estimated. The updated version of the analysis grid is presented below.

Goal: Improve the quality of the deliverables.

Sub-goal: Reduce the number of bugs detected by QA in order to lower the number of cycles before a system goes to production from 5 to 2.

Domain: Software development in the IS Department of a large corporation.

Sub-domain: A strategic software development project employing 30 engineers for a period of 3 years.

Questions	Sub-Questions	Characteristics to be Measured	Measures	Sources and Rating	Analysis	Application
Would automating unit testing help the Department achieve its objectives?	How much would automating unit testing cost?	Cost of developing the unit testing automation process and tools.	20 days	Paired comparison (C) Historical data (A)	The results are acceptable.	Proceed with development.
		Cost of original deployment and maintenance.	Experts' opinion: Not much when compared to all the benefits.	Martin Fowler (C) Bruce Eckel (C)		This measurement corresponds to the "variable costs" of the project. Crucial to the overall profitability.
			Ratio of automation time to initial development time: 20%	Pilot (A- and B)	The real cost should decrease as team members familiarize themselves with the unit testing automation framework.	The observed cost falls within the additional effort that has been budgeted for unit testing.
	What would be the impact on the quality of the deliverables?	Quality enhance-ment ratio.	Experts' opinion: No single action would have a greater positive impact.	Martin Fowler (C) Bruce Eckel (C)		Expected "benefits."
			Ratio of bugs detected by the automation process to the number of known bugs: 15%	Pilot (A-)	Manual unit testing had already been performed on this module.	Should be compared with the result of the reverse situation (automated testing before manual testing.)
			Ratio of bugs that could not have been discovered by any other method of testing to the number of known bugs: 5%	Pilot project (A-)	Mostly non-functional bugs.	Suggests a positive impact on long-term maintenance.
	Will the team members accept this new way of operating?					Never underestimate the human factor!

The "A-" rating was given to all measurements from the pilot project because, although they were obtained through a formal numerical process, the data did not come from a large sampling of modules and did not cover a sufficient length of time to draw definitive conclusions.

Once again satisfied by the results, our engineer decides to present his findings to the Development Team for immediate implementation. Despite a generally favorable response, however, it was decided to extend the experimentation phase. Some team members remarked that the measurement of the cost of implementation did not cover the second part of the question (the cost of maintenance). Others, although not contesting our engineer's findings, wonder if alternative quality enhancement techniques such as formal inspections could offer a better cost/benefits ratio.

Consequently, another iteration of the evaluation process was started. New questions were added to the grid and additional characteristics were considered in order to better address the questions and comments raised.

Assessment of the Scenario

This example gave us an outline of the proposed process for the continuous acquisition and validation of the decision-support data. We saw how new information was incorporated in the grid, as well as how this representation allowed the various people involved to understand and intervene in the process. More importantly, however, we saw how the measurements themselves were actually merged with other sources of information such as experts' opinions.

Conclusion

We began by presenting an assessment of the decision-making process in the context of software development. Many decisions made in this context are based more on intuition and hunches than on formal measurement. The high importance of information quality and the reliability of sources as defined by Nicolas Zvegintzov was then presented, followed by a discussion on metrics and their not-so-obvious applicability. The GQM++, a model proposed by Gray and MacDonnell, is an attempt to give a solid foundation to measurement programs. Jacquet and Abran proposed a process for the design, validation and application of measures. A parallel was established between this process and the GQM++ model. An extension of these approaches to integrate them in the larger context of decision-making was then proposed. This extension places emphasis on iterative aspects, and covers other forms and sources of information in addition to measurements. A concrete example was then presented to illustrate how this process can be practically applied.

This proposition is interesting from two different angles. On one hand, it makes it possible to combine external information sources, internal measurements and historical data in a single model, and then make comparisons between them. On the other hand, its iterative and progressive nature allows it to efficiently support decision-making even in situations that are urgent or complicated by a number of uncertainties. Thus, the tight schedules and numerous unknowns that still plague this industry should no longer be regarded as obstacles to rational and accurate decision-making.

ACKNOWLEDGMENTS

We wish to thank the Bell Canada and the Natural Sciences and Engineering Research Council of Canada for research funding. The opinions expressed in this paper are solely those of the authors.

REFERENCES

[1] Jacquet, J.P. & Abran, A., "From Software Metrics to Software Measurement Methods: A Process Model", *Third International Symposium and Forum on Software Engineering Standards, ISESS'97*, Walnut Creek (CA), June 2-6, 1997.

[2] Zvegintzov, Nicholas, "Frequently Begged Questions and How to Answer Them", *IEEE Software*, Vol. 15, no 2, March/April 1998, p. 93-96.

[3] Gray, A. & MacDonnell, S.G., "GQM++ A Full Life Cycle Framework for the Development and Implementation of Software Metric Programs", *Australian Software Measurement Conference*, Canberra, November 1997.