# Development of an OO Energy Management System using the ami Approach

**Heinz Dobler**
CD Laboratory for Software Engineering
Linz, Austria
Tel: ++43 (0)7236/3343-205
Fax: ++43 (0)7236/3343-24
eMail: heinz.dobler@fhs-hagenberg.ac.at

**Angelika Mittelmann**
VOEST-ALPINE Stahl Linz GmbH
Linz, Austria
Tel: ++43 (0)732/6592-9159
Fax: ++43 (0)732/6592-9138
eMail: angelika.mittelmann@ps.stahl.voest.ada.at

## Abstract

VOEST-ALPINE Stahl Linz GmbH is a producer of flat steel situated in Linz, Austria. Planning the design and implementation of an energy management system using object-oriented technology, we decided to introduce process improvement in software development in order to overcome problems with the new technology at a very early stage. We followed the four ami phases closely because we had no experience in using a process model. Summing up the benefits, we now have a homogeneous software engineering know-how within the project team, written standard procedures, and better organized as well as excellent documented projects.

## Keywords

Capability Maturity Model (CMM), Energy Management System, Object-oriented Analysis and Design (OOA, OOD), Object Modeling Technique (OMT), C++.

## Author's Experience and Expertise

**Heinz Dobler** studied computer science, worked for six years as assistant at the University of Linz on problems of programming language implementation and tool building. In his doctoral thesis he concentrated on software reengineering using source-to-source translation and in the past few years he studied and used object-oriented techniques in several projects. Since 1993 he is a member of the CD Laboratory for Software Engineering in Linz and since 1995 he is professor at the Polytechnic University for Software Engineering in Hagenberg.

**Angelika Mittelmann** studied computer science, was employed as a systems programmer at VOEST-ALPINE responsible for capacity planning, performance management, tuning and accounting of MVS mainframe computers for eight years. Afterwards she was involved in the information strategy planning, building a corporate model and developing detail strategies in the fields of UNIX, CAD, CASE, and AI. For the last two years she was the project and metrics promoter of the ESSI Project 10024 introducing the ami process model.

## 1. Background Information

### 1.1. Company, Project Team, and Subcontractor

VOEST-ALPINE Stahl Linz GmbH (VASL) is an integrated iron and steel plant, situated in Linz, Austria. The software activities of the company are partly dezentralized. On one hand there is a major department concentrating on development of production planning and supervision systems as well as on activities pertaining to financial transactions. On the other hand there are several small groups of process engineers situated at the main production plants.

The project team for the development of the energy management system (EMS) consists of six to eight persons coming from both groups. At the beginning of the EMS project they were mostly skilled Fortran and partly C programmers doing their development work individually or in very small groups, having no experience in object-oriented software development.

The subcontractor of the EMS project is the Christian Doppler (CD) Laboratory for Software Engineering, an institution situated at the University of Linz. The CD Laboratory members mainly provide know-how transfer, bringing the state of the art of object-oriented software engineering to the industry.

### 1.2. Objectives

The objective of the application experiment is to test and evaluate advanced object-oriented technology in order to evolve from the situation of experienced Fortran and C programmers working in various small project groups using little to no standards at all, to a situation where

- an object-oriented method and language is used,

- prototyping-oriented and object-oriented life-cycle models are executed,
- effective software quality assurance procedures are installed,
- project management methods are used, and
- programming guidelines and documentation standards are implemented.

The baseline EMS software development deals with forecasting and optimization of energy production and consumption in an integrated steel plant. In this project an object-oriented language and the life-cycle model are used for the first time to a greater extent in our company.

## 2. ami Process Model

The acronym **ami** stands for **a**pplication of **m**etrics in **i**ndustry and for **a**ssess/**a**nalyse, **m**etricate and **i**mprove. The ami process model was developed in the ESPRIT project 5494, has been tested in several projects all over Europe, and is documented in the ami handbook [AMI]. It consists of four phases, each of them having three steps. The basis for ami is the capability maturity model (CMM) published by the Software Engineering Institute (SEI) of the Carnegie Mellon University [SEI93a, SEI93b].

The ami method ensures that quantitative approaches are used to achieve relevant company objectives and to improve the software development process. It especially includes the definition and use of adequate metrics. A metric is the measurement of a characteristic of a product or process and can be classified as objective or subjective depending on whether the data is the result of a counting process or subjective evaluation against a certain scale.

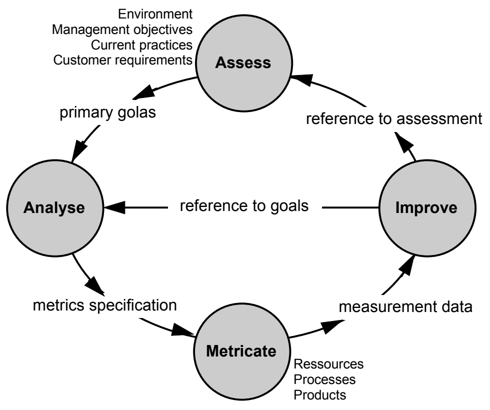

**Figure 1: ami Process Model**

The twelve steps comprising the ami method read as defined in the following table 1:

| Phase | Nr. | Activity |
|---|---|---|
| **Assess** | 1 | Assess the environment |
| | 2 | Define primary management goal(s) |
| | 3 | Validate primary goal(s) |
| **Analyse** | 4 | Break down management goal(s) into sub-goals |
| | 5 | Check the consistency of the resulting goals tree |
| | 6 | Produce a table of questions to identify metrics |
| **Metricate** | 7 | Write and validate the measurement plan |
| | 8 | Collect primitive data |
| | 9 | Verify the primitive data |
| **Improve** | 10 | Distribute, analyse and review the measurement data |
| | 11 | Validate the metrics |
| | 12 | Relate the data to goals and implement actions |

**Table 1: Phases and Activities of the ami Method**

## 3. Project History

In order to install a continuous improvement program in software development, the project managers decided to follow strictly the ami method as described in the ami handbook. Since the ami approach would be an overkill being applied to one project only, the whole organizational unit of VASL, where the EMS project is situated, was included in the ami cycle.

Starting with an informational meeting (subject "What is ami and why ami?") where the team members were informed on the ami approach and the goals of the project, the expectations of the team members concerning the results of the project as well as the obstacles that might hinder achieving the goals were worked out. In the following we will discuss the implementation of ami in our organization.

## 3.1. Phase "Assess"

After the opening session eight projects were selected to participate in the ami program. According to the ami process model they were first assessed with aid of the CMM. For this reason, the CMM questionnaire contained in the ami guide was translated into German and adapted for our use. Especially the technical terms had to be defined according to the organizational understanding. The questionnaire contains 85 questions, each of them assigned to one of the five maturity levels (Level 1: *initial*, Level 2: *repeatable*, Level 3: *defined*, Level 4: *managed*, and Level 5: *optimized*) and to one of the catagories "organization", "resources, personnel, and training", "technology management", "documented standards and procedures", "process metrics," "data management and analysis", and "process control." The questions are all *yes/no* answers. Questions labeled with a hash sign (#) are deemed to have a slightly greater importance at their level. Everyone starts at level 1. To reach level 2, 80 % of all questions denoted L2 and 90 % of all questions denoted L2 # must have yes answers. To reach the next level all questions of the previous levels must have yes answers and the above mentioned percentages for the level in question must hold.

We are at maturity level 1. The results of our self-assessment (see figure 2) enable us not only to identify our maturity level but also the main problems of our organization, which trigger the definition of the management goals.

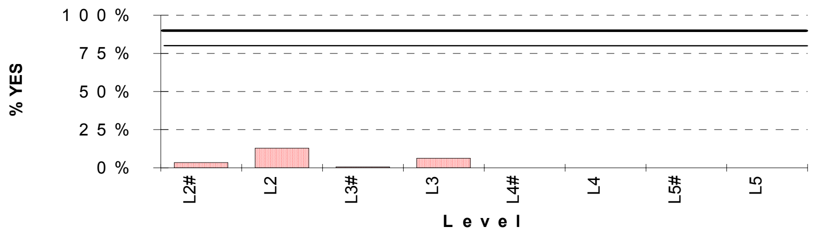

**Figure 2: Results of First Assessment**

The assessment results led to the definition of the following management goals: (1) improve insight into the software engineering process, and (2) record project productivity. During an in-depth team discussion, the management as well as the team members validated and accepted the defined primary goals. When we detected during the assessment, that there was a lack of understanding in the business processes involved in software development we immediately started with training activities in order to improve the understanding of all team members in the processes of software engineering (software development process, project management, configuration management, and quality assurance).

## 3.2. Phase "Analyse"

Breaking down the primary goals into sub-goals during the following team discussions, we also checked the consistency of the resulting goals tree (see figure 3).

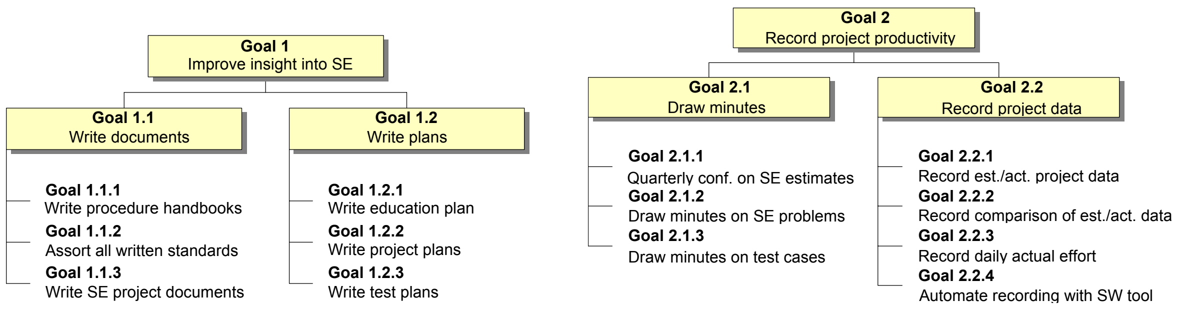
Goal 1
Improve insight into SE
Goal 1.1
Write documents
Goal 1.2
Write plans
Goal 1.1.1
Write procedure handbooks
Goal 1.1.2
Assort all written standards
Goal 1.1.3
Write SE project documents
Goal 1.2.1
Write education plan
Goal 1.2.2
Write project plans
Goal 1.2.3
Write test plans
Goal 2
Record project productivity
Goal 2.1
Draw minutes
Goal 2.2
Record project data
Goal 2.1.1
Quarterly conf. on SE estimates
Goal 2.1.2
Draw minutes on SE problems
Goal 2.1.3
Draw minutes on test cases
Goal 2.2.1
Record est./act. project data
Goal 2.2.2
Record comparison of est./act. data
Goal 2.2.3
Record daily actual effort
Goal 2.2.4
Automate recording with SW tool

**Figure 3: Goal Trees of Primary Goal 1 and 2**

We especially took care of the amount of operational goals. Therefore our goals tree contains two primary goals, four goals at the second level, and thirteen operational goals. Using the goal/question/metric method [Basi88] and the entity templates explained in the ami handbook we worked out the metrics for every goal.

## *3.3. Phase "Metricate"*

The precise specifications of the defined metrics were documented in the measurement plan in order to enable every team member to understand the meaning and use of every metric. The structure of a measurement plan as outlined in the ami handbook fitted well in our purpose. According to the process model the specification is divided into two parts. Part A contains the metric definitions and analysis procedures, Part B the primitive data definitions and collection procedures.

Since the assessed organizational unit as well as the performed projects were at the initial maturity level, we introduced so-called *existence metrics* besides the necessary qualitative and quantitative metrics. This kind of metric measures whether or not all needed documents for a well-organized software development process exist. They are mainly of qualitative and subjective type. Table 2 shows a list of our existence metrics.

| Nr. | Metric |
|---|---|
| 1 | Existence of Procedure handbooks (SE, PM, CM, QA) |
| 2 | Existence of Project documents |
| 3 | Existence of Education plan |
| 4 | Existence of Project plans |
| 5 | Existence of Test plans |
| 6 | Existence of Software Engineering process minutes |
| 7 | Existence of Project minutes |
| 8 | Existence of Test minutes |
| 9 | Existence of tool for Project Management and its usage |

**Table 2: Existence Metrics**

Table 3 contains the quantitative respectively the qualitative metrics we defined. For a complex metric, built of one or more primitive metrics, the primitive metrics are given in brackets.

| Nr. | Metric |
|---|---|
| 1 | Project Effort (Development, Specification, Rework, Quality) |
| 2 | Project Development Time |
| 3 | Project Type |
| 4 | Project Complexity |
| 5 | System Size |
| 6 | Number of Changes |
| 7 | Number of Document Pages |
| 8 | Number of Project Collaborators |
| 9 | Number of Training Days |
| 10 | Maturity Level (in terms of CMM) |

**Table 3: Qualitative/Quantitative Metrics**

For all projects included in the ami initiative, the primitive data were collected using the collection procedures defined in the measurement plan. Afterwards the primitive data was validated and verified by the project managers using their historical data.

### *3.4. Phase "Improve"*

Using the analysis procedures as defined in the measurement plan, the measurement data was analyzed. The presentation and the review of the data as usually was done during a team meeting. The validation could easily be done by looking up their graphical representation. In the following, the data was related to the goals and the team members decided the implementation of the following actions:

- Documentation of all needed procedure handbooks.
- Daily recording of project development effort.
- On-line availability of document templates for project plan and status report, project metrics, customer requirements, and problem reports.
- On-line help for the measurement plan and all procedure handbooks.
- Collection and analysis of defined metrics periodically according to the measurement plan.

### *3.5. Second Assessment*

As the application experiment had a limited time scale we decided to re-assess the organization after one year. Normally, SEI assessments are repeated in a two-year-cycle, because improving the organization from one maturity level to the next one takes about two years. The results of the second assessment (figure 4) show that the CMM makes it possible to measure and visualize software development process improvements. The reason why our improvements are spread from Level 2 to 4 is that our primary goals do not lead directly to Level 2.

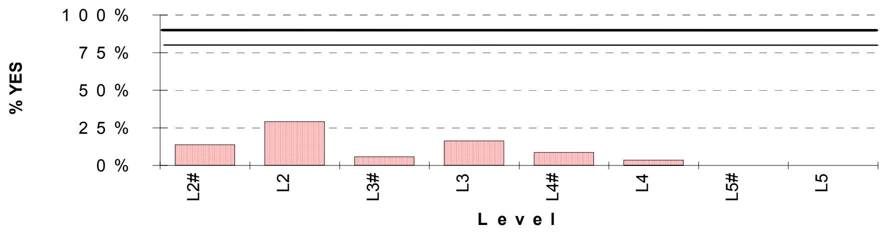

**Figure 4: Results of Second Assessment**

A comparison of the results of the first and second assessment shows the effect of our goal definitions on the key areas of the software development process. In case of redefinition of primary goals these results might be used, too.

## 4. Lessons Learned

Overall, the main objectives of the application experiment presented in the first chapter have been met. Table 5 lists a comparison of aims and results.

| Nr. | Aim | Result |
|---|---|---|
| 1 | Usage of OO method and OO language | method: in part, language: yes |
| 2 | Execution of prototyping and OO life-cycle | yes |
| 3 | Installation of effective SQA procedures | some |
| 4 | Usage of project management methods | yes |
| 5 | Usage of prog. guidelines and doc. standards | yes, especially for documentation |

**Table 5: Comparison of Aims and Results**

Some comments concerning the results: (1) The OO method involved (OMT) proved intuitive and has been used for analysis, design, and documentation – but not for code generation. The selection of C++ as an OO implementation language proved to be a success, although intensive training was necessary. (2) Prototyping has been used from the very beginning of system implementation. (3) The effectiveness of the SQA procedures (mainly code review) is questionable. (4) Programming guidelines have been defined, their usage is not checked strictly, but documentation standards are strictly imposed by tools (e.g. DocToHelp).

In the following sections, we give some more details of the lessons learned structured in four groups: ami process model, metrics, human/organizational factors and OO technology.

### *4.1. ami Process Model*

The goal-driven approach of ami turned out to be very useful, because the link of the metrics and agreed actions to the goals improves the understanding of the whole process. The CMM makes it possible to measure the progress of the improvement program in a practical manner. The wording of a measurement plan turned out to be a necessity in order to enable its permanent use. The only problem encountered was that the individual goals and metrics set is not transferable without changes to any other organizational unit even of the same company. The reason for this is that every organizational unit might suffer from different problems and therefore only other goals and metrics can lead to the desired effect.

### *4.2. Metrics*

If the maturity level of the organization is relatively low (less or equal to 2), it is advisable to use a greater amount of qualitative metrics. They are easier to understand by people inexperienced with metrics. Getting familiar to metrification, some of the qualitative metrics might be changed to quantitative ones. We also developed a special form of qualitative metrics which we call existence metrics. These metrics check, whether necessary documents for an orderly execution of the software engineering process and all related processes in software engineering exist in an adequate form. They are mainly of qualitative and subjective nature.

When metrics are introduced first, there is no possibility of comparisons. Therefore they are less useful at the beginning, but keeping them for a reasonable period of time makes them very useful for later projects. In this connection it must be emphasized to use the same metrics set for a reasonable period of time in order to get statistically relevant data.

### *4.4. Human/Organizational Factors*

It cannot be overemphasized that management involvement is the key critical success factor for installing ami in an organization. A second prerequisite is to find an engaged employee accordingly skilled in software engineering as well as in project management to play the role of a metrics and project promoter. For convincing the team members to buy-in the new techniques the opinion leaders have to be won first. Enough budget and time should be planned for training activities involving all team members.

External skilled help by persons experienced in object-oriented analysis and design accelerates the know-how transfer. We also found that the software development process is much better understood, if it is well documented. Gaining the necessary documentation by all team members intensifies this effect. The prepared document templates facilitate the documentation of user requirements and problems as well as the reporting of the project plan and status.

Organizational restructuring during project life-time makes it difficult to achieve the defined goals at first sight. It is still too early to decide on all effects of the restructuring, but it is clear that it offered the chance to spread the ideas worked out in the ESSI Application Experiment within the company. At least one other project (implementation of a gas pressure control system in the steel production process) has been successfully brought to an end using our approach. So it will be used in future projects as well.

### *4.5. Object-oriented Technology*

We decided to use Rumbaugh's Object Modeling Technique (OMT) for analysis and design, supported by the tool Select OMT with its extensions of the Jacobson method (use cases). It seems that this combination is superior to the original OMT, because use cases are more easily understood and modeled than state transition diagrams. Although the tool is a low-cost one, it is easy to use and very well suitable for design documentation. Code generation still suffers from difficulties of translating class relationships.

The GUI builder we are using is object-oriented and obviously allows very rapid development and usage of inheritance in an elegant manner. The only disadvantage is the "dissipation" of code fragments, which is inherent to the object-oriented technology itself. Provided that they are fairly complicated, the absence of a good class browser makes it difficult to gain an overview of the class relationships and also the inheritance hierarchy.

For C++ programming on the Unix, we use a class library offering container classes and data types, and which supports task communication, access to RDB's and garbage collection.

## Acknowledgment

We want to thank the reviewer for the encouragement and the useful comments.

## References

[AMI] ESPRIT project 5494: ami application of metrics in industry, Metric Users' Handbook. Eigenverlag ami User Group, o.O. o.J.

[Basi88] Basili, V.R.; Rombach, D.: The TAME project. Towards improvement-orientated software environments. In: IEEE Trans Soft Eng, 14 (1988) 6, S. 758 - 773.

[SEI87] Humphrey, W.; Sweet, W.: A Method for Assessing the Software Engineering Capability of Contractors, Technical Report, CMU/SEI-87-TR-23. SEI Carnegie Mellon University, Pittsburgh 1987.

[SEI93a] Paulk, Mark C.; et al.: Capability Maturity Model for Software (Version 1.1), Technical Report, CMU/SEI-93-TR-24. SEI Carnegie Mellon University, Pittsburgh 1993.

[SEI93b] Paulk, Mark C.; et al.: Key Practices of the Capability Maturity Model, Version 1.1, Technical Report, CMU/SEI-93-TR-25. SEI Carnegie Mellon University, Pittsburgh 1993.