

Mathematical Models- Based Software Modernization



Neli Maneva, Software Engineering
Department, IMI, BAS

Kraicho Kraichev, Musala Soft Ltd

Nikolay Grozev, Musala Soft Ltd

In jeder Naturlehre ist nur so viel "eigentliche"
Wissenschaft, als darin Mathematik anzutreffen ist.

Immanuel Kant

In every department of physical science
there is only so much science, properly so-called,
as there is mathematics.

Immanuel Kant

Presentation Overview

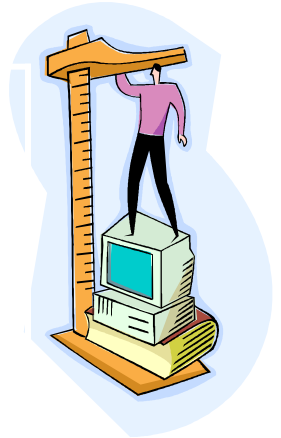
- Software Engineering: Science and Practice
- The MMM (Management through Models and Metrics) – approach to SE activities
- A Case Study: The MMM - approach to Software Modernization
- Conclusion

INTRODUCTION



Software Engineering: Science and Practice

Software engineering – an interdisciplinary area, mixture of scientific, technological, managerial, etc. methods applied in order to assure an **efficient** process for producing **high quality** software products.



Current observation: Only **10%** of the developed and validated scientific methods are in real-life usage

Possible explanations:

In SE science: sophisticated, too formal and unclear methods description without transfer procedures and supported (automated) tools;

In SE practice: too many projects with insufficient resources - financial, human (both in number and qualification), time, etc.

Our MMM – approach to SE activities

Management through Models and Metrics - an attempt to assure continuously improved software development cycle under a stated *real-life problem* and *goals* to be achieved

Main characteristics

- Scientific – applies some validated formal and rigorous methods;
- General – can be used in any SE activity, identified as significant;
- Flexible – can be tuned to a different application context.

A feasible procedure for practical use

Step 1. Analyze the activity under consideration

Step 2. Repeat the following cycle within the planned resources:

<modeling - application of models - estimation through metrics>

Step 3. Identify and describe in a standard way a set of best practices for this SE activity.

A Case Study: Software modernization

Software erosion: *the constant decay of the internal structure of a software system that occurs in all phases of software development and maintenance.*

Symptoms

- Maintenance costs are rising
- Development projects run over-time and/or over-budget
- User dissatisfaction

Causes

- Changing requirements
- Changing environment
- Poor documentation
- Lack of knowledge
- Employee turnover
- Inadequate management and/or software development practices

A stepwise MMM-procedure for software modernization

Step 1. Study the software modernization activity and define:

- Goals to be achieved;
- Scope of the activity;
- Goal-oriented artifacts, for which a sequence of formal models should be created

Step 2. Repeat the next activities within the available resources:

- Modeling
- Application of models
- Estimation of models, performance, achievements (through a set of metrics)

Step 3. Incremental construction of the set of best practices for software modernization

Mathematical models

- I. Determine the available artifacts and classify them, e.g.
 - Derived – executable code, database dump
 - Basic – programs source code , database data definition scripts
 - Unstructured / Informal – human targeted documentation

- II. Define the scope and complexity level of the performed analysis

- III. Decompose the identified artifacts into components
 - Determine the granularity depending on the objective
 - The result is a set of software components

Mathematical models

IV. Classify the obtained components on the base of their type characteristics:

1. Programs – according to:
 - Programming language – COBOL, Assembly, C/C++, Java, etc.
 - Execution environment – CICS, batch, .NET, App server, etc.
 - etc.
2. Data
 - Relational
 - Hierarchical
 - Object-oriented
 - Unstructured
 - etc.
3. Documentation
 - Internal – architecture, detailed design, admin guide, dev guide
 - External – user guide, tutorials, etc.
 - etc.

Mathematical models

v. Build a model – a higher level of abstraction

All the steps below are automated

1. Assign a parser to each component category
 - Formal grammar
 - Domain-specific rules
2. Run the parser
3. Build the model
 - Cross reference the model instances of the different types of components
 - Annotate with human readable text and graphics

Mathematical models

VI. Evaluate

1. Assess the degree of software erosion for each component
2. Select the components to be analyzed further

VII. Improve

1. From the relatively simple...
 - dead code elimination
 - copy-paste section elimination
2. ... to the complex ...
 - refactoring of an entire sub-system
3. ... to the extreme ...
 - in some cases rewriting from scratch is the only viable option

VIII. Evaluate again

1. Get valuable feedback what the effect was
2. Plan the next iteration accordingly

Practical validation of the approach

Scope - the IT system of a large manufacturing company

Object

- IT system, based on mainframe technology
- 8 million lines of source code
- Developed and maintained over 30 years

A modernization project has been successfully completed

- Proof-of-technology in a pilot run
- Modernization of the entire IT system

Results

- Significant decrease in IT operating and maintenance costs
- More flexible and future-proof IT environment

CONCLUSION AND FUTURE WORK



Conclusion

Basic results

- A feasible approach to certain SE activities has been developed
- The approach was validated in a real-life project for a selected activity – software modernization

Ideas for further research and development

- to improve the modeling process and its application
- to continue the experiments with software modernization, including the third step - the description of identified best practices
- to design a framework for metrics implementation and interpretation of the results obtained
- to increase the level of automation supported by our software tools
- to improve the human-computer interface of our tools

Authors and Acknowledgements



Assoc. Prof. PhD Neli Maneva,
Software Engineering Department, IMI-BAS
neli.maneva@gmail.com



Kraicho Kraichev, Musala Soft Ltd,
kraicho.kraichev@musala.com



Nikolay Grozev, Musala Soft Ltd,
nikolay.grozev@musala.com

Questions



Thank you!