10TH INTERNATIONAL DESIGN STRUCTURE MATRIX CONFERENCE, DSM'08
11 – 12 NOVEMBER 2008, STOCKHOLM, SWEDEN

INTRODUCTION OF SOFTWARE RELATED DSMS TO SOFTWARE ENGINEERS (A CASE STUDY)

Han van Roosmalen
D2Groep BV

Keywords: Software architecture, software engineering, DSM Introduction, experiences

1 INTRODUCTION

Introducing a new method of engineering to software developers is not always an easy task. In the software development world a number of methods and processes exist and even these are most of the times not followed to the letter. A minority of software developers, at least in The Netherlands, is proficient with the Unified Modelling Language (UML) [1] or neither are stringent adopters of an established process model like: Waterfall, Rational Unified Process [2], or any of the Agile development processes like XP, Scrum [3] or Essential Unified Process that gain popularity.

However today's software systems tend to become more complex even with their initial release and tend to become even more complex over time. In many cases software systems are not sufficiently documented and basic knowledge about application structure and development guidelines are missing. Maintenance problems tend to get even bigger when the principal engineer leaves the project or even the company.

Therefore it is necessary to introduce a method that provides insight in the software system without too much effort and without the need for special skills. In many cases a software architecture rediscovery [4], [5] undertaken with the DSM can be helpful. Software architecture rediscovery is the process to reconstruct the structure of a system based on inspection of the code base.

2 REAL LIFE CASES

The method of introduction as described here is based on my personal findings within a number of companies, amongst others some of these are:

A company developing firmware for optical drives;
A company producing pick-and-place machines;
A large company developing high-end medical imaging applications;
A large music company;
A medium sized company making Point-Of-Sales terminals and related software;
A large company making telecom network management and control software.

3 INTRODUCTION OF THE SOFTWARE DSM

Introducing the methodology of a software DSM to a group of (software) engineers is done in a number of steps. At first the group is introduced to the concepts of the DSM, thereafter a representative software applications' code base is analysed in conjunction with the chief software architect. The results of this activity is shown to the group and discussed in detail. A plan of action for improvement of the analysed system is set-up at the end of this session. By following this route the engineers are trained to use DSMs in their own environment. In all cases it is noted that improved insight in the application structure is achieved and vivid discussions based on DSMs are held only a few hours after the training started.

3.1 Introduction of the main concepts

All software developers are introduced to the main concepts within less than an hour. In most cases this is sufficient to provide a basic understanding of how to read and correctly interpret software DSMs. During the introduction a number of examples are presented and discussed to some extent. The session ends with a demonstration of the capabilities of the DSM tool Lattix.

3.2 Working with the software architect/architecture

After the introduction most participants leave and a real example is prepared by the software architect. A representative software application is selected and transferred into the matrix. A number of hours, depending on the internal complexity and analysability of the application are spent on structuring the DSM. This is done by defining abstract layers and modules that are (or should be present) in the software architecture of the application. The result of this action provides the desired or logical software architecture composition in the left column of the DSM with the matrix shows the realised architecture. When the software architect is not available one of the participants is asked to fulfil this task. If the participant is not able to draw a component diagram of the application a diagram from the documentation is used. The participant is guided through the functions of the DSM tool and will receive some hands-on experience.

When the DSM structuring is finished the results are discussed and some of the findings, both major and minor are noted. Improvement scenarios are executed without making any change to the applications' codebase.

3.3 Presenting the software architects' findings

If sufficient insight in the DSM is gathered the findings of the previous activities are presented to all the participants of the first session. In all cases a considerable amount of enthusiasm of the participants shows when they see their application in a DSM format. A number of problem areas in the DSM are discussed with the participants and in a short amount of time deep technical discussions over larger and minor disturbances are filling the air. In many situations the discrepancies between the architects' intend with the architecture and the real implementation is elaborated and insight in the application grows by the minute. After some hours solution scenarios are discussed.

Depending of the complexity of the application a full session takes a day or two.

3.4 Short term results

Based on a session as described above a number of short term results are obvious, such as:

Improved communication;

Because of its implicit simplicity insightful discussions start almost immediately when the DSM is structured in a digestible format.

Reasons for changing the (project) organisation;

In two of the cases the DSM let to a non-software action, being a major change in the project organisation to better cope with the implemented architecture.

Reasons for postponing a release;

In another case a number of essential issues let the project manager decide to postpone a release to first concentrate on fixing those issues.

Reasons for cancellation of a project/product;

In one case the complexity of the system had the team decide to cancel the project and start over with a better designed architecture.

3.5 Long term results

For the long term the development teams where able to set-up an environment in which they were able to better manage the software architecture. By doing so they were able to prevent amongst other things further erosion of the software architecture (e.g. deviation from the designed software architecture). Beside the top down approaches they were also able to use the DSM for performing impact analysis with which they were able to better estimate a proposed change in the application. In one case an enterprise architect used the DSM output of Lattix to incorporate this information his Enterprise Architecture tool. By doing this he was able to get almost real-time information of the company's databases in the companies IT assets. He complained that he liked the output of the Enterprise Architecture tool to be in a DSM format in stead of the provided graphs.

4 CONCLUSIONS

Training software developers and architects using the DSM method is fairly easy. The learning curve of applying a DSM to software is low in contrast with other better-known methods like UML.

Although previous results are positive not all companies that D2Groep addressed were convinced that the DSM was applicable to them. Even in situations in which D2Groep was sure that the DSM would be extremely helpful the developers were rejecting. This could be because many developers close their eyes for more formal engineering methods, because the DSM method is insufficiently promoted as a best practise way of engineering, or the lack of sexiness.

REFERENCES

[1] Rumbaugh J., Jacobson I. and Booch G. The Unified Language Reference Manual, Second Edition, 2005.

[2] Kruchten P. The Rational Unified Process: An Introduction, Third Edition, 2004.

[3] Kniberg H. Scrum and XP from the Trenches, 2007

[4] Clements P. et al. Documenting Software Architectures: Views and Beyond, 2003

[5] Roock S., Lippert M. Refactoring in Large Software Projects *Performing complex restructurings successfully*, 2006

Contact: Han van Roosmalen
D2Groep BV
Hakgriend 42
3371 Hardinxveld-Giessendam
The Netherlands
Phone: +31-184-621 232
Fax: +31-184-630-183
han.van.roosmalen@d2groep.nl
www.d2groep.nl

MANAGE COMPLEX SYSTEMS

FOLLOW THE FLOW OF INFORMATION!

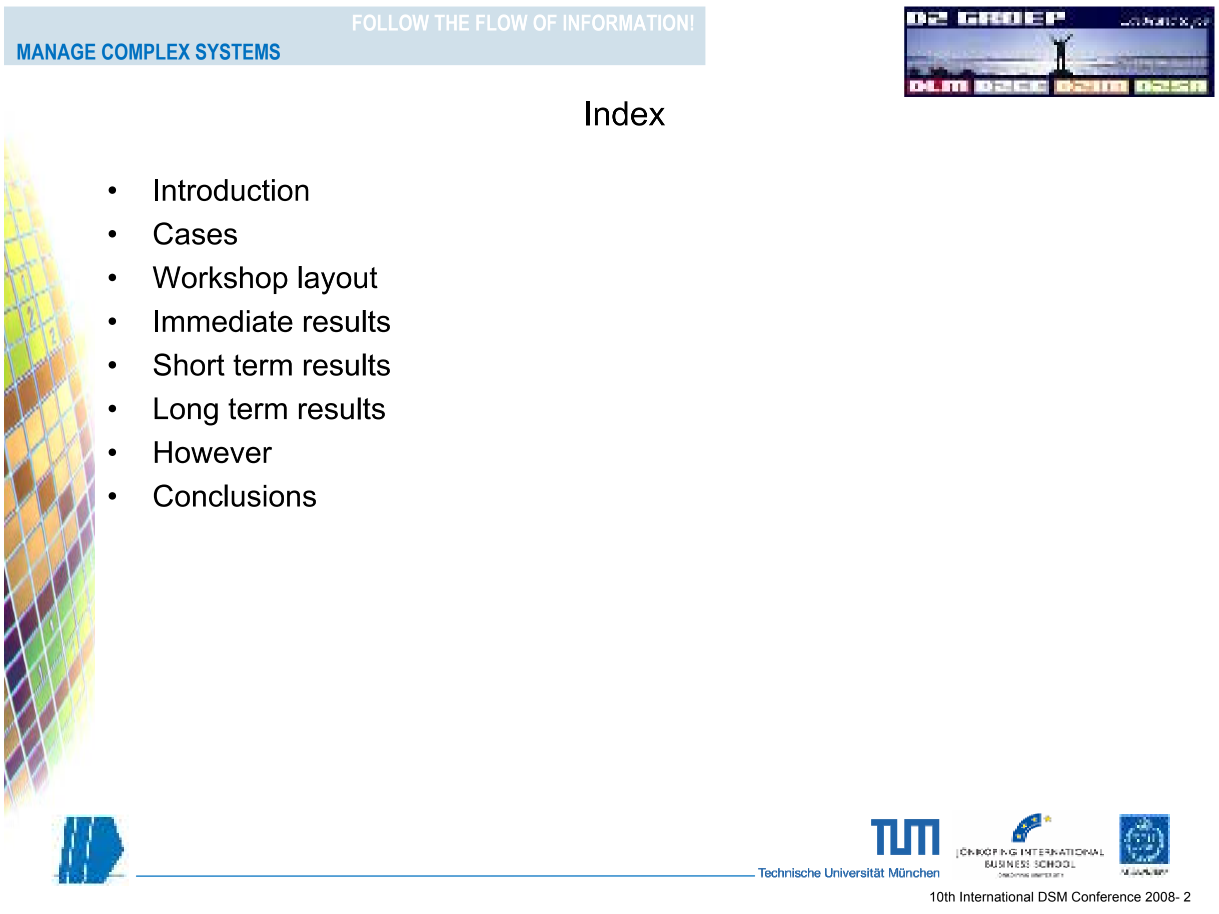
10TH INTERNATIONAL DSM CONFERENCE

Introduction of software related DSMs to software engineers (A case study)

Han van Roosmalen

D2 Groep BV

Technische Universität München



MANAGE COMPLEX SYSTEMS

FOLLOW THE FLOW OF INFORMATION!

Index

- Introduction
- Cases
- Workshop layout
- Immediate results
- Short term results
- Long term results
- However
- Conclusions

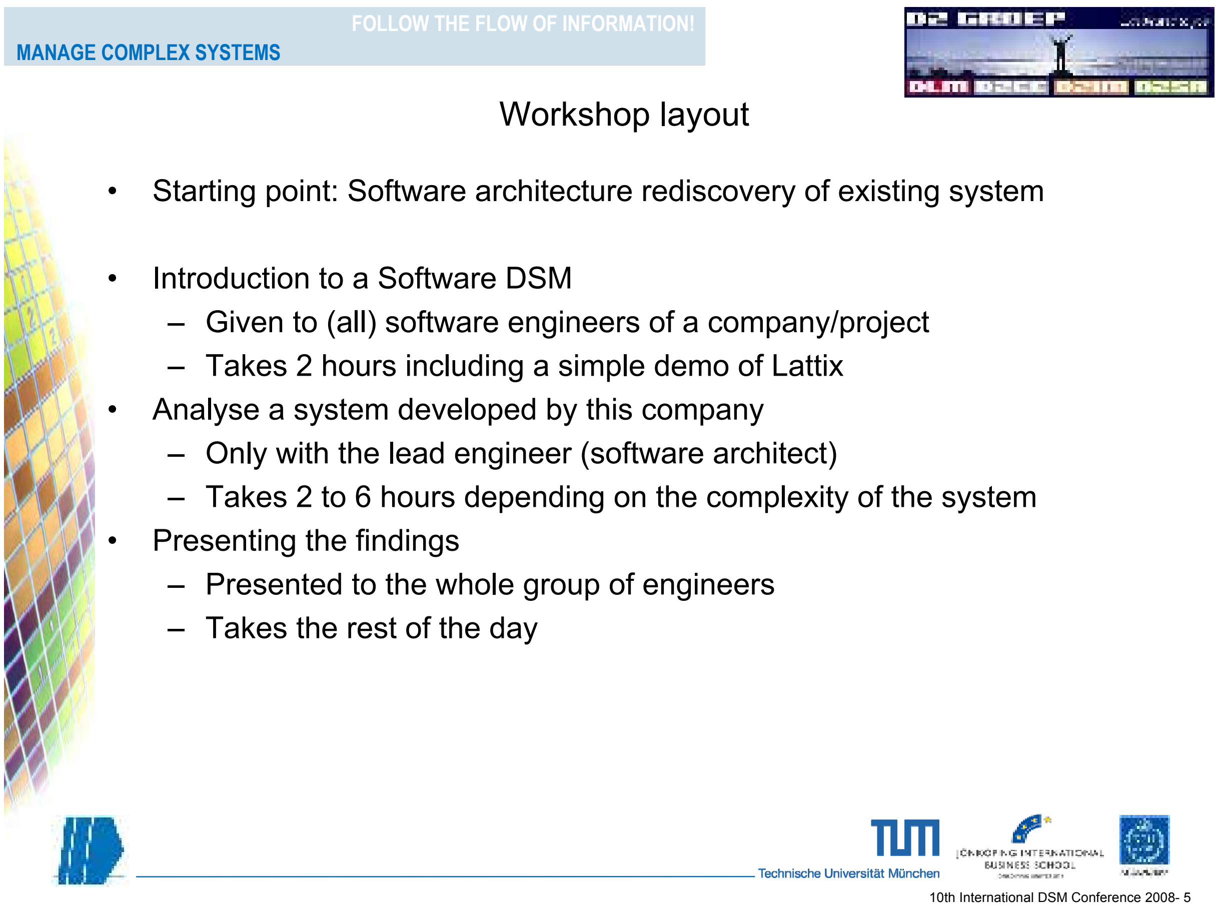
Technische Universität München

Introduction

- Observations:
  - Software system complexity
  - More legacy systems are created everyday
  - Each software project wants to follow its own engineering process
  - Domain knowledge people leave and newbie's come
  - Lack of documentation
  - Increasing maintenance costs
  - Many software applications are build and maintained with “screwdrivers”
  - No insight in application structure/behaviour
- Requirements:
  - Improved insight
  - “Communication visuals”
  - Easy to learn and use software engineering tool
- DSM solves some of these problems
- Important note: Software DSMs show structure not sequence!

Cases

- DSMs introduced to some companies:
  - Firmware development for optical disks
  - Embedded software development for pick-and-place machines
  - Software for medical systems
  - Music company
  - Point-Of-Sales terminal and application manufacturer
  - Facility management software manufacturer
- Interested in DSM for very different reasons


FOLLOW THE FLOW OF INFORMATION!

MANAGE COMPLEX SYSTEMS

Workshop layout

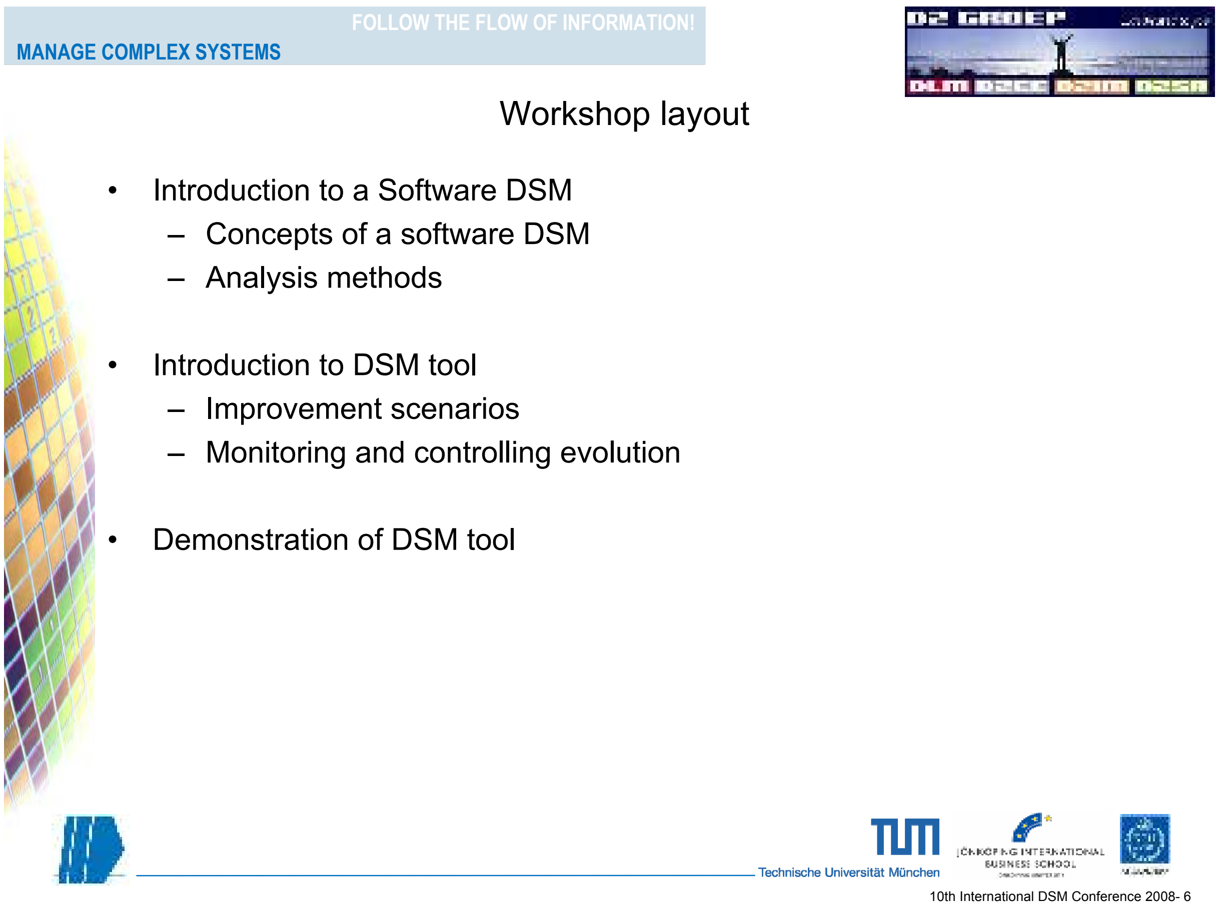
- Starting point: Software architecture rediscovery of existing system

- Introduction to a Software DSM
  - Given to (all) software engineers of a company/project
  - Takes 2 hours including a simple demo of Lattix
- Analyse a system developed by this company
  - Only with the lead engineer (software architect)
  - Takes 2 to 6 hours depending on the complexity of the system
- Presenting the findings
  - Presented to the whole group of engineers
  - Takes the rest of the day

Technische Universität München

JÖNKÖPING INTERNATIONAL BUSINESS SCHOOL

10th International DSM Conference 2008- 5


FOLLOW THE FLOW OF INFORMATION!

MANAGE COMPLEX SYSTEMS

Workshop layout

- Introduction to a Software DSM
  - Concepts of a software DSM
  - Analysis methods

- Introduction to DSM tool
  - Improvement scenarios
  - Monitoring and controlling evolution

- Demonstration of DSM tool

Technische Universität München

JÖNKÖPING INTERNATIONAL BUSINESS SCHOOL

10th International DSM Conference 2008- 6

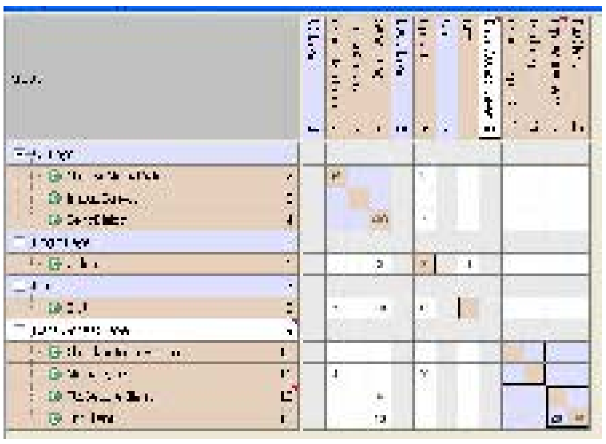
Workshop layout

- Analyse a system developed by this company/project
  - Goal: Visualisation of desired vs. realised architecture
  - Software architect decides on software application (part)
  - Software application code base/database schema is loaded in Lattix
  - Playing around with “buttons, tabs and menus”
  - Modelling desired software architecture
  - Moving modules/classes/tables into desired software architecture
  - Find weak points in architecture

Workshop layout

- Presenting the findings
  - DSM of analysed system is presented
  - Software architect pinpoints to problems areas
  - Zoom in and out into specific parts of DSM
  - Heated discussions

  - I leave at 5 pm



Immediate results

- During Introduction
  - Simplicity of the DSM draws everybody's attention
  - Highly interactive sessions
- During Software Architecting Session
  - Hands-on experience by software architect
  - Generation of ideas when DSM is being structured
  - “I knew that something was wrong there!”
  - “Okay, I have to talk to that guy!”
  - “Oh no, that is not the way it should have been done!”
- During End presentation
  - Everybody is standing pointing to cells in the DSMs
  - Going into details while retaining overview
  - Insight leads to discussions that make sense
  - This day was too short

Short term results

- Improved team communication
- Reason to change (project) organisation
- Reason to postpone release
- Reason to get extra funding
- Reason to cancel project

Examples of short term results

- Change in global project organisation
- Impact analysis improvement
- Thought-out small refactoring(s)
- Introduction of DSM (tool) to other projects

- Rejection because of political hurdles
- Disbelieve in two ways:
  - I did not know our application was that rotten!
  - This approach is too simplistic!

Long term results

- Performing better impact analysis
- Preparation of (large) refactoring(s) of a code base
- Preventing software architecture erosion
- DSM usage becomes part of the engineering practise
- DSM type visualisation is asked for in more software engineering tools

Examples of long term results

- Introduction of DSM (tool) to other projects
- Making DSM (tool) part of software factory
- Better management of software architecture
- Monitoring and control of off-shored development

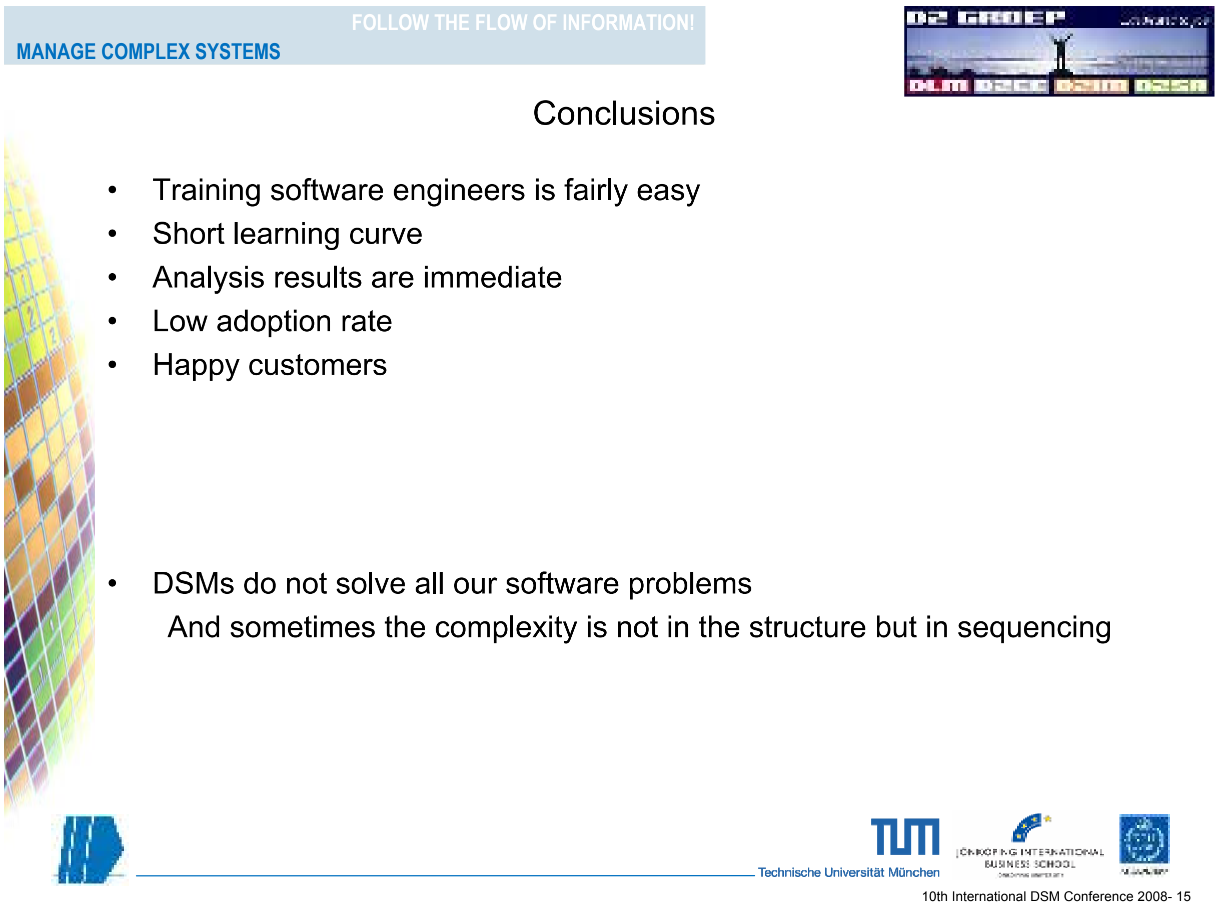
However

- It is difficult to find a company that sees the benefits
  - Where is the visionair (and budget holder)?

- What they say:
  - Again another approach!
  - That's not the way we work we model!
  - We can do without that!
  - We have other problems to take care of!
  - We have Eclipse/Visual Studio!

- Struggle
  - Is DSM as sexy as UML?
  - Is DSM promoted sufficiently?

- Problem in software land (google):
  - DSM stands for Domain Specific Modelling (UML)



FOLLOW THE FLOW OF INFORMATION!

MANAGE COMPLEX SYSTEMS

Conclusions

- Training software engineers is fairly easy
- Short learning curve
- Analysis results are immediate
- Low adoption rate
- Happy customers

- DSMs do not solve all our software problems
  And sometimes the complexity is not in the structure but in sequencing

Technische Universität München

10th International DSM Conference 2008- 15