Overview

- Research questions
- Part 1 - Related work in software maintenance
- Part 2 - Related work in software engineering maturity models
- Part 3 - Proposed maturity model
- Part 4 - Case studies, conclusions and future work
Research questions

1- If software maintenance is a specific domain of software engineering what are its specific processes and activities?
2- Are the unique processes of software maintenance well reflected in the current international standards?
3- Is there an existing maturity model proposal that covers the entire set of software maintenance unique activities?
4- What would be the proposed architecture of a capability maturity model that could address the entire set of software maintenance unique activities?
5- How can such a model be used in practice to support the improvement of software maintenance?

Part 1 - Related work in software maintenance
1 - Small maintenance?

1- Modification requests come in more or less randomly, and cannot be accounted for individually in the annual budget planning process;

2- Modification requests are reviewed and assigned priorities, often at the operational level – most do not require senior management involvement;

3- The maintenance workload is not managed using project management techniques, but rather queue management techniques;

4- The size and complexity of each small maintenance request are such that it can usually be handled by one or two maintenance resources; ISBG and UKSMA → 5 days or less

5- Priorities can be shifted around at any time, and requests for corrections of application errors can take priority over other work in progress.

1 - Issues of maintenance

Top 10 issues by priority [Parik02 and Dekleva92]

1  Managing changing priorities
2  Inadequate testing techniques
3  Difficulty of measuring performance
4  Absent and/or incomplete software documentation
5  A large backlog of requests
6  Difficult to measure the maintenance team contribution to the organization
7  Low morale of maintenance personnel
8  Not many professionnals in the domain
9  Little methodology, few standards, procedures and tools specific to maintenance
10 Source code in existing software is complex and unstructured
1 - New body of knowledge

Co-edited: SWEBOK chapter 5 -> Software Maintenance (www.swebok.org)

Fundamentals: Definitions, need, nature and costs
Key Issues: Technical, management and measurement
Maintenance Process: Processes and activities
Techniques for maintenance: Program comprehension, reengineering, reverse engineering and impact analysis

1 - Need for SM\textsuperscript{mm}

\textbf{CMM and CMMi focus}
- Software Development and Maintenance Projects
- Teams of developers

\textbf{Software Maintenance Specific Processes (SWEBOK)}:
- Transition
- Service Level Agreements
- Acceptance/Rejection of Change and Corrective Requests
- Planning Maintenance activities
- Supporting operational software
Part 2 - Related work in software engineering maturity models

2- How to build a maturity model

1. Understand the knowledge area
2. Look in standards to find processes, activities and best practices
3. Look to Framework and SWEBOK to create domains and KPAs
4. Decide practices to be included in the model and their maturity level
5. Build or Refine the model Architecture
6. Find a site and conduct case studies
7. Review the content with Experts
8. Improve model as necessary
2- How to validate a mat. model

- ISO/IEC 15504 (SPICE) - 1994
  - 35 Case studies (20:Europe, 14:Pacific Rim and 1: Canada);
  - Questionnaires (3), rating forms & report (3);
  - Demographic Analysis and questionnaire analysis.

- IT Service CMM (Dr. Niessink) - 2000
  - 2 case studies (a quick scan, a 3 day on-site assessment);
  - Questionnaires analysis, KPA discussions.

- CM³ Corrective maint. MM (Dr. Kajko-Mattsson) - 2001
  - 17 case studies (14 non-ABB, 3ABB);
  - Checked if the CM³ proposed processes are present or absent ex: 14/17 document their problem management process;
  - Checked 10 control activities as well.

2- Why the CMMi Architecture?

- Contains the essential elements of effective processes for software related activities
- Contains a framework that provides the ability to generate multiple models and associated training and assessment materials. These models may represent:
  - software and systems engineering
  - integrated product and process development
  - new disciplines
  - combinations of disciplines
- Provides guidance to use when developing processes

Source: P.Croll: 14th Annual DoD Software Technology Conference - IEEE-Sponsored Track - 1 May 2002
2- What current MM could help?

Year | Software Engineering CMM proposals
--- | ---
1991 | Bootstrap
1992 | Trillium
1993 | CMM©
1994 | Camélia, automated testing (Kra94)
1996 | TMM (Bur96), Zit96, Dov96
1997 | Som97
1998 | Esi98, Top98, Baj98
1999 | Wi99, Vet99, Sch99
2000 | Cob00, Str00, Bev00, Lod00
2001 | Kaj01d & 01e, Ray01, Sch01, Luf01, Tob01, Sri01
2002 | CMM©, Nie02, Mul02, Vee02, Pom02, Raf02, Sch02, Ker02, Cra02

2- Chosen based on criteria

<table>
<thead>
<tr>
<th>Model</th>
<th>Criterion 1</th>
<th>Criterion 2</th>
<th>Criterion 3</th>
<th>Criterion 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMM© [Sei02]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ISO 9001:2000 Interpretation [Boo90, Lod01]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Zitouni [Zit96]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>CMM [Kra94]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Bootstrap [Boo91]</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>TeleSpice and R-Spice [Esi98b]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Trillium/Camélia [Cam94, Tri96]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Checklist Testing Maturity Model [Vee92]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Testing Maturity Model [Bur99]</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Maturity Model (Automated Testing) [Kra94]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Outsourcing Management Maturity Model [Kra92]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>IT Service Capability Maturity Model [Nie04]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Business-IT Alignment Maturity Model [Luf01]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>IT Management, Control and Audit Maturity Model [Ca0066]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Change Management Capability Model [Nie04]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Publicly available, detailed, alive, all industri
Part 3 – Proposed software maintenance maturity model

3 - Standards used in SM\textsuperscript{mm}

Standards, are consensus-based documents that codify best practice. Consensus-based standards have seven essential attributes that aid in process engineering. They:

- represent the collected experience of others who have been down the same road,
- tell in detail what it means to perform a certain activity,
- can be attached to or referenced by contracts,
- help to assure that two parties have the same meaning for an engineering activity,
- increase professional discipline,
- protect the business and the buyer,
- improve the product.

Source: P.Croll: 14th Annual DoD Software Technology Conference - IEEE-Sponsored Track – 1 May 2002
3- CMMi structure in SM\textsuperscript{mm}

- Contains the essential elements of effective processes for software related activities
- Contains a framework that provides the ability to generate multiple models and associated training and assessment materials. These models may represent:
  - software and systems engineering
  - integrated product and process development
  - new disciplines
  - combinations of disciplines
- Provides guidance to use when developing processes

Source: P.Croll: 14th Annual DoD Software Technology Conference - IEEE-Sponsored Track - 1 May 2002

3- Sources to build SM\textsuperscript{mm}

- ISO/IEC 15504 (Spice)
- ISO/IEC 12207, Information Technology Software Life Cycle Processes
- ISO/IEC 14764, Software Engineering, Software Maintenance
- ISO/IEC 1219, Standard for Software Maintenance

- Zitouni/Abran Software Maintenance
- Camélia model
- CM\textsuperscript{3} Corrective Maintenance Maturity Model
- ITIL Service Management
- IT Service Continuity Management
- Cobit
- Malcolm Baldrige

- CMMI® Software Eng. v1.1 Process Area/Specific Practice

- Framework
- Standards
- Best Practice guides and Maturity Models
3- SM\textsuperscript{mm} context (Scope)

![Diagram showing the context of Software Development, Service Level Agreement, Maintenance services, and other related processes.]

3- SM\textsuperscript{mm} proposed process model

![Diagram illustrating the proposed process model with stages such as Pre-delivery and Transition, Event and Service Request Management, and monitoring processes like Operational Support, Changes, and Rejuvenation.]

University of Luxembourg
Ecole de technologie supérieure
Otto-von-Guericke Universität Magdeburg

19

20
3- SMmm proposed process model

- The process model was presented to ISO/IEC 14764 to be used to update the current process model. Their answer was:

This comment is out of scope with the NWI which is to merge the content of the IEEE 1219 into ISO/IEC 14764. This comment will be placed in a "deferred comment database" for the future revision. This revision will be started by a NWI proposal, subject to approval by SC7 and JTC 1.'

3- Influencing factors

- Software Maintenance
- Software Product
  - Size
  - Age
  - Maturity
  - Composition
  - Application type
  - Quality caract.
- Maintenance Processes
  - Type
  - Input
  - Outcome
  - Performance
  - Quality caract.
- Maintenance Resources
  - Type
  - Description
  - Requirements
  - Performance
  - Quality caract.

Kitchenham et al
At first, we define the software product as a (software) system as:

\[ SP = (M_{SP}, R_{SP}) = (\text{artefacts}, R_{SP}) \]

\[ = (\{\text{programs}, \text{documentations}\}, R_{SP}) \]

where the two sets are divided in the following elements:

- programs \( \subseteq \{\text{sourceCode, objectCode, macro, plugIn}\} \)
- documentations = \{userManual, referenceManual, developmentDocumentation\}

and \( R_{SP} \) describes the set of the relations over the \( SP \) elements.
Software product formalization

The given subsets could be described as follows:

developmentDocumentation = \{documentationElements\} \quad (4)
= \{productRequirements, productSpecification, productDesign, implementationDescription\}
documentationElements \subseteq \{model, chart, architecture, diagram, estimation, review, audit, verificationScript, testCase, testScript, pseudoCode, extensionDescription, qualityReport\} \quad (5)
productRequirements = systemRequirement \subseteq \{functionalRequirements, qualityRequirements, platformRequirements, processRequirements\} \quad (6)

We define software products formally using this approach.

Maintenance process ontology

Operational Support in SM

Preventive in SM

Adaptive in SM

Perfective in SM

Evolutions in SM

Changed Requirement

New Requirement

Changed Implementation

Modification Activity

Correction

Enhancement

Investigation Activity
The definition of an investigation can be expressed as the requirements identified on a specific Support Request with the use of the operational support workflow resulting in a software that is investigated for this specific request:

\[ a^{(\text{correction})} \in A_{SM} : SP \times SRI_{\text{supportRequirements}} \times \text{operationalSupportWorkflow} \rightarrow SP^{(\text{investigated})} \]  

We define software activities formally using this approach.
Model process formalization

\[ SM = (A_{SM}, R_{SM}) = (\{maintenanceActivity, maintenanceResources\} \cup SP) \]

Therefore, we can define the software maintenance resources \( SR \) as follows:

\[ SR = (M_{SR}, R_{SR}) = (\{peoplewareResources, softwareResources, platformResources\}, R_{SR}) \]

peoplewareResources = \( \{\text{maintainer, analyst, developer, customer, user}\} \)

We define software maintenance resources formally using this approach.
Architecture alignment to CMMi

SM** - Resulting KPA's

<table>
<thead>
<tr>
<th>SM** Process Domains</th>
<th>Key Process Areas of Software Maintenance</th>
</tr>
</thead>
</table>
| Process Management   | 1- Maintenance Process Focus  
|                      | 2- Maintenance Process/Service definition  
|                      | 3- Maintenance Training  
|                      | 4- Maintenance Process Performance  
|                      | 5- Maintenance Innovation and deployment |
| Maintenance Request Management | 1- Request & Event Management  
|                                    | 2- Maintenance Planning  
|                                    | 3- Monitoring & Control of maintenance requests  
|                                    | 4- SLA & Supplier Management |
| Evolution Engineering   | 1- Predelivery and Transition  
|                        | 2- Operational Support Services  
|                        | 3- Software Evolution & Correction Services  
|                        | 4- Verification and Validation |
| Support to Evolution Engineering | 1- Configuration and Version Management  
|                                    | 2- Process and Product Quality Assurance  
|                                    | 3- Measurement, Decision Analysis  
|                                    | 4- Problem Resolution and Causal Analysis  
|                                    | 5- Software Rejuvenation, Migration and Retirement |
SMmm - Overview

- Model in numbers
  - 4 Process Domains
  - 18 KPA's
  - 74 Roadmaps
  - 443 Practices with supporting text and references

Part 4 - Case studies
4 - \textit{SM}^{mm} \text{Validation}

- \textit{SM}^{mm} v1.0 model (2002)
- Update of the \textit{SM}^{mm} v1.5 model (2005)
- Malcolm-Baldrige
- ITIL
- CMMi
- IT Service CMM
- Swebok

Last update with conclusions of this thesis

Trial in Canadian telcomm.

Case studies in Middle east telcomm.

4 - \textit{SM}^{mm} \text{Experimentation}

- Consolidated Maturity Profile for Level 1 and 2 maturity

\begin{itemize}
  \item Level 1: Managed
  \item Level 2: Performed
\end{itemize}
4 - SM\textsuperscript{mm} Validation

- SM\textsuperscript{mm} (April)
  - 4 model V&V steps since 1999;
  - Experiment with 3 Case studies (Telecommunications Cie);
  - Four 3 days assessments with assessment plan;
  - Use the model, 1 context form & 1 observation/problem report form
  - Output:
    - Consolidated Maturity Profile for Level 1 and 2 maturity;
    - Identification of 3 company process improvement projects;
    - Model improvement list.

- 3 Improvement Projects- 2000-2 (Published Results)
  1) Maintenance Process Measurement;
  2) Definition of a Maintenance Service Level Agreement (SLA);
  3) Software Product Measurement (source code measures).

4 - SM\textsuperscript{mm} Evaluations

- Model strength and weaknesses

<table>
<thead>
<tr>
<th>Rank</th>
<th>Comments</th>
<th>+/-</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>There are important areas that the model does not address (ITIL, Cobit and Malcolm Baldrige)</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>The model accurately portrayed the process state of the organization</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>The model was useful for identifying what has to be improved</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>Model has too many practices to be used in a small maintenance organization</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Some KPAs have too many practices in order to achieve the level</td>
<td>-</td>
</tr>
</tbody>
</table>
Conclusions of research questions

1- Software maintenance is a specific domain of software engineering and its specific processes/activities are published in SWEBOK.

2- We are awaiting the ISO/IEC 14764 next NWI proposal on our process recommendations.

3- No maturity model proposal covers the entire set of software maintenance unique activities as proposed by SWEBOK.

4- We have presented the proposed architecture of a capability maturity model that could address the entire set of software maintenance unique activities.

5- We have shown how the model was used in practice to support the improvement of software maintenance.

Future Work

- Full model release in a French Book - 2005
- Discussions on the English/German versions
- Evaluation tool built by Msc student
- Knowledge Based to support training
- Are posted on my WEB site at: 
  Http://profs.logti.etsmtl.ca/aapril/English/Autres/index.html
**SMAssess tool**

1. Choose the language to answer.
2. Enter the details of the assessment.
3. Select the questions.

**Thank You**
Related Publication

1- Software Measurement Conference (FESMA-AEMES), Madrid, Spain, October 18-20, 2000.
3- Internal Technical Report 02-001, Montréal, ÉTS Software Engineering Laboratory, 10-11-2002.
4- Internal Technical Report 02-002, Montreal, ÉTS Software Engineering Laboratory, 30-12-2002.
5- 13th International Workshop on Software Measurement - IWSM Montréal (Canada), 2003.
8- Preprint, Faculty of Informatic, University of Magdeburg, November 2004.

10- 8th European Conference on Software Maintenance and Reengineering (CSMR2004), Tampere (Finland), Mar. 24-26, 2004
12- International Conference on Software Measurement (IWSM/MetriKon 2004), Königs Wusterhausen, Germany, 2004;