University of Sofia, Faculty of Mathematics and Informatics

MÄLARDALEN UNIVERSITY
SWEDEN

Mälardalen University, School of Innovation, Design and Engineering

MASTER THESIS IN SOFTWARE ENGINEERING

A COMPARISON OF CBSE AND MDD FROM THE PROCOM PERSPECTIVE

Nikolay Grozev

Supervisor: Juraj Feljan, Mälardalen University
Consultant: Sylvia Ilieva, University of Sofia

OVERVIEW

- Introduction
- General comparison - background
- General comparison of CBSE and MDD
- Overview of ProCom
- Comparison of CBSE and MDD with respect to ProCom
- Conclusion
- Q & A

INTRODUCTION

INTRODUCTION

- Software development complexity is proverbial – especially in the embedded domain. Ways to tackle it:
  - Component-based software engineering (CBSE)
  - Model Driven Development (MDD)
- ProCom – component model for the embedded domain. Uses both CBSE and MDD
- Thesis purposes
  - Systematically compare CBSE and MDD in general
  - Enrich the general comparison with a case study comparing CBSE and MDD with respect to ProCom.
  - Analyze the outcomes of the comparisons.

GENERAL COMPARISON - BACKGROUND

- ✓ CBSE BASICS
- ✓ MDD BASICS
- ✓ COMPARISON METHOD

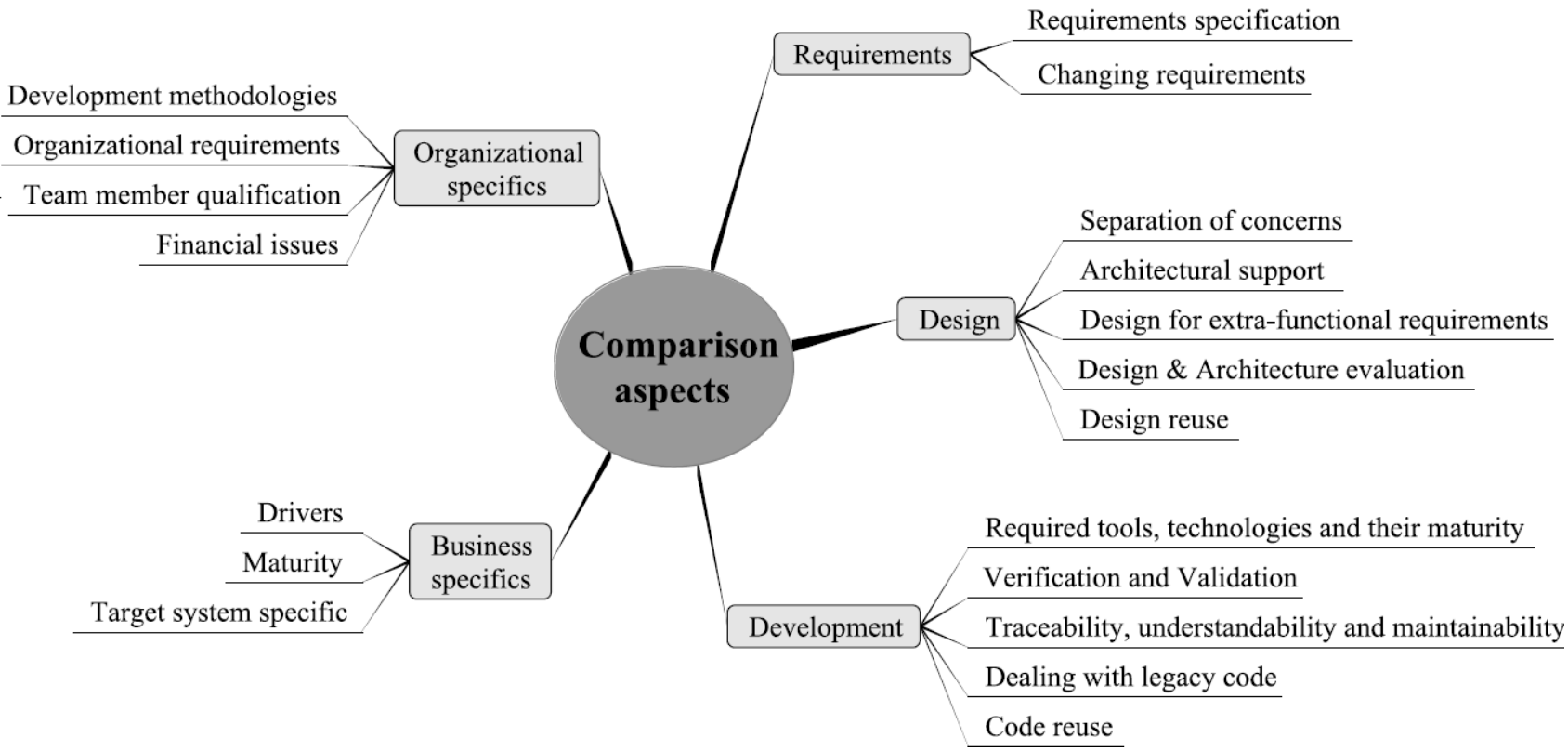
CBSE BASICS

- CBSE - building software from pre-existing components
- Component - many definitions
  - "*A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party*" – Szyperski
- Interface - a specification of a component's access point
- Component model
  - A set of standards and conventions for a component.
  - Defines what a component is and how it interacts with other components
- Component framework – a component model implementation.

MDD BASICS

- Rahmani: "*[A model is] a description or specification of a system and its environment for some certain purposes*“
- Model formalisms
- Model transformation - an operation which converts a model of a system described in formalism F_1 to one described in formalism F_2.
- MDD refinements – MDA, AMDD

METHOD OF COMPARISON



- Two-level hierarchy of “comparison aspects”

GENERAL COMPARISON

- BUSINESS SPECIFICS
- REQUIREMENTS
- DESIGN
- DEVELOPMENT
- ORGANIZATIONAL SPECIFICS

BUSINESS SPECIFICS

	CBSE	MDD
Drivers	Component reuse	Higher level of abstraction than code. Complexity management.
Maturity	Relatively mature. Some theoretical achievements still have not made it into practice. Many practitioners still misunderstand basic concepts.	Controversy about real-life usefulness of modeling. Not yet proven in practice, though some types of modeling have been a basis for success stories.
Target system specifics	Suitable for all kinds of systems, except for ones with high demands for extra-functional requirements. Especially useful for product lines.	Modeling can be applied in all kinds of systems. Most useful in large and complex ones.

REQUIREMENTS

	CBSE	MDD
Requirement specification	Most current technologies do not allow requirements specification, though CBSE relies on well specified requirements.	Some models are very useful for requirement specification - UML use cases, BPMN etc.
Changing requirements	Changing requirements pose great risks - hard traceability, cyclical requirement-component dependency (CRCD)	Allows adequate reaction to such a change, since the software is more traceable. If transformations are automatic changes can be applied very quickly.

DESIGN

	CBSE	MDD
Separation of concerns (SoC)	SoC is implemented through components and interfaces.	Does not define explicitly how SoC is achieved.
Architectural support	Technology-specific formalisms. Lack of popular standards. Many technologies do not facilitate modeling at all.	Suitable for architecture and design specifications -ArchiMate, some UML models, ADLs
Design for Extra-functional requirements	“Credentials” are still not widely used.	UML extensions especially designed for the purpose.
Design & Architecture evaluation	Does not help in architecture/design evaluation. Standard evaluation approaches are used.	Standard evaluation approaches can be used. Some models help when discussing design/architecture properties.
Design reuse	Some component system models can be reused. This is hindered by the lack of popular and standard component modeling formalisms.	Models can be reused in case the "donor" and "recipient" systems use the same formalisms. Using modeling standards like UML is a best practice.

DEVELOPMENT

	CBSE	MDD
Required tools, technologies and their maturity	The only tool needed - component framework. Currently CBSE tools are considerably mature, though some theoretical achievements have not been widely incorporated yet. Vendor lock-in can be avoided by using implementations of public component model specifications.	Sophisticated tools are required. If a popular modeling formalism is used (e.g. UML) there are mature tools available. Vendor lock-in can be avoided by using tools that can export models to a popular exchange formats - e.g. XMI.
Verification and Validation (V&V)	Components' quality is crucial and they should be heavily verified and in many cases certified. Component based systems are easier for verification since a lot of the work has been done at the component level. Their validation is similar to that of standard systems.	Models allow for early analysis. Model based testing can make verification easier. Having models expressed in a mathematical formalism allows automatic verification.

DEVELOPMENT

	CBSE	MDD
Traceability, understandability and maintainability	Depends on the quality of the documentation and the architecture. Maintenance depends on the quality of the glue code and the maintenance support of the components.	High traceability and understandability. Maintainability is usually also high, especially if the transformations are well automated. Problematic model versioning, comparison and merging.
Dealing with legacy code	Legacy code is encapsulated into designated components.	Legacy code must be "brought" to the level of the models, so that the interaction with it can be modeled. Reverse engineering tools are usually used for the purpose.
Code reuse	Reuse is the paramount idea. Custom components still need to be developed, but they are typically reused in-house later. Reuse benefits can be diminished by developing too much glue code.	Not focused on code reuse. Code reuse may be a side effect of model reuse.

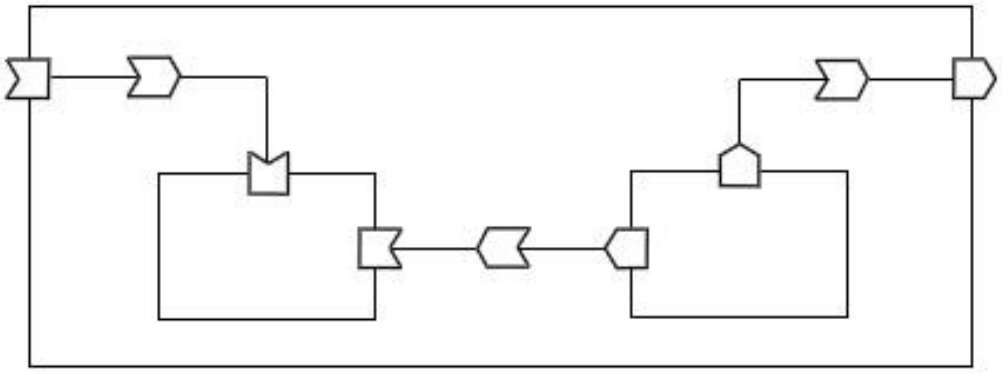
ORGANIZATIONAL SPECIFICS

	CBSE	MDD
Organizational specifics	Lack of real-life proven development methodologies. Existing methodologies are customized to facilitate CBSE specific activities, roles etc. Staff should be trained in the basic theoretical concepts of CBSE. Use a pilot project for gathering hands-on experience.	Can be used with almost all existing development methodologies, with minor adaptations. Steep individual learning curve due to complex formalisms and tools. Enterprise modeling tools contribute to the price of adoption as well.

OVERVIEW OF PROCOM

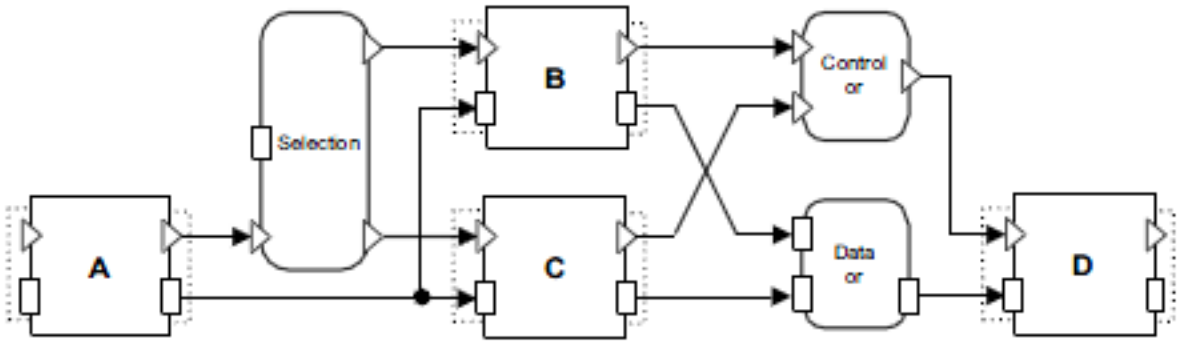
OVERVIEW OF PROCOM

- ProCom - component model for embedded systems used throughout SDLC
- Design at two level of granularity – ProSys and ProSave
- ProSys - coarse grained modeling
  - Subsystems – concurrent active components
  - Message passing - typed input and output message ports
  - Hierarchical components



OVERVIEW OF PROCOM

- ProSave – fine grained modeling
  - Components encapsulate relatively small and rather low-level, non-distributed functionality
  - Components are hierarchical, passive and present at design time only. Primitive components are C functions
  - Ports and connectors
- Deployment modeling



COMPARISON WITH RESPECT TO PROCOM

- BUSINESS SPECIFICS
- REQUIREMENTS
- DESIGN
- DEVELOPMENT
- ORGANIZATIONAL SPECIFICS

BUSINESS SPECIFICS

	Summary
Drivers	Drivers for using ProCom are the same as those for CBSE & MDD ➢**CBSE perspective:** Component reuse ➢**MDD perspective:** Work at a higher level of abstraction than code. Complexity management. Early evaluation.
Maturity	Relatively mature, though still under development. Still not tested in industrial environment. ➢**CBSE perspective:** Relatively mature from theoretical perspective - incorporates components throughout the whole lifecycle, allows specification of extra-functional properties ➢**MDD perspective:** Incorporates the basic ideas of MDD - defines modeling formalisms for representing a system from different viewpoints, a system is developed as a series of models related with transformations.
Target system specifics	Embedded systems.

REQUIREMENTS

	Summary
Requirement specification	No a general purpose way for specifying requirements. ➢**CBSE perspective:** The attribute framework for specifying extra-functional component properties can be used to model extra-functional requirements. ➢**MDD perspective:** N/A
Changing requirements	Better than many existing component technologies, because of the amount of modeling extending the core CBSE ideas. ➢**CBSE perspective:** Model driven techniques mitigate many of the inherent for CBSE problems. Cyclical requirement-component dependency (CRCD) is still a problem. ➢**MDD perspective:** ProCom systems are more traceable due to the employed modeling techniques. This allows for an adequate reaction to a change in the requirements.

DESIGN

	Summary
Separation of concerns (SoC)	➢**CBSE perspective:** SoC is implemented through components and interfaces. ➢**MDD perspective:** Components in ProCom are design-time entities and thus they are the SoC elements in terms of MDD as well.
Architectural support	ProSys models and the deployment models can be considered as two architectural views. ➢**CBSE perspective**: introduces its own ways for architecture specification formalisms - ProSys and the deployment modeling formalism. ➢**MDD perspective**: ProSys can be seen as a modeling formalism for specifying architecture. ProSys and the deployment models can be seen as two related architectural models of a system.
Design for Extra-functional requirements	➢**CBSE perspective**: uses a custom attribute framework for specifying extra-functional requirements based on the "credentials" approach. ➢**MDD perspective**: N/A

DESIGN

	Summary
Design & Architecture evaluation	➢**CBSE perspective**: N/A ➢**MDD perspective**: ProCom takes a model driven approach to design and architecture evaluation by providing interrelated models/views suitable for different analyses.
Design reuse	➢**CBSE perspective**: N/A ➢**MDD perspective**: Functional design in ProCom can be reused, since it is expressed through ProSys & ProSave components which are design time entities (models).

DEVELOPMENT

	Summary
Required tools, technologies and their maturity	➢**CBSE perspective**: N/A ➢**MDD perspective**: Due to the support for visual modeling formalisms ProCom implies the use of sophisticated IDEs like ProCom
Verification and Validation (V&V)	The research about V&V of ProCom components and systems is in an early conceptual phase. ➢**CBSE perspective**: N/A ➢**MDD perspective**: model based testing, model analyses.

DEVELOPMENT

	Summary
Traceability, understandability and maintainability	➢**CBSE perspective**: N/A ➢**MDD perspective**: High traceability and maintainability because of the many interrelated models. Problematic versioning and merging of the visual models.
Dealing with legacy code	➢**CBSE perspective**: Legacy code is divided into components of different granularity and then reused. ➢**MDD perspective**: Components in ProCom are design-time entities and thus when the legacy code is divided into components these can be used for modeling as well.
Code reuse	➢**CBSE perspective**: Components are the main artifacts of reuse. Because of the specifics of the embedded domain, significant reuse is most likely to occur within a suite of similar projects. ➢**MDD perspective**: In ProCom model reuse and component reuse is more or less the same thing. Reusing deployment models leads to reuse of executables.

ORGANIZATIONAL SPECIFICS

	Summary
Organizational specifics	➢**CBSE perspective**: implies customization of existing development methodologies. New development roles in the process. Suitable for product lines. A pilot low-priority project may be used for experimenting and training purposes. ➢**MDD perspective**: Unlike most MDD technologies, ProCom implies significant changes in the used development methodologies. New modeling formalisms and thus employees' qualification is of even greater concern than in other MDD technologies. Steep individual learning curve and great need for training

CONCLUSION

CONCLUSION

- General comparison
  - CBSE is superior to MDD when it comes to code reuse, legacy code and tool support
  - Component technologies imply less steep learning curve
  - CBSE implies changes of existing development methodologies
  - MDD results in more traceable, maintainable and resilient to changes systems
  - MDD technologies are superior in terms of defining software architecture or design.

- By extending the core CBSE concepts with adequate modeling capabilities these shortcomings of the current component based approaches can be mitigated.

- ProCom comparison
  - Augments the core concepts of CBSE with several model driven approaches
  - Shortcomings of CBSE concerning traceability, maintainability, analyzability and specifying architecture and design are mitigated in ProCom.
  - Steep learning curve due to the new modeling formalisms.
  - Need for specific development methodologies as in CBSE.

QUESTIONS

Thank you!