Perspectives on applying Normalized Systems Theory to Conceptual Modeling

Prof. Dr. Jan Verelst

Universiteit Antwerpen

Overview

Introduction

What is Normalized Systems Theory ?

Perspective 1: Modularity/Evolvability of CM

Perspective 2: Leveraging Domain Knowledge & CM Languages

Conclusions

Introduction

Conceptual Modeling in the context of enterprises, ie. Enterprise/Organizational Modeling, plays a **crucial role** in the **development of information systems**.

- **Correct specs** is a well-documented success factor
- Lots of **budgets** spent on analysis/specs
- As the **size and complexity** of systems increase (IoT), the complexity reduction offered by CM is more and more valuable
- **Digital transformation** requires a common language between business and IT is of increasing importance. Society and enterprises depend on more and more software (AI, BI, …), so quality of software becomes more important and CM plays an ever more important role.

Introduction

Although the **CM community's focus** is often on issues like:

- Accuracy of representation
- Reasoning
- Development of CM (languages)
- …

This **keynote** focuses on enriching current Normalized Systems research with:

- **Modularity/Evolvability of CM**
- **Leveraging Domain Knowledge & CM Languages**

In the **context** of:

- Theory and practice, not research-only
- Realistic, large-scale models, not small-scale examples
- (Transactional) Information Systems, not specifically new evolutions such as Blockchain , IoT, …

Overview

Introduction

What is Normalized Systems Theory ?

Perspective 1: Modularity/Evolvability of CM

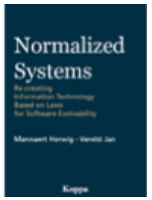
Perspective 2: Leveraging Domain Knowledge & CM Languages

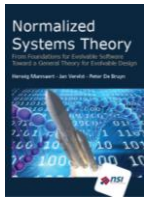
Conclusions

About Normalized Systems Theory

A theoretical framework investigating **Modular Structures** under **Change**

- Based on concepts from **Systems Theory** and **Thermodynamics**
  - ➔ Completely independent of any framework, programming language, package, …
- Initial scope: Modular Structures in **Software Architectures** for Information Systems**,** now **initial steps in CM**
- Publications: book, >50 papers & conference proceedings, (invited) lectures at different universities…
- Education: undergraduate, postgraduate
- Industrial practice: substantial installed base





NS Principles

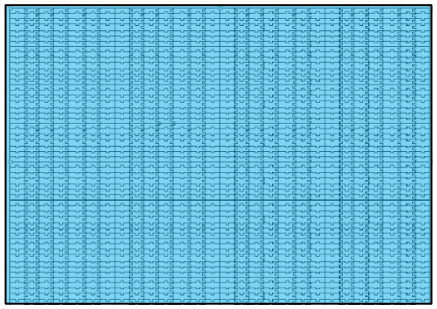
Systems Theoretic Stability

Core concept:
Coupling/Ripple Effects/
Combinatorial Effects

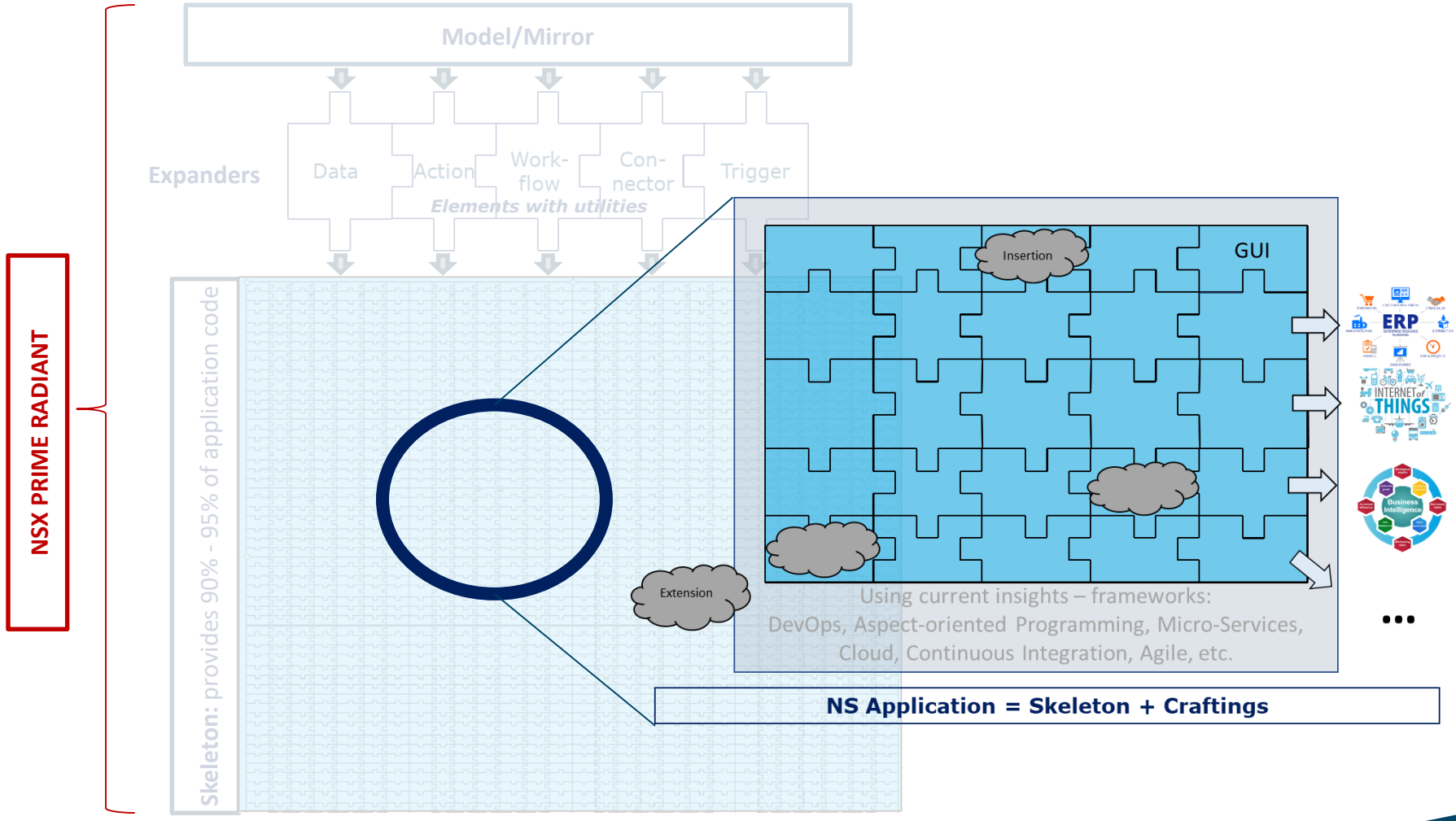
NS Principles

- Separation of concerns
- Data version transparency
- Action version transparency
- Separation of state

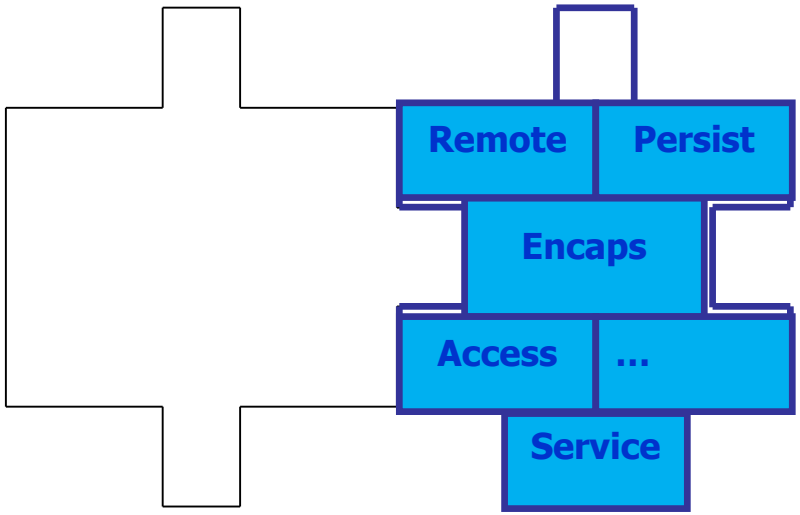
NS=Fine-grained Modularity



NS Elements


NSX PRIME RADIANT
Model/Mirror
Expanders
Data
Action
Work-flow
Con-nector
Trigger
Elements with utilities
Skeleton: provides 90% - 95% of application code
Insertion
GUI
Extension
Using current insights – frameworks:
DevOps, Aspect-oriented Programming, Micro-Services,
Cloud, Continuous Integration, Agile, etc.
ERP
INTERNET of THINGS
Business Intelligence
...
NS Application = Skeleton + Craftings

NS Elements



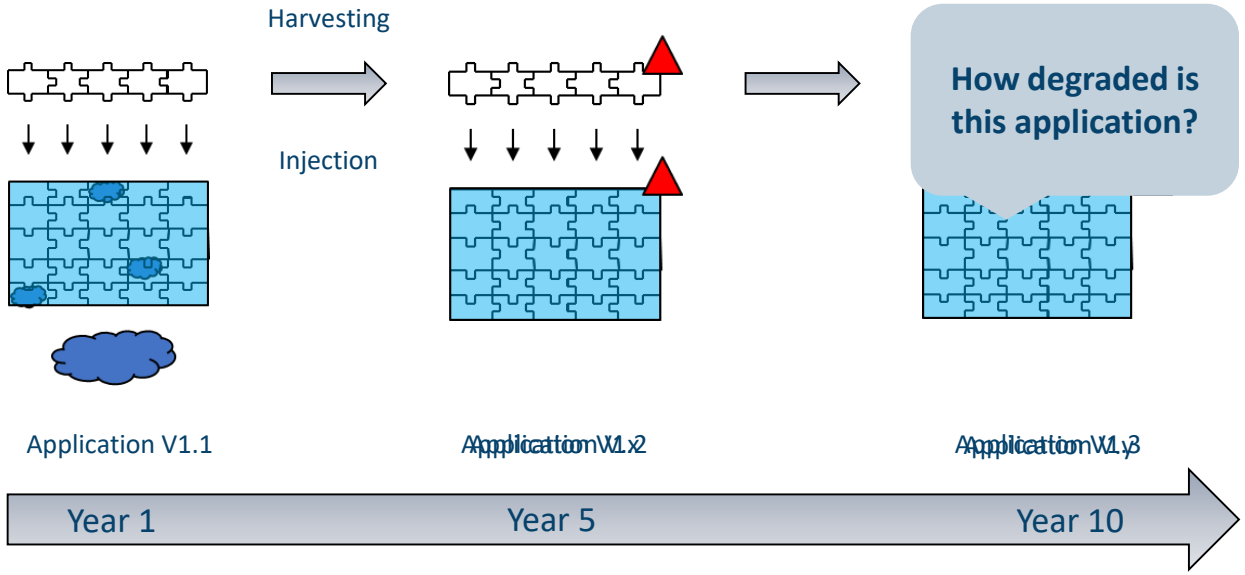
Core Insight:

Rigorously
Separating
IT concerns

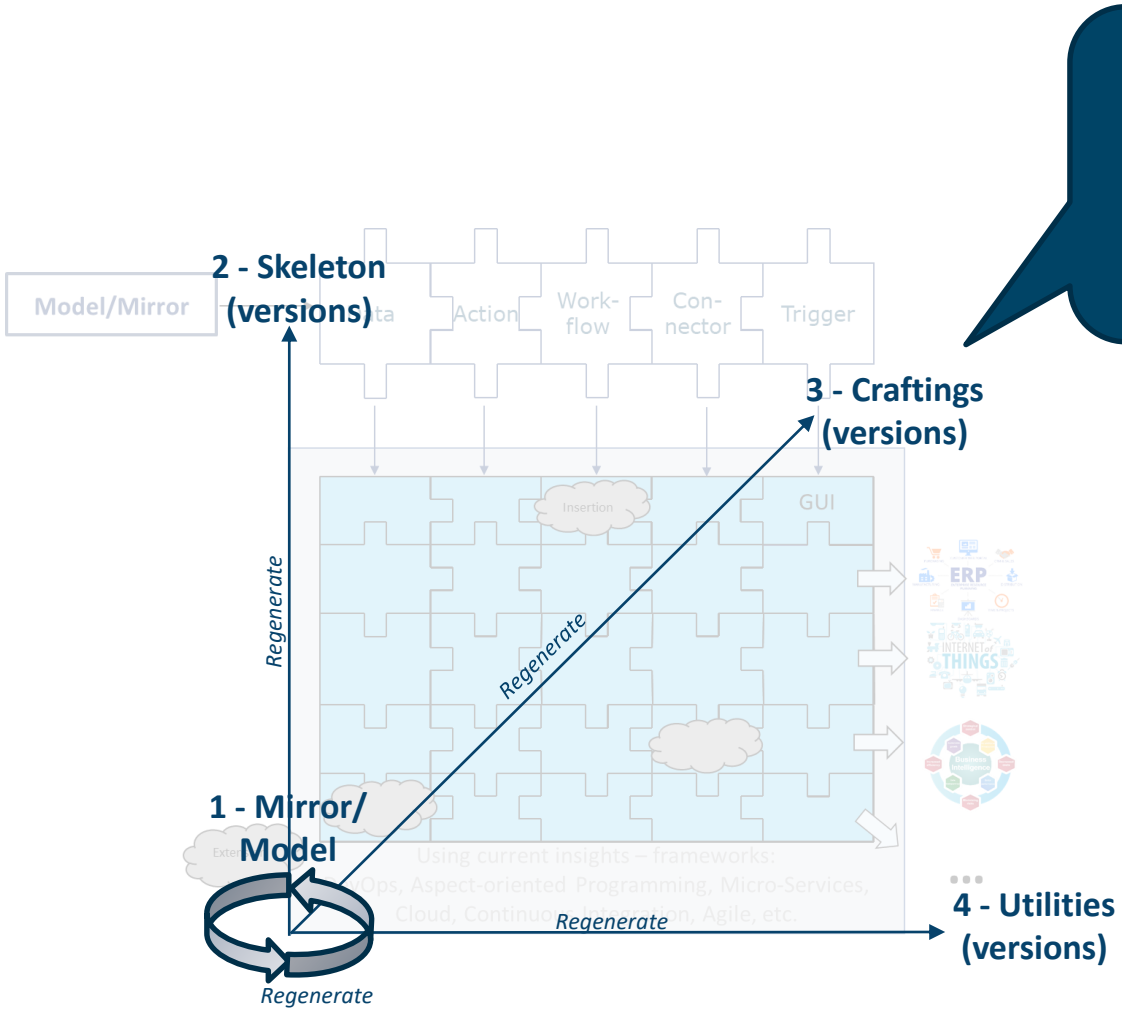
+

Integrating/Aggregating
IT concerns
into NS Elements

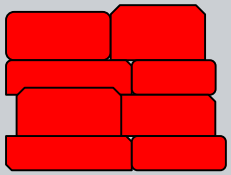
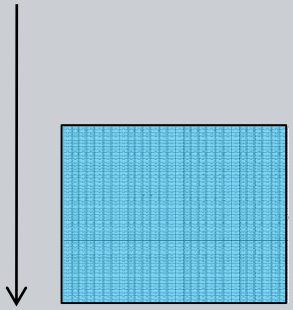
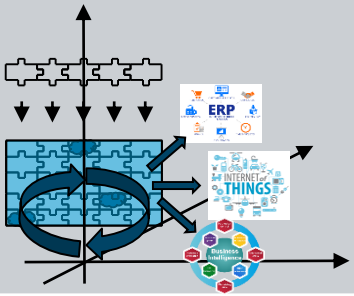
Rejuvenation

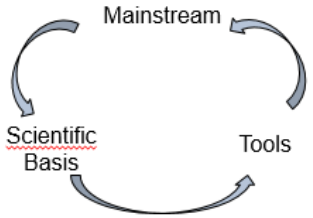


NS: Rejuvenation across 4 Dimensions



NS-Wrap up

Current	Step 1: Principles	Step 2: Elements	The Result
Lehman 	Fine-grained Modularity 	Expansion (Re) Generation 	- **(Eternal) Rejuvenation of IT landscapes** - Systems Theoretic **Stable** & **Isentropic** software - and **Industrial-scale Practice**



Using insights/frameworks/… from:
- DevOps
- Aspect-Oriented Programming
- Micro-Services
- Continuous Integration
- Agile…

The Dream: Doug Mc Ilroy

"expect families of routines to be constructed on *rational principles* so that families fit together as **building blocks.** In short, [the user] should be able safely to regard components as black boxes."

uit: McIlroy, *Mass Produced Software Components*,
1968 NATO Conference on Software Engineering, Garmisch, Germany.

Perspective 1: Modularity/Evolvability in CM

Overview

Introduction

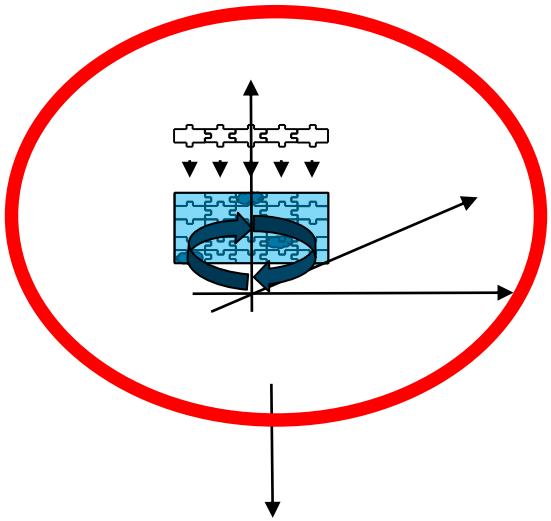
What is Normalized Systems Theory ?

Perspective 1: Modularity/Evolvability of CM

Perspective 2: Leveraging Domain Knowledge & CM Languages

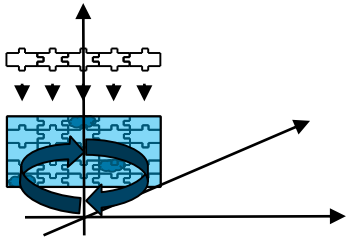
Conclusions

The Potential: Applying NS to CM…



CM-level, business concerns:

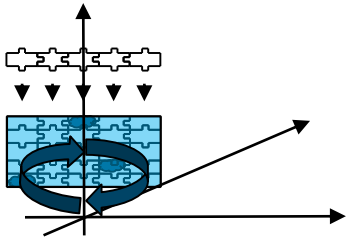
- Financial concerns ?
- Marketing concerns ?
- Legal concerns ?
- Accounting concerns ?…



Software-level, IT concerns:

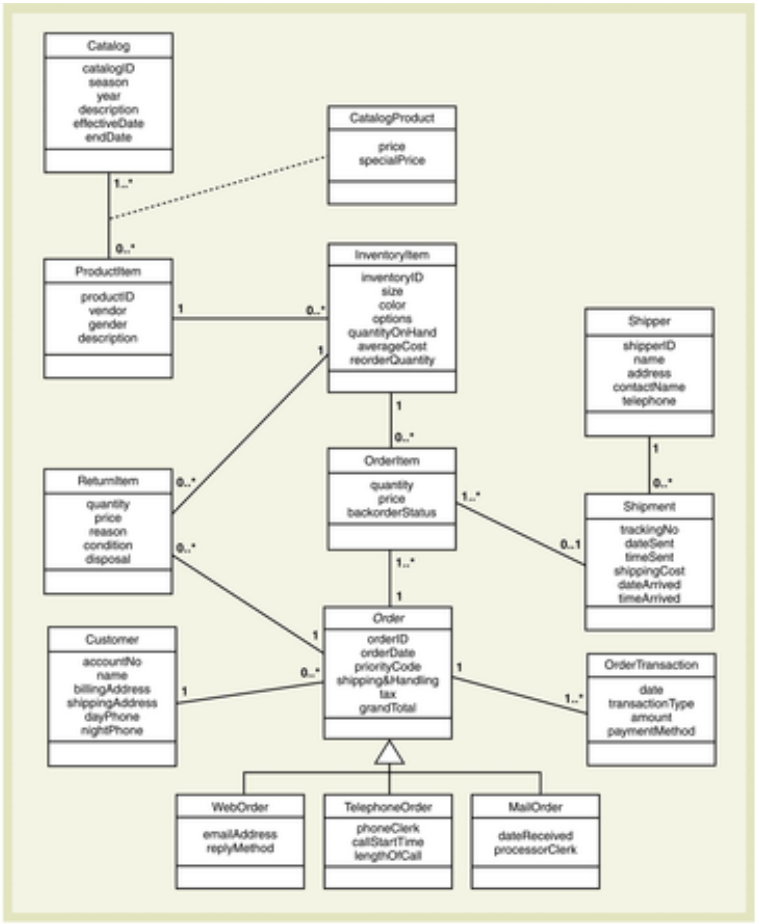
- Performance
- Reliability
- Logging
- Security …

The beginning: little Modular Structure…



In practice:
Low on reuse
Low on integration
Low on automated support

Example of CE in OO data model (UML)



Example of BE in BPMN

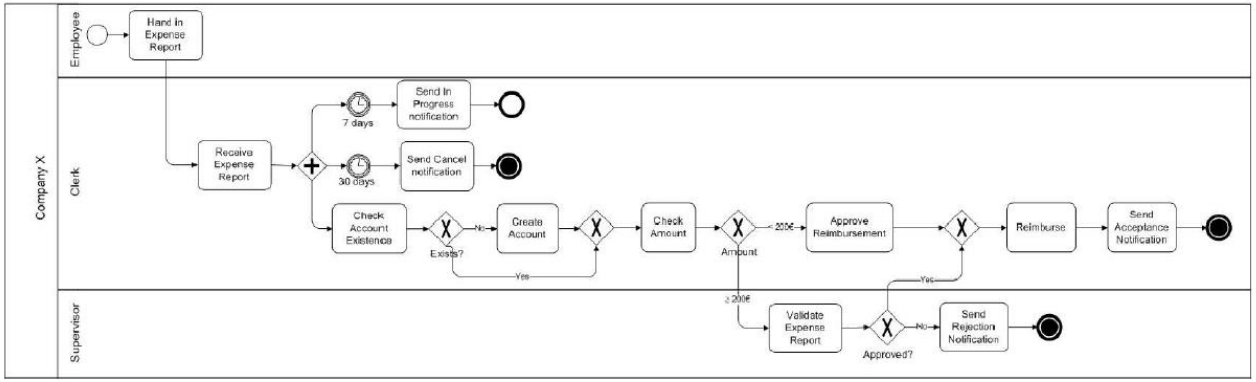


Figure 3.2: BPMN model of an expense reimbursement business process.

Versions & Variations cause CE !

Why modularity and evolvability ?

- Examples of size in CM
  - Business processes: 500 additional pages of BP per year
  - Ontologies: hundreds of concepts
  - Data models: 12 page-definition of ‘marriage’

- Reuse REQUIRES Evolvability !

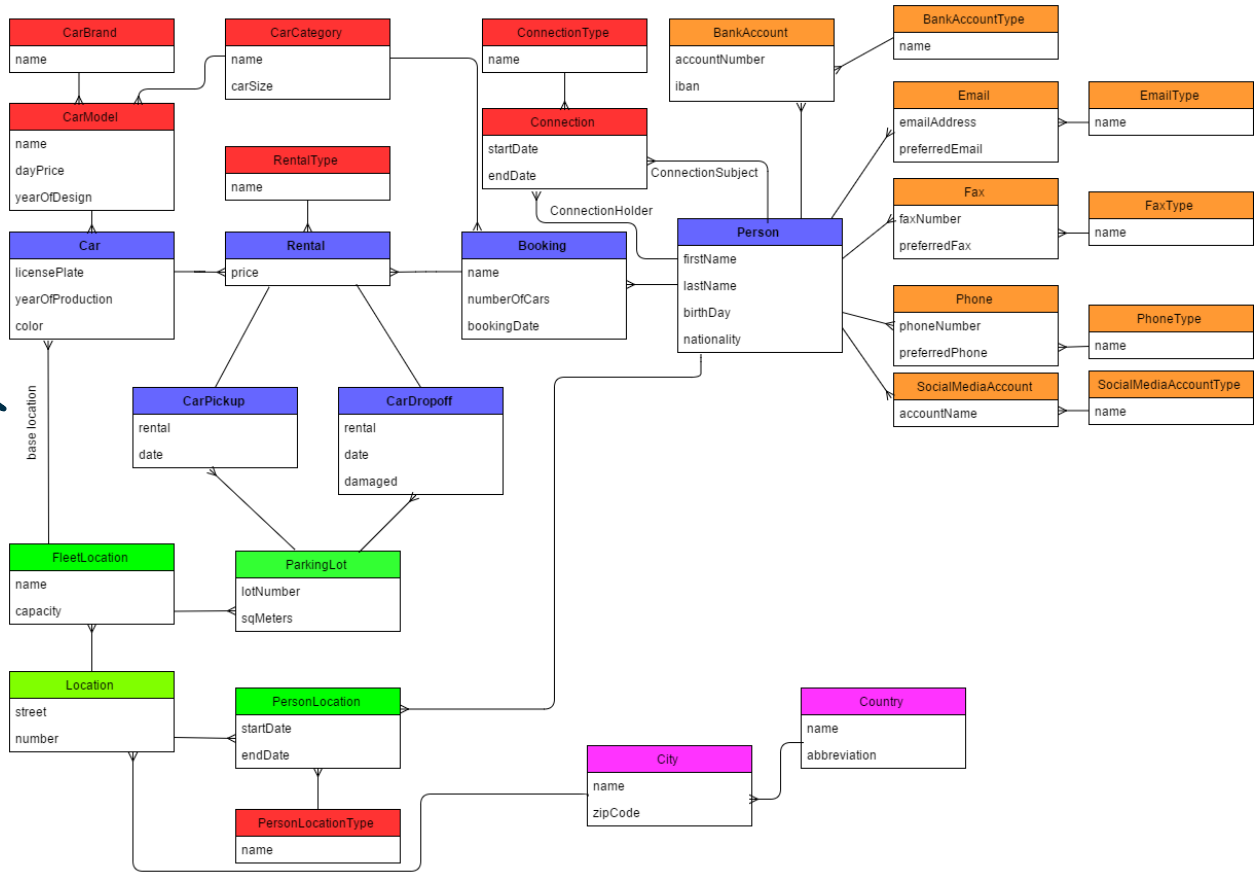
Coupling in CM is too high, and is not systematically adressed in Theory & Practice

In the Business Process Management domain, research on modularity has set off as well. Introducing modularity within business process models mostly resembles the use of sub-processes (Reijers and Mendling, 2008; Reijers et al., 2010). Sub-processes reduce the complexity of the models (Gruhn and Laue, 2009), and as a consequence enhance the model's understandability by hiding irrelevant information (Mendling et al., 2007b; Reijers et al., 2010). Identifying such modular sub-processes might be guided by selecting process modules exhibiting a single input and a single output (Basu and Blanning, 2003). Although multiple authors have already indicated the usefulness of the concept within business process design — Adler (1988), for instance, investigated the decomposition of data flow diagrams — concrete design rules to modularize business processes are still lacking (Reijers and Mendling, 2008). Therefore, modularizing business processes often happens in an ad-hoc way, indicating the need for explicit guidelines to introduce modularity within business processes (Reijers and Mendling, 2008; Reijers et al., 2010).

Some initial examples of applying NS to CM…

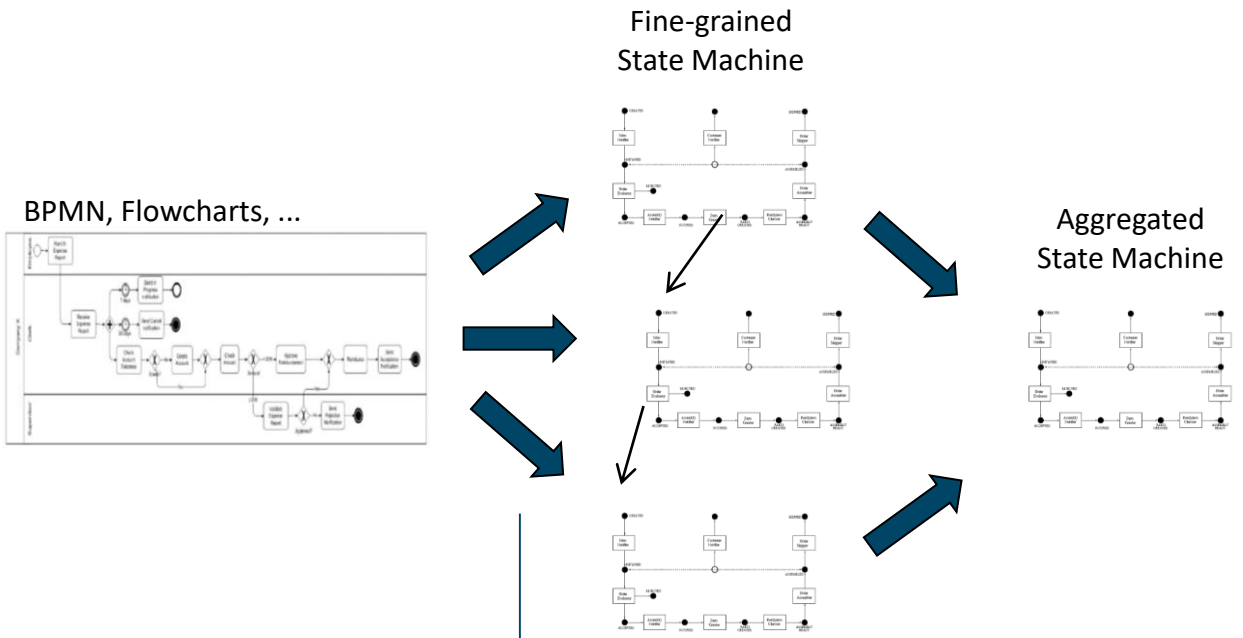
Data Models: Control CE ~ Inheritance

Coupling between dimensions should be ‘close to 0’, not ‘low’ ?

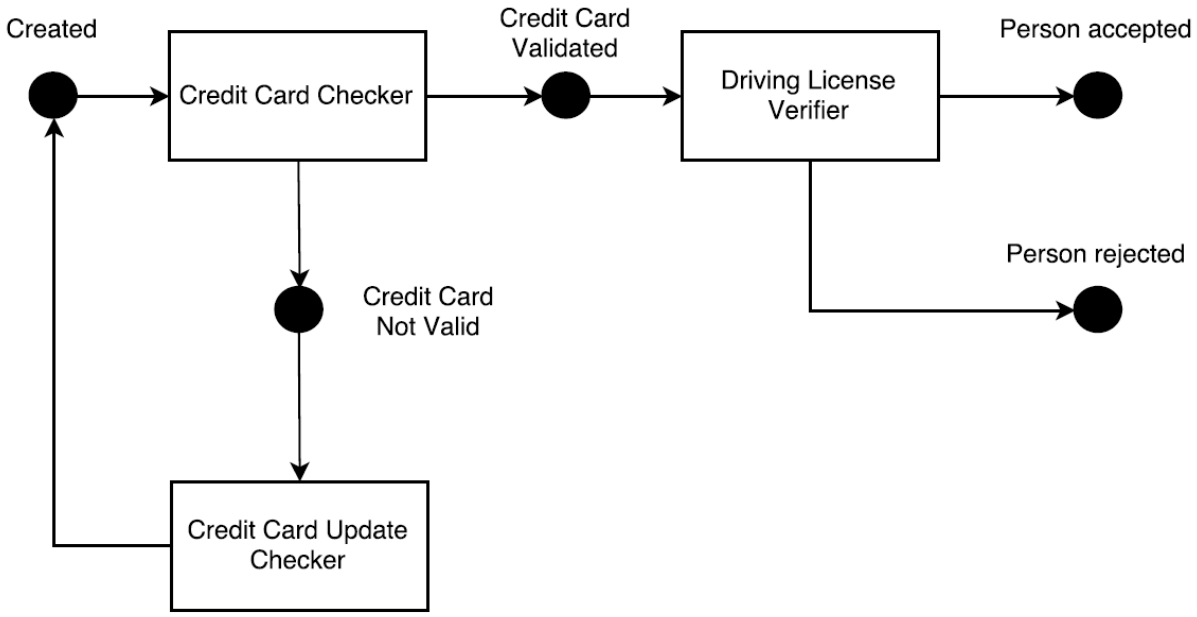


Versions & Variations cause CE !

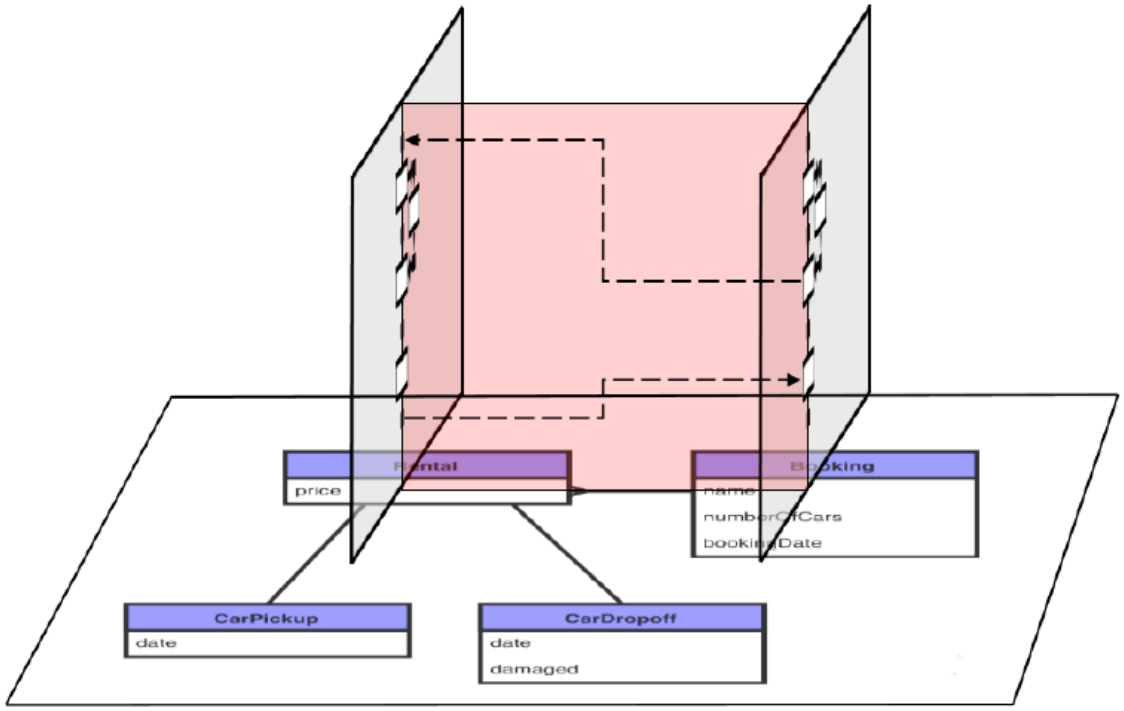
Process models: State machines



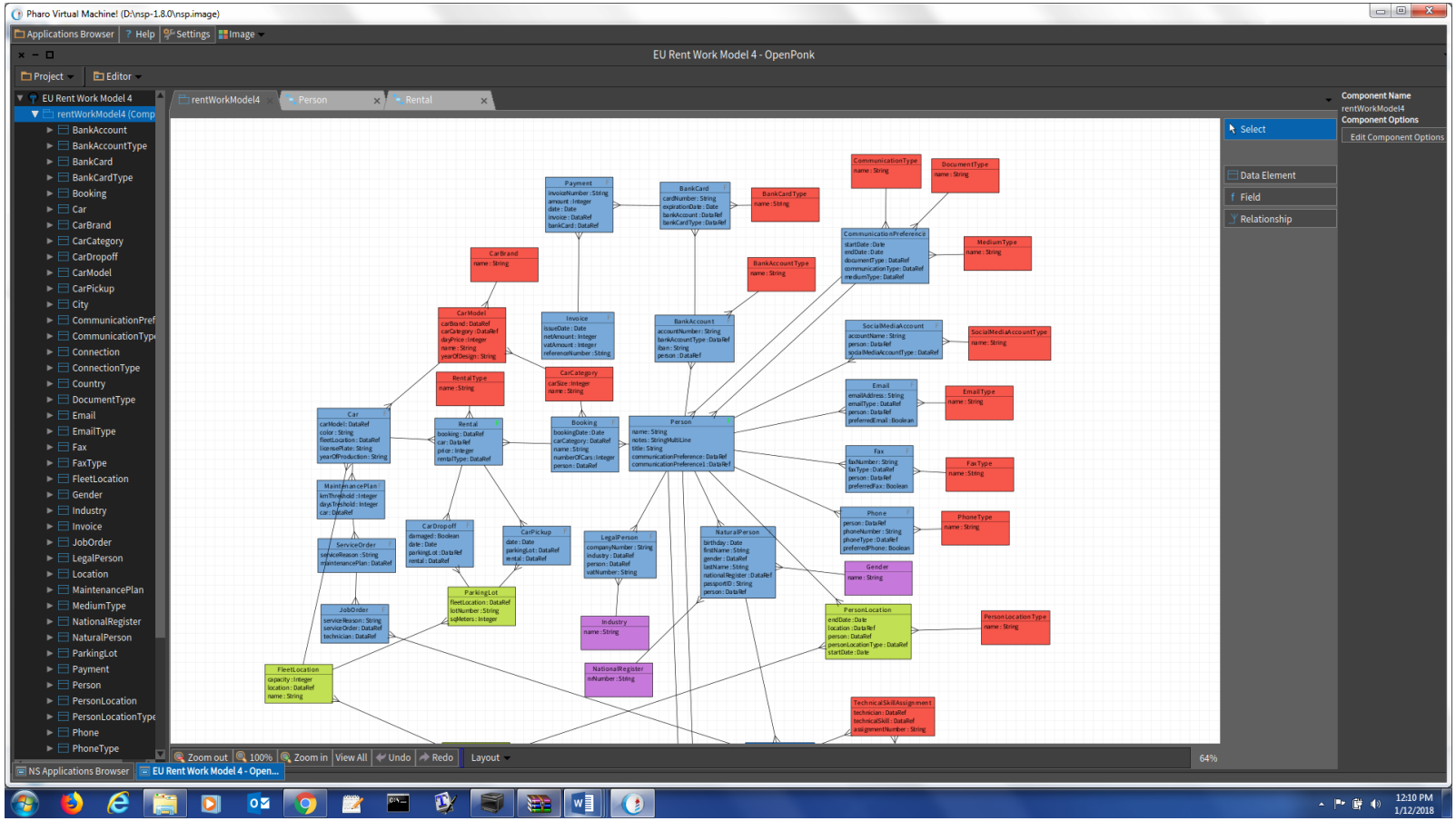
Process models: State machines



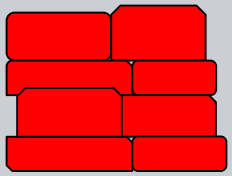
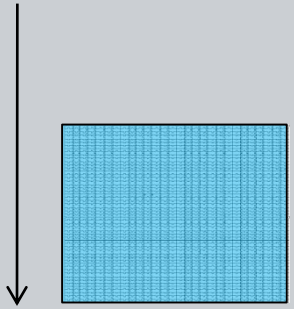
Data models and process models are conceptually interrelated



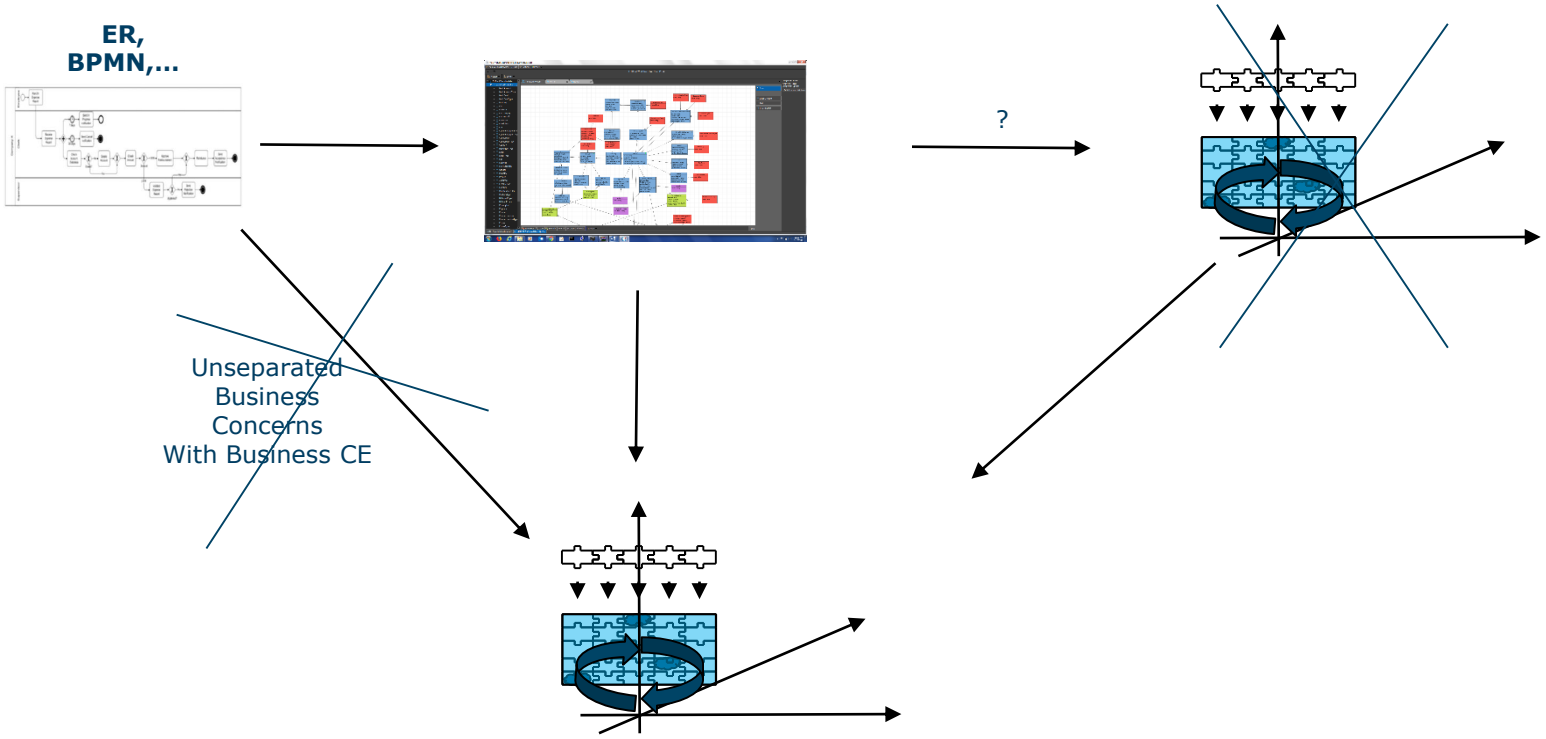
Supporting Systems Analysis: NS Modeler



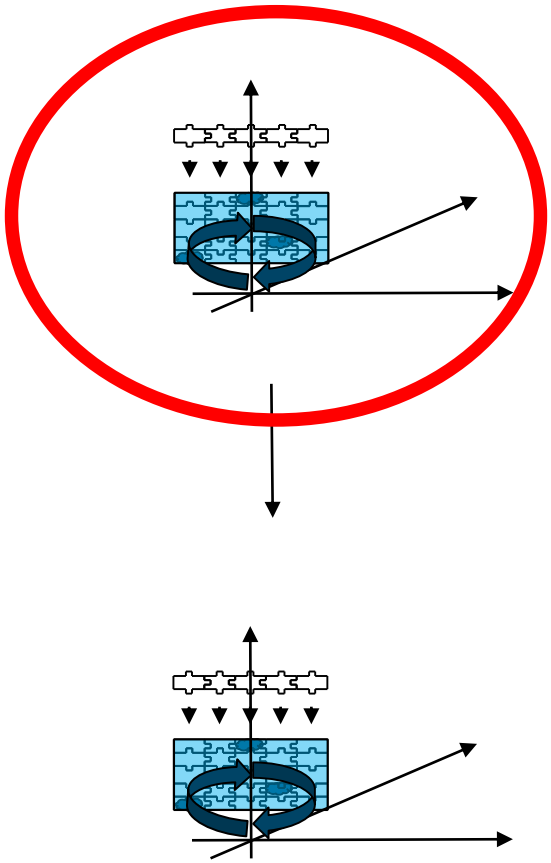
Perspective 1 – Wrap up

Current	Step 1: Principles	Step 2: Elements	The Result
Lehman 	Fine-grained Modularity 		Design guidelines for increasing structure in CM and controlling some CE Coupling remains too high

Current situation: Applying NS to CM…



The Potential: Applying NS to CM…

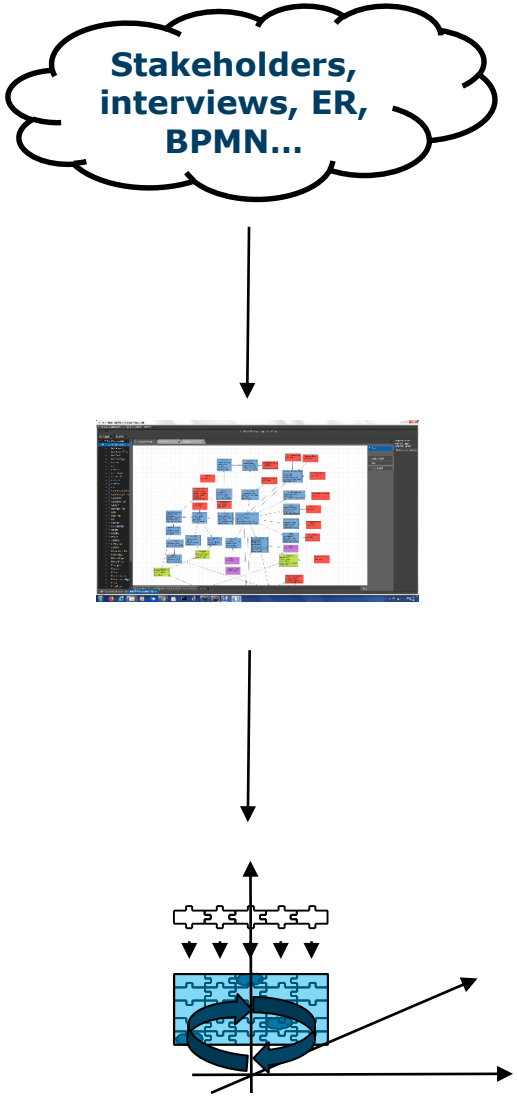


Looking to the future:

- Generation: Expanders f.e. injecting attributes
- Re-generation:
  - Advanced customization mechanisms (harvesting, injecting)
  - Advanced versioning mechanisms
- Advanced integration mechanisms
- Advanced verification mechanisms

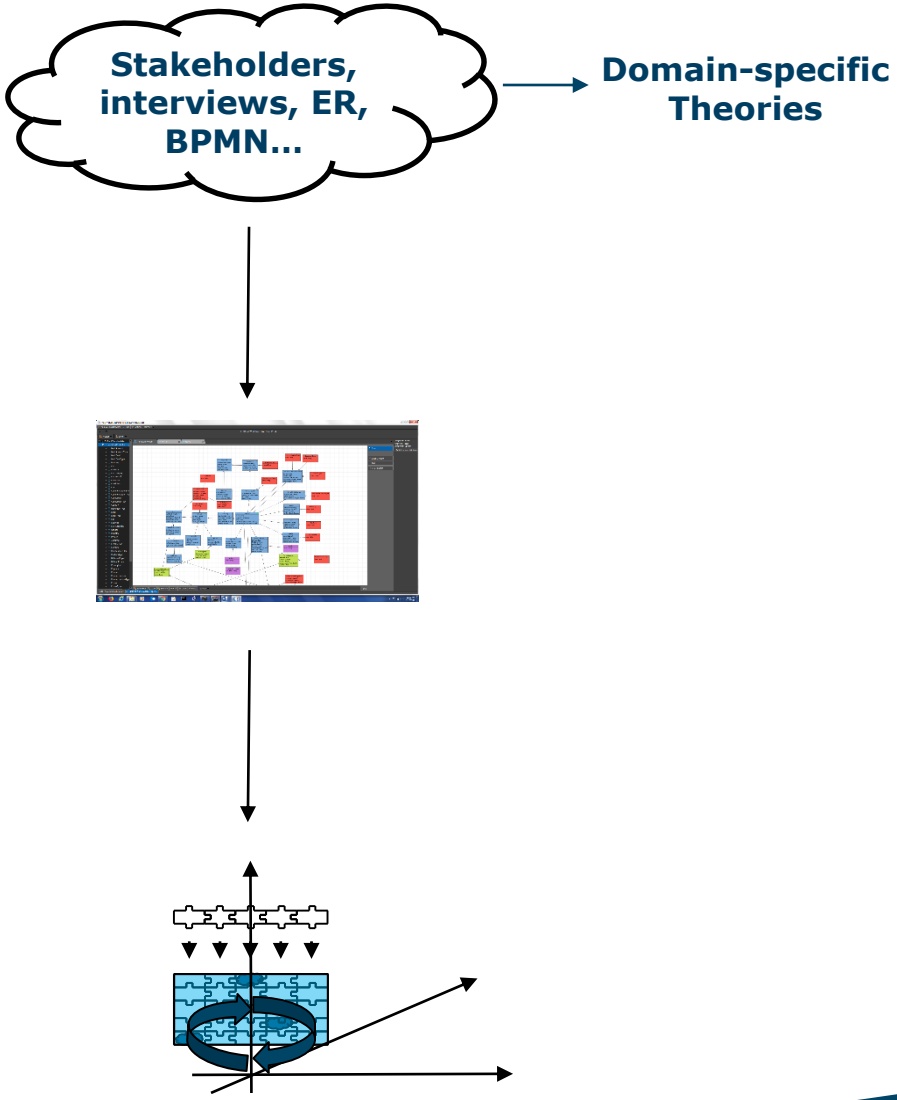
Perspective 2:
Leveraging Domain Knowledge & CM Languages

The situation after perspective 1



In practice:
Low on reuse
Low on integration
Low on automated support

Leveraging domain knowledge



Ontologies

Financials

- FIBO (Financial Industry Business Ontology)

Human Resources

- HR-XML (https://hropenstandards.org/)

Financial reporting

- XBRL (www.xbrl.org)

Business Process Standards

Domain-specific

- Supply Chains: SCOR (www.supply-chain.org)
- Insurance: IAA (c)
- Automotive: STAR (www.starstandard.org)
- Manufacturing: ISA-95 (www.isa-95.com)
- Telecommunications: eTOM(enhanced Telecom Operations Map)
- APQC (www.apqc.org/pcf)

Domain-independent

- OAGI (www.openapplications.org)

The Potential: Applying NS to CM…

CM Languages
&
Domain-specific
Reference models,
Ontologies, Patterns…

Supply Chain Reference Model (CML 2)

UML

OntoUML

I*

Financial Ontology (CML 3)

OWL

Gov't/ Accounting Ontology (CML 4)

ERP

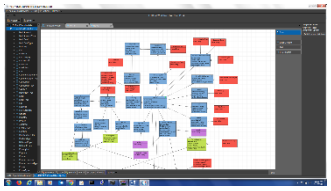
PhD

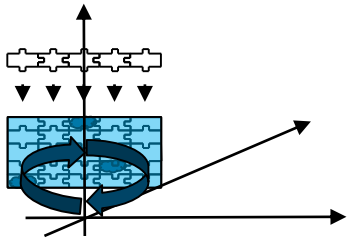
Legal Ontology (CML 1)

FIBO

REA

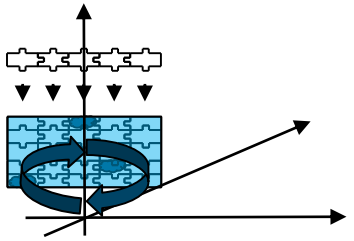
DEMO





Some Interesting Possibilities…

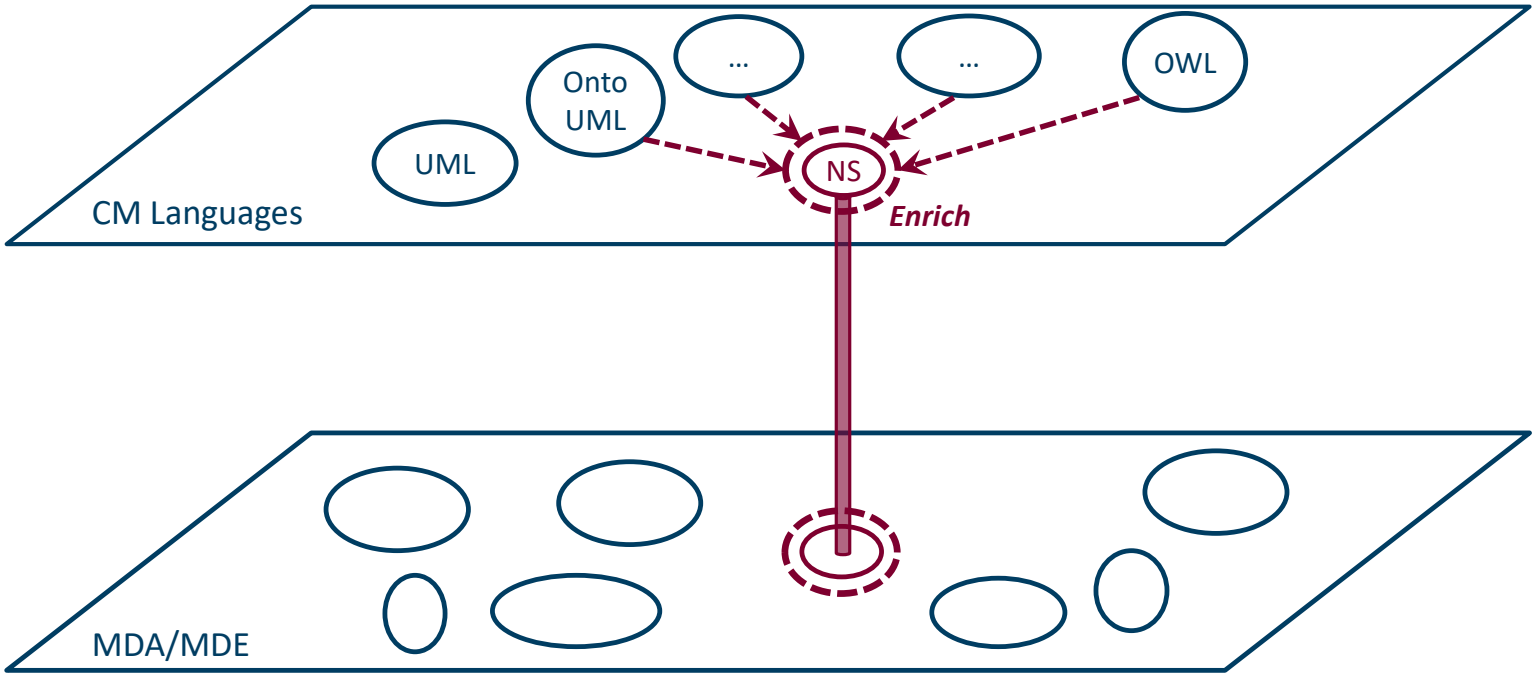
- Implementation of CM is often difficult
  - How to implement in a landscape with software package and COTS ?
  - Over different (versions of) technologies ?



- (Partially) Expand **custom-built CM** in each of these CM Languages on (versions of) several technology stacks in large-scale JEE infrastructures
  - Use multiple CM Languages, as suited to your needs
  - Implementations across CM Languages are similar, reducing effort
  - Integration is facilitated at the software level
    - Between multiple CM-based implementations
    - Between a CM-based implementation and an EA landscape (out of the 'pocket of innovation')

- (Partially) Expand **reference models/ontologies** in each of these CM Languages on (versions of) several technology stacks in large-scale JEE infrastructures

Toward a NS Gateway Ontology



Dealing with other (non-NS) ontologies

Other ontologies, having a more elaborate metamodel than NS, exist

- Some of them allow for code expansion (cf. MDA)
  - fine-grained structure, evolvability, …?
- Some of them do not allow for code expansion
  - how to transform the model in to code?

Research is directed towards the creation of a gateway ontology

- Enables the translation of a non-NS ontology into the NS “gateway ontology”
- Once a model is formulated in this gateway ontology, NS code can be generated
- This requires
  - mappings
  - extension of the NS metamodel/gateway ontology (many features of non-NS ontologies cannot be directly expressed in the NS metamodel/gateway)
- PhD research at University of Antwerp and TU Prague

Overview

Introduction

What is Normalized Systems Theory ?

Perspective 1: Modularity/Evolvability of CM

Perspective 2: Leveraging Domain Knowledge & CM Languages

Conclusions

Conclusions

- NS is about **studying modular structures under change**, based on concepts such as systems theoretic stability.

- At the level of **software architectures**, achieving systems theoretic stability against change is possible, both in theory and in industrial practice.

- At the level of **CM**, initial attempts to provide **more structure and control** of CE have been made, and are operationalized in the **NS Modeler**.
  - Some of these **correlate** with results from other research (ie. eliminating standard inheritance constructs, and the use of state machines).
  - However, the CM field would **benefit** from a **systematic focus on evolution and modularity**. Right now, no elements, expansion or re-generation has been realized, nor are advanced features for integration and reuse currently available.

Conclusions

- **Reusing domain knowledge** seems very promising to improve structure (identification of business concerns) and productivity in CM.

  - We aim to develop an **NS gateway ontology** that allows several CM languages to be used and integrated with existing IT landscapes.

  - In addition to its current focus on representation and reasoning, we believe this can unlock **decades of knowledge, know-how, heuristics and experience** from this community, and make CM a cornerstone of more modular enterprises in the future.

Thank you for your attention !