Defining Domain Specific Modelling Languages to Automate Product Derivation: Collected Experiences

Authors: Juha-Pekka Tolvanen and Steven Kelly

Presented By: David Dietrich
CS846

Authors

- Juha-Pekka Tolvanen
- CEO of Metacase
- Blog:
  http://www.metacase.com/blogs/jpt/blogView

- Steven Kelly
- CTO of Metacase
- Blog:
  http://www.metacase.com/blogs/stevek/blogView

Metacase:

- Creators of MetaEdit+, a tool for creating domain specific models

Outline

- Introduction
- Study
- Conclusions
- Impact
- Further Work

Introduction

- Domain Specific Modelling Languages (DSML) raise the level of abstraction
  - Niche languages have high expressibility
  - Domain specific constraints can be assumed
  - Can translate to analyzable language
- All of the DSMLs in the study were used for code generation

Purpose

- Very little work has previously looked at DSML creation
  - Graphical DSML creation has seen even less
- Goal is to investigate and categorize approaches for defining DSMLs
  - Specific focus on automating variant creation

Study

- 23 industrial applications of DSM explored
- Qualitative study performed
  - Data gathered from Interviews and Discussions with language creators
- All languages were metamodel-based

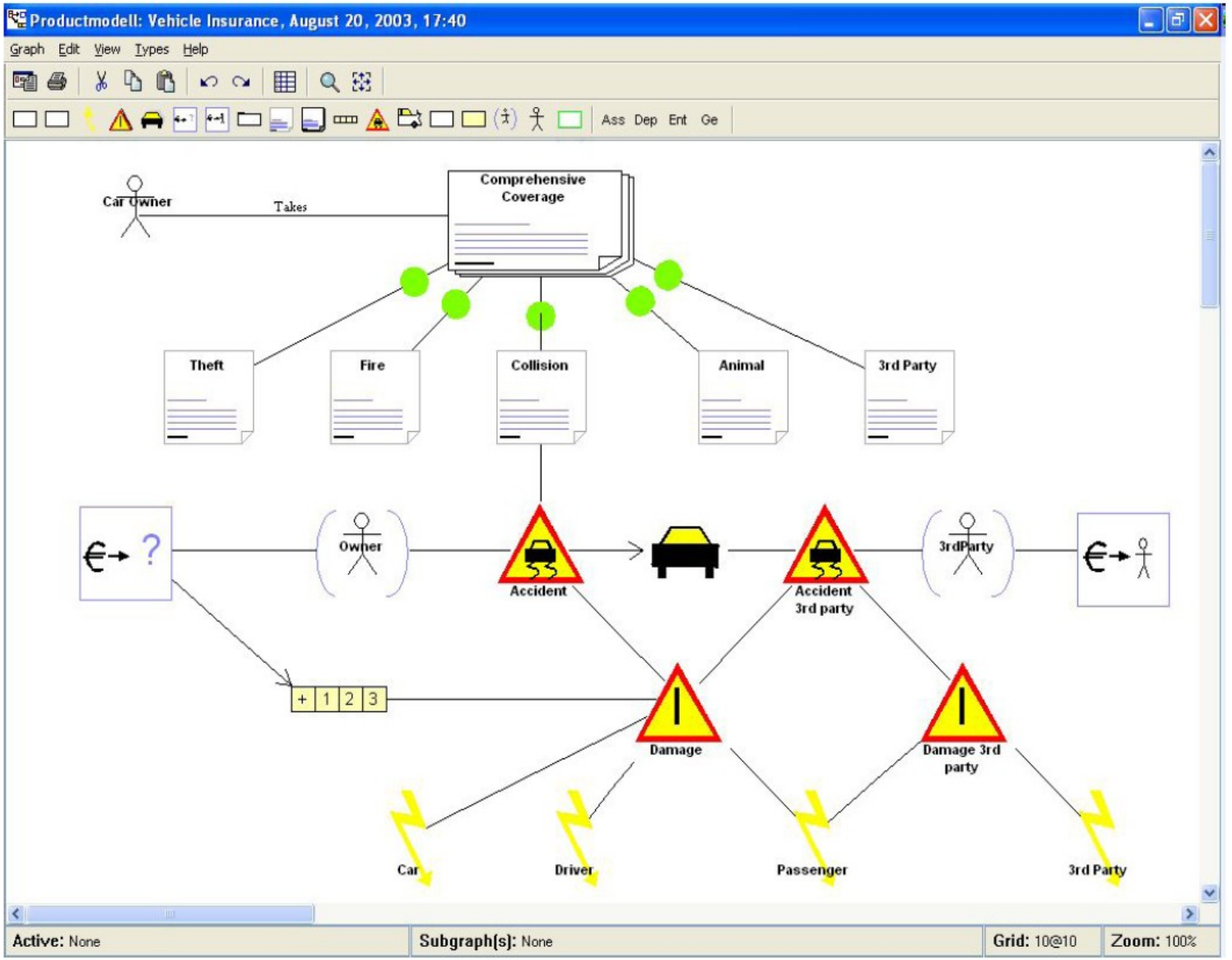
Study

- There were 4 different construction approaches discovered:
  - Domain Expert's Concepts
  - Generation Output
  - Look and Feel
  - Variability Space

Domain Expert's Concepts

- Model is based upon the semantics of the domain
- Can be used by people with little to no programming experience

Domain Expert's Concepts



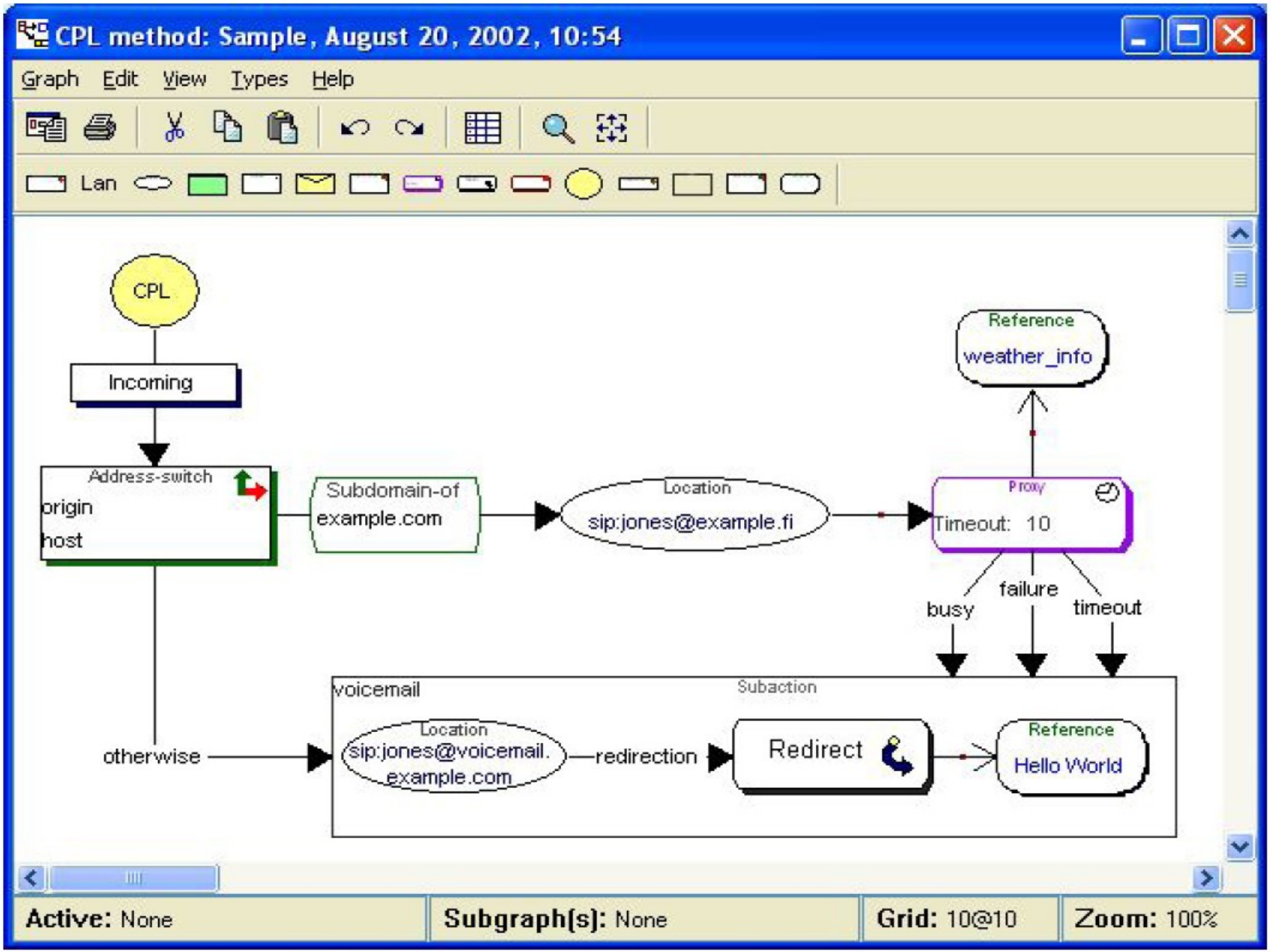
Domain Expert's Concepts

- Very abstract language
- Relatively easy to define
- Code generation guided by relationship between elements
- Need to be careful to ensure the domain is mature enough to handle this approach

Generation Output

- Modelling constructs derived from the structure of the generated code
- Similar to languages like UML
- Level of abstraction is not much higher than programming languages
  - Lower productivity improvement

Generation Output



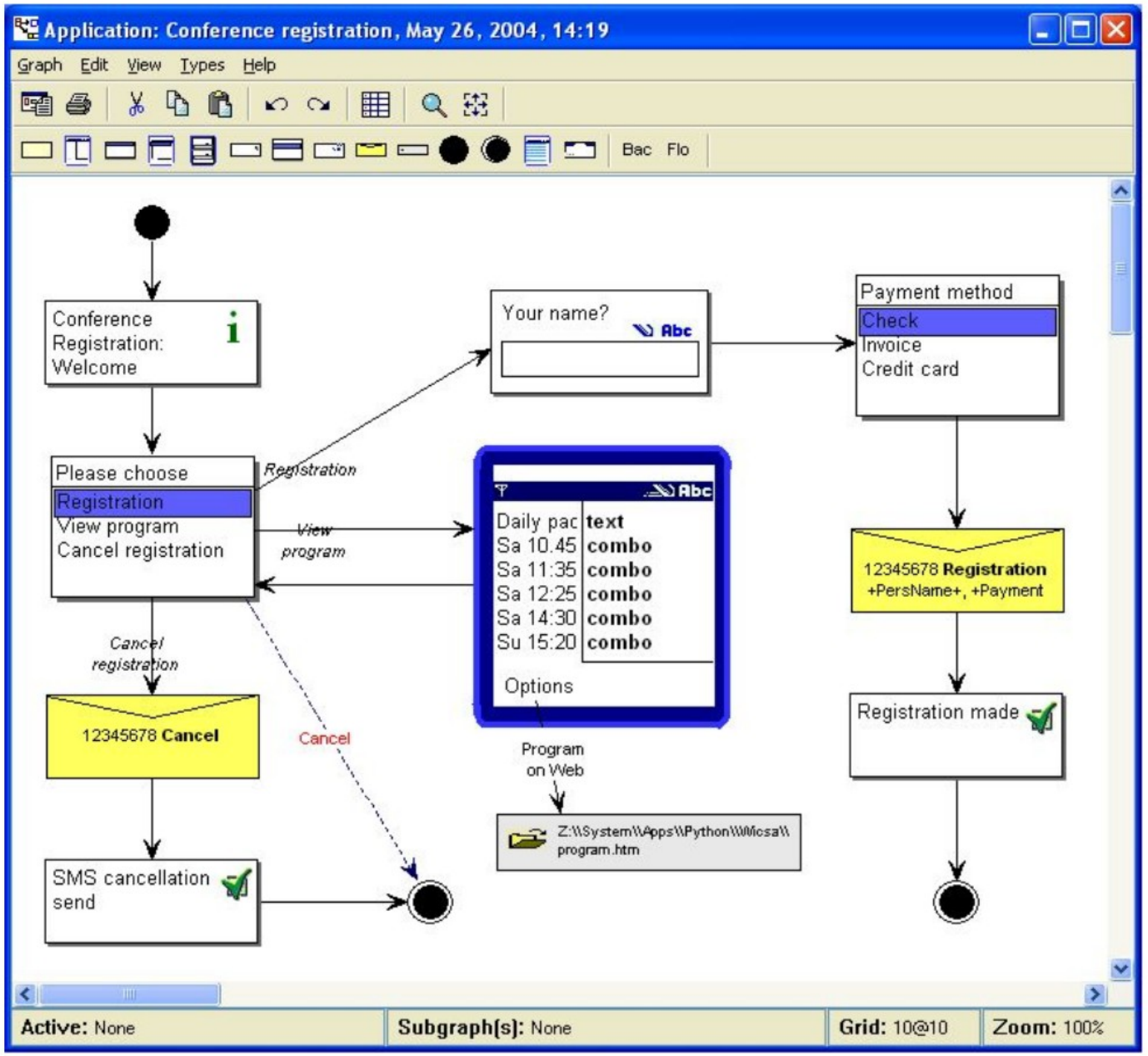
Generation Output

- The easy part is describing the static constructs
- Defining the behaviour was much more difficult
- Work best when the output is a domain specific language
- The output should already be mature for this approach to work

Look and Feel

- Modelling language applies end user concepts as modelling constructs
  - UI widgets and behaviour
- The most common type of DSM studied

Look and Feel



Look and Feel

- The “simplest” language to create
  - Everything is tangible and has an obvious meaning
  - Can be created with little knowledge of domain
- The difficult part is relating the behaviour to the structural elements
- Code generation implemented:
  - Per widget
  - State based

Variability

- Language based upon expressing variability
- Extremely useful for software product lining

Variability

- Variability languages need to be flexible in order to support any new features
- Creation requires an in depth analysis of the domain
- Feature modelling is too general to be applied to DSM concepts
- Most difficult DSML to define
- Combines with the Look and Feel approach

Conclusion

- In all cases DSMs lead to a productivity increase
- These languages can be used by developers with little programming experience
- In most cases the generated code interacted with a framework that was already created
- Combining multiple approaches seemed very common

Impact

- 112 citations
- Experience Reports and Case Studies are important for informing others about MBSE
- The guidelines outlined in the paper were used during research to define a DSL for Service Oriented Architectures [3]

Further Work

- Several more field studies and experience reports
- A majority of the work since 2005 looks like it has focused on variability DSMLs

Systematic Approach (2009) [1]

- Uses an experimental approach to find a systematic method for creation
- The method used can be applied when creating DSMLs

- J.P. Tolvanen also describes guidelines for creating DSMLs [2]

Discussion

- Could these graphical approaches be applied to textual modelling languages?
- Why are there no cases that use 3 or more approaches?

References

[1] An Approach for the Systematic Development of Domain Specific Languages

[2] http://www.devx.com/enterprise/Article/30550

[3] Domain Specific Languages for Service Oriented Architectures: An Explorative Study