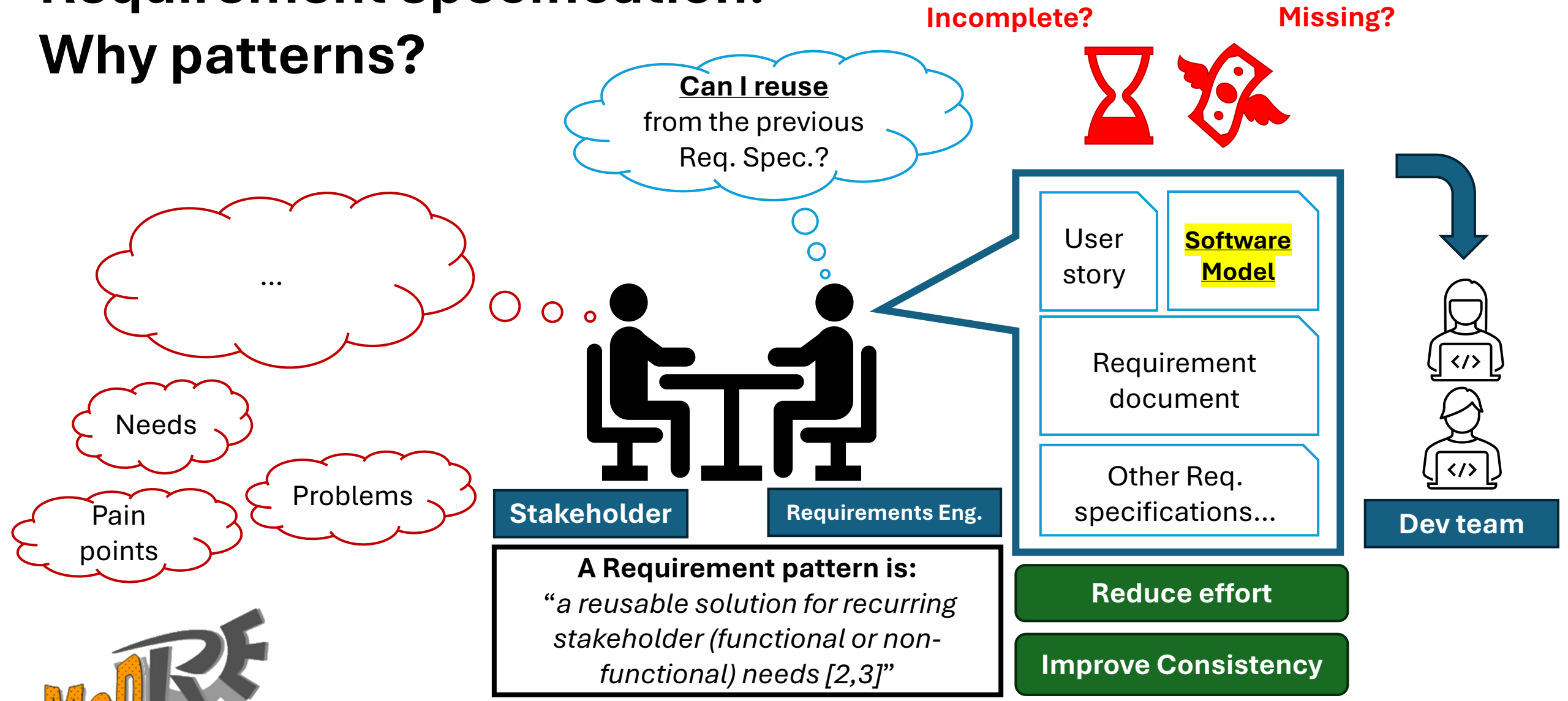
A Domain-Specific Language For Specifying Requirement Patterns for Model-Driven Software Development



David Mosquera[1,2], Marcela Ruiz[1], and Anastassios Martakos[1]

[1] *Zürich University of Applied Sciences, Winterthur 8400, Switzerland*
[2] *PROS-VRAIN: Valencian Research Institute for Artificial Intelligence – Universitat Politècnica de València, València, Spain*

Requirement specification: Why patterns?



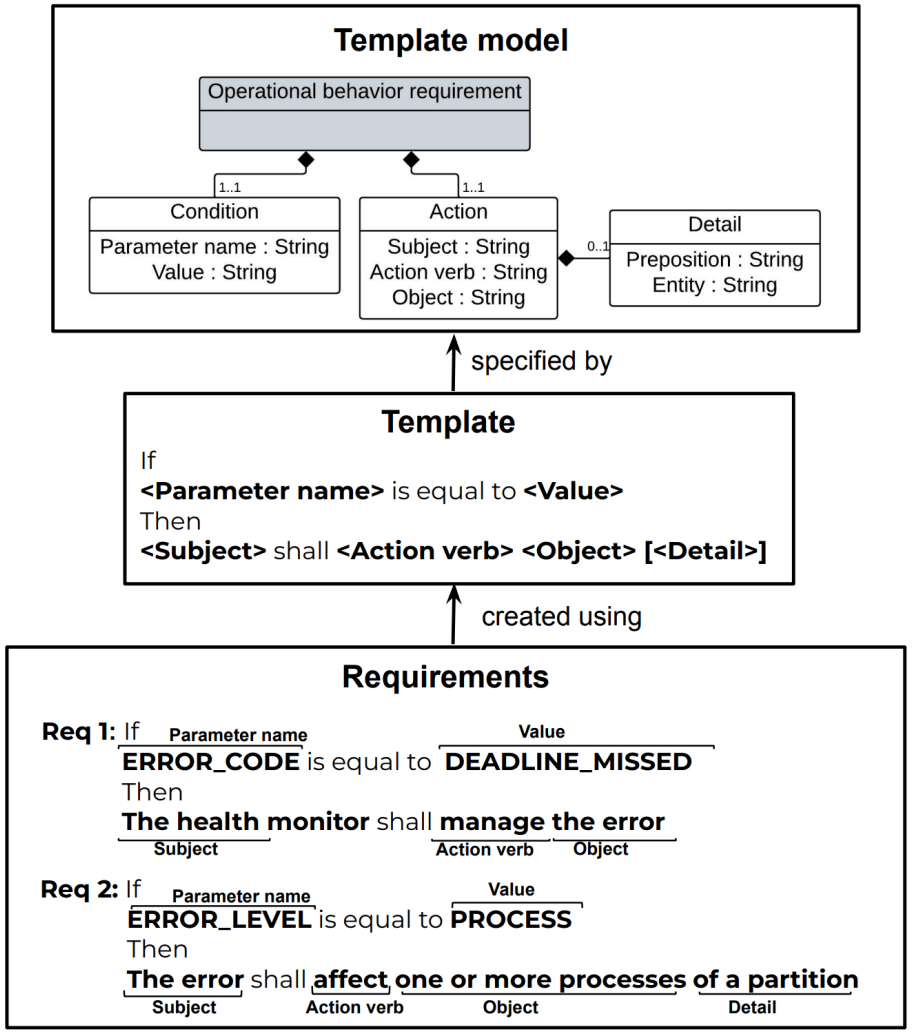
Requirement patterns in Req. Specification

Approaches that aim to reuse requirement patterns to generate

Text-based requirement documents

Requirement Pattern *Data Exchange*	**Description**	This pattern expresses the need of having the system functionality for importing and exporting data.	
	Comments	This is just an import and export data system functionality (usually a component).	
	Pattern goal	Exchange data with legacy and external systems	
	Author	SSI, GESSI	
	Sources (0..*)	Requirement books from SSI Specialized literature	
	Keywords (0..*)	Data Integration, Export Data, Import Data	
	Dependencies (0..*)	----	
	Fixed Part	**Question text**	----
		Form text	The system shall offer functionalities for importing and exporting data.
	Extended Part *Data Exchange Formats*	**Question text**	----
		Form text	The system shall be able to %exchangeDirection% data in %formNames% formats
		Parameter	**Metric**
		exchangeDirection: represents one of the possible directions for exchanging data	ExchangeDirection: ExchangeDirections= Domain(Import, Export, Import and Export
		formNames: is a non-empty set of names of data formats, if the value is Some Standard, then the size must be one	FormatTypes: FormatTypes = Set(FormatType) FormatType = Domain(Some Standard, XML,HTML,...)

PABRE [4,9]

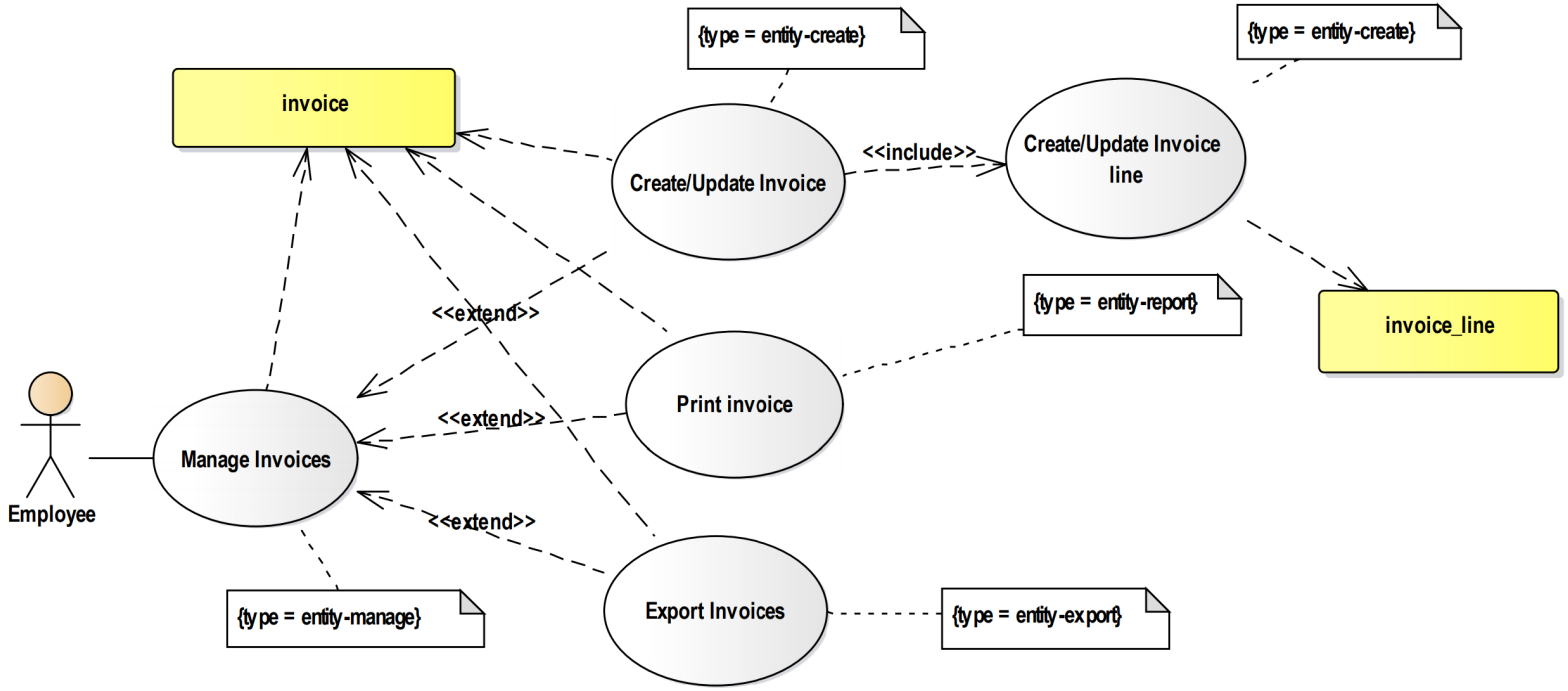


UTL [10]

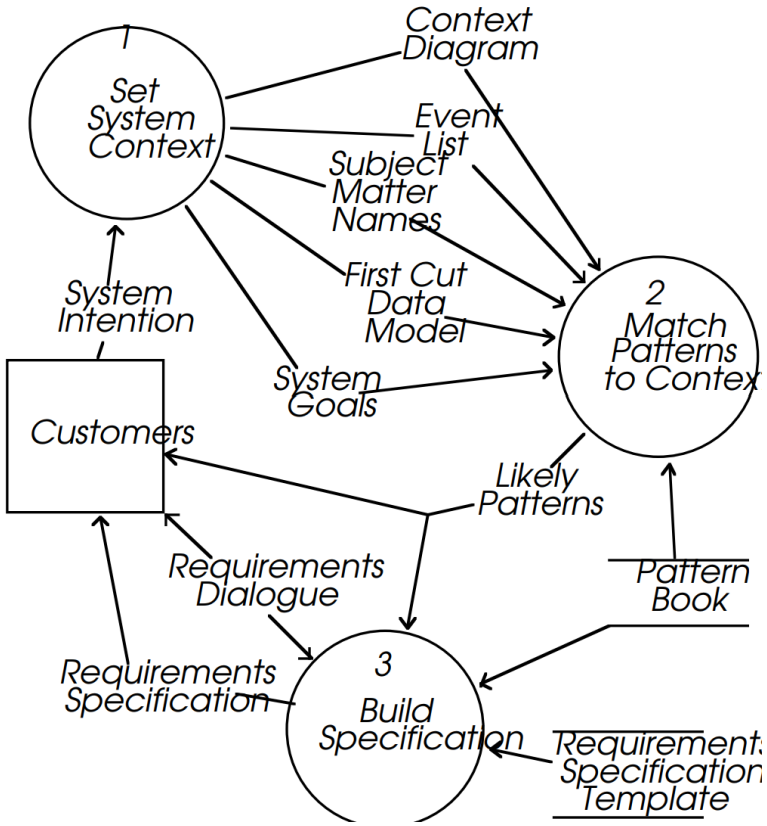
Requirement patterns in Req. Specification

Approaches that aim to reuse requirement patterns to generate

Software models for MDSD



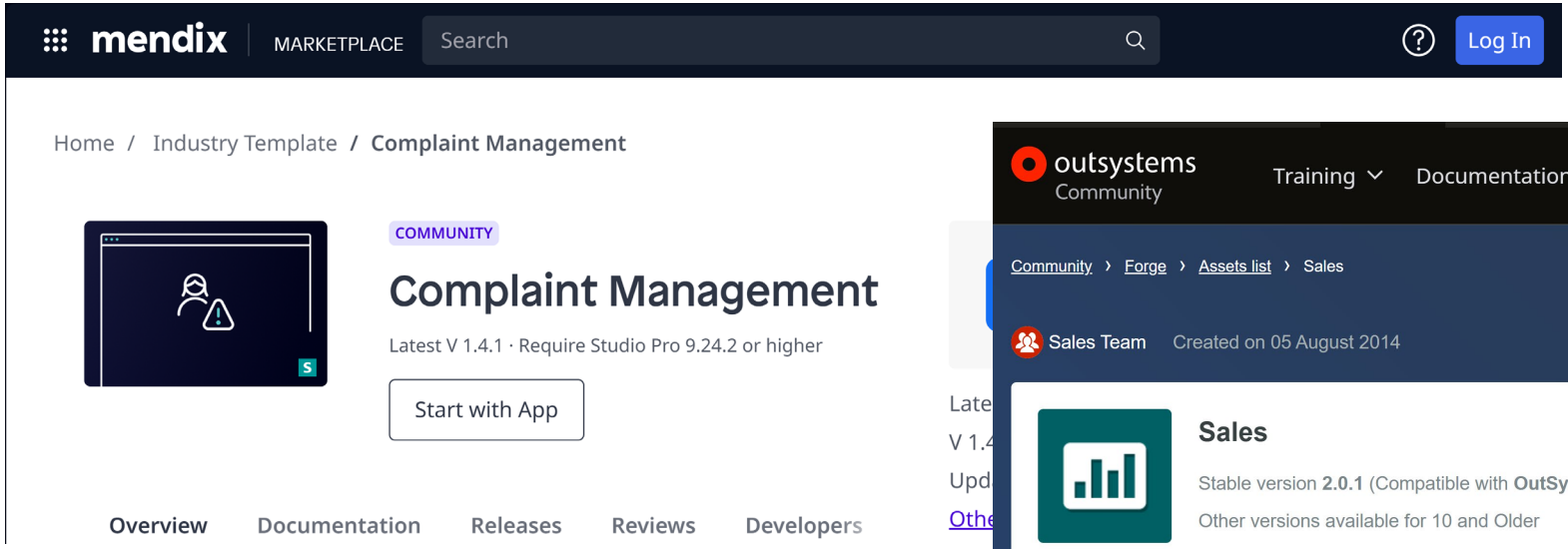
Silva et al. [12]



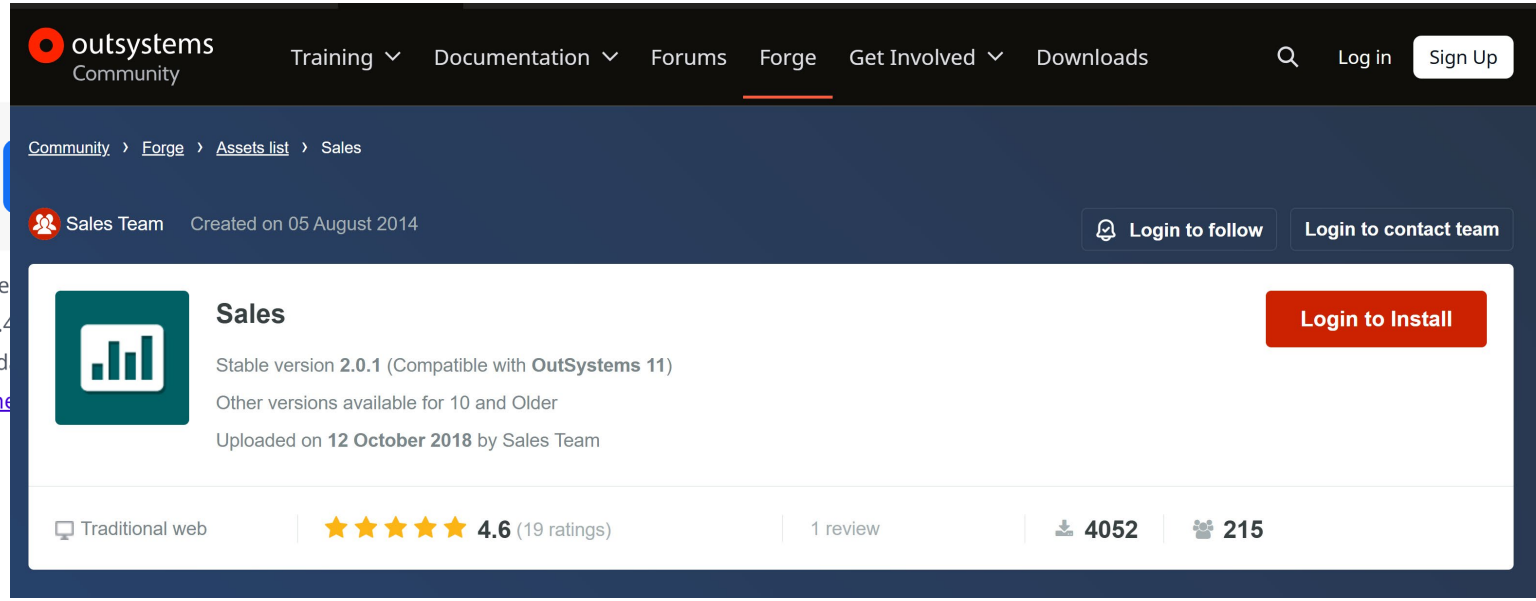
Robertson [11]

Requirement patterns in Req. Specification

Approaches that aim to reuse requirement patterns to generate
Software models for MDSD

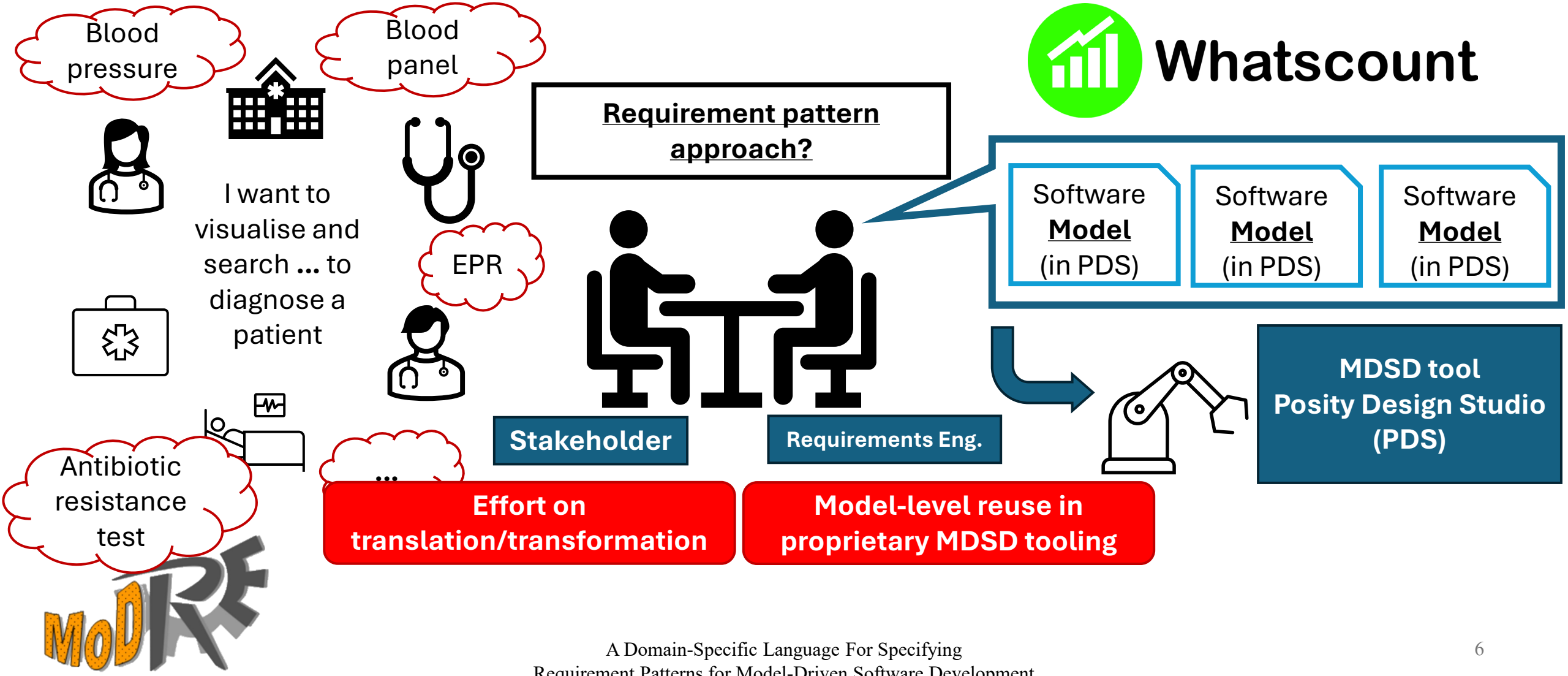


Mendix Templates
[13]

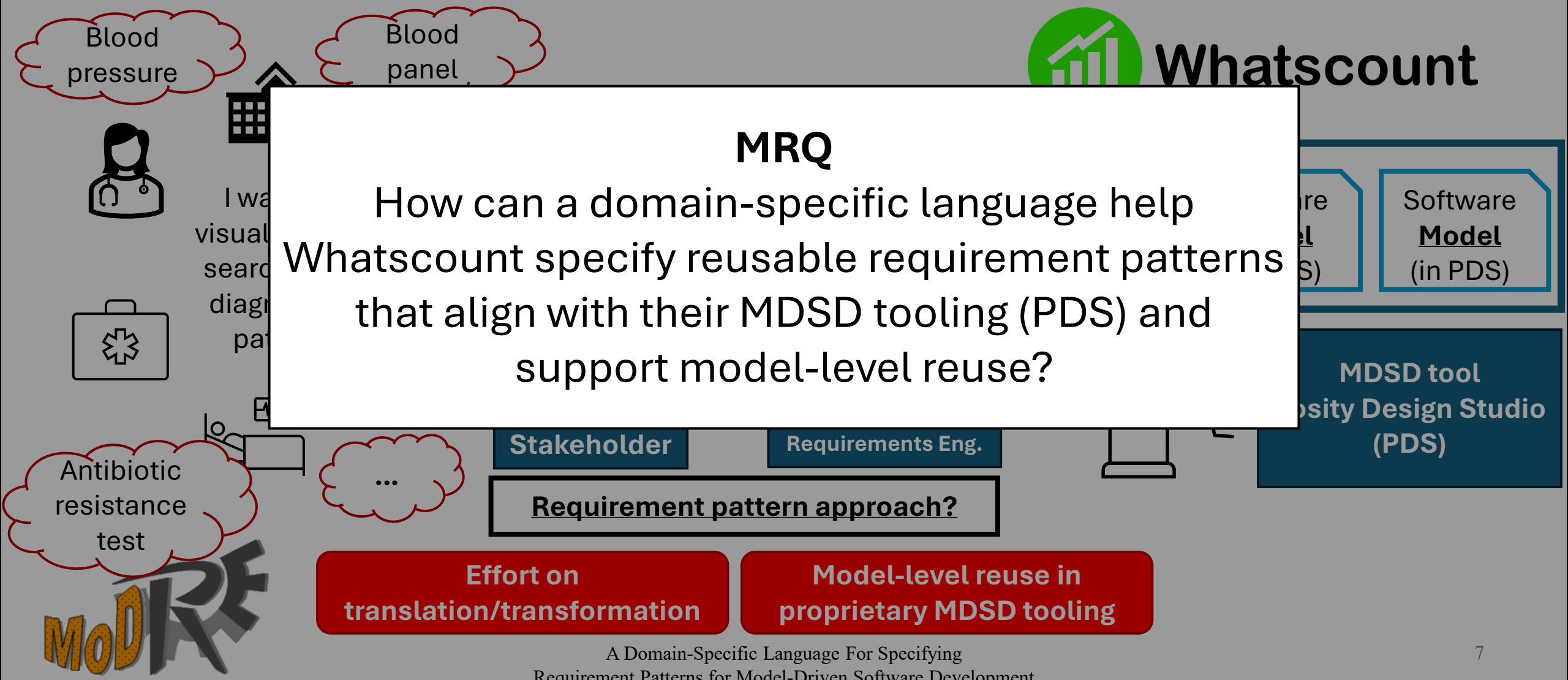


OutSystems Forge
[14]

The Case at Whatscount GmbH



The Case at Whatscount GmbH



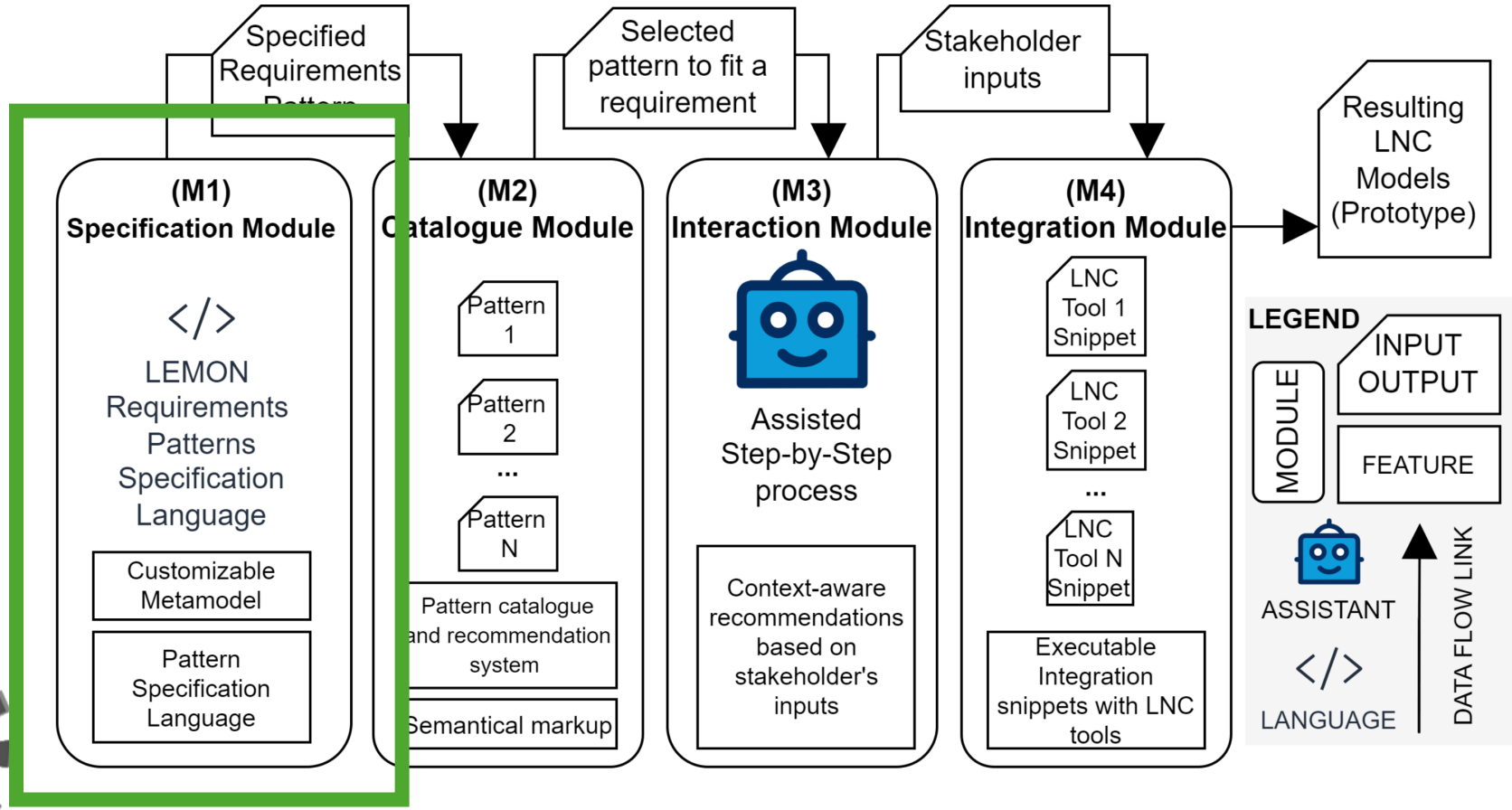
MRQ

How can a domain-specific language help Whatscount specify reusable requirement patterns that align with their MDSD tooling (PDS) and support model-level reuse?

Our approach: LEMON

Our approach:

Language for sp**E**cifiying and **MO**delling (Requirement) patter**N**s

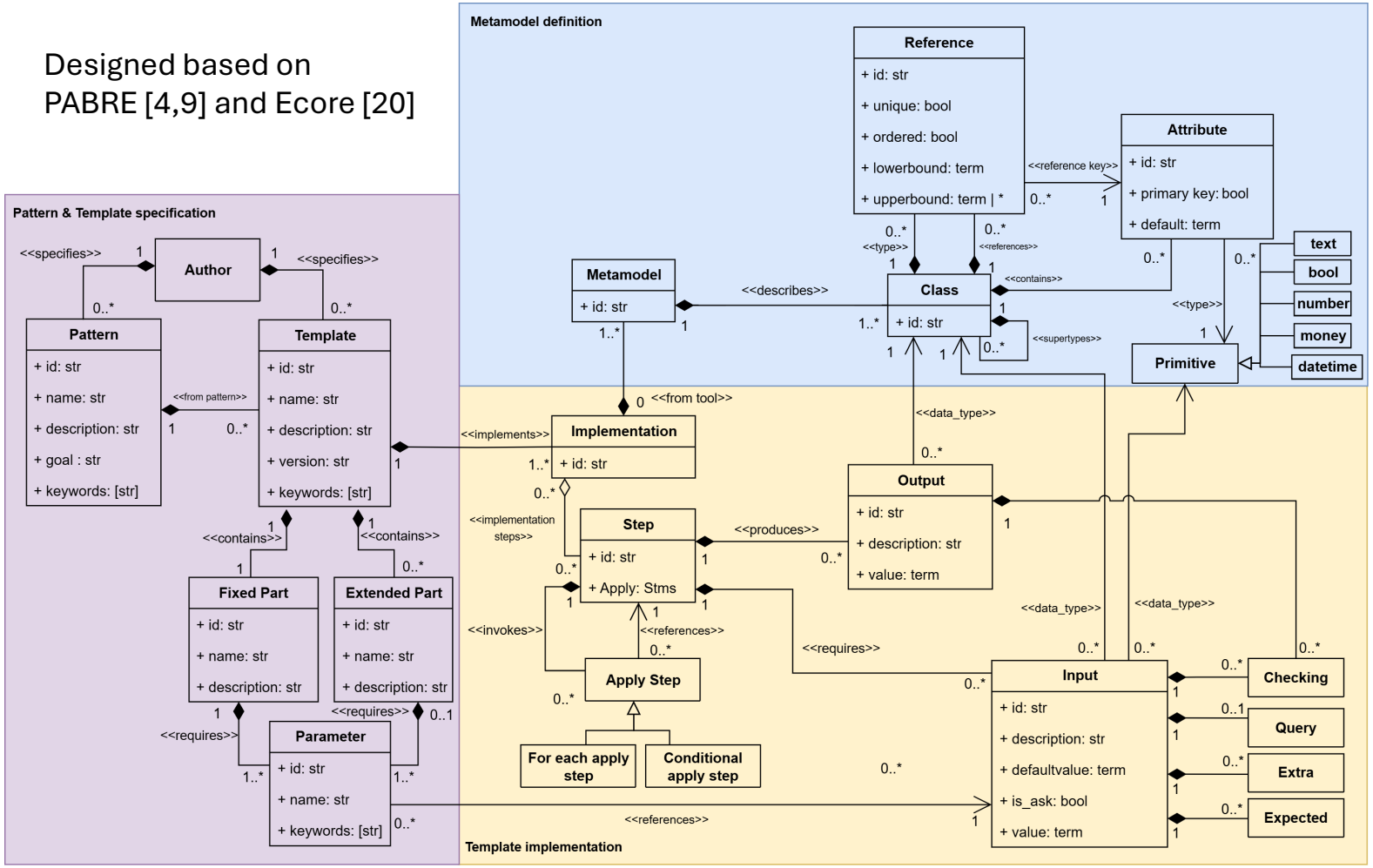


The LEMON Framework [15]

The LEMON (Language) Metamodel

1. Pattern & Template Specification
2. Metamodel Definition
3. Template Implementation

Designed based on
PABRE [4,9] and Ecore [20]



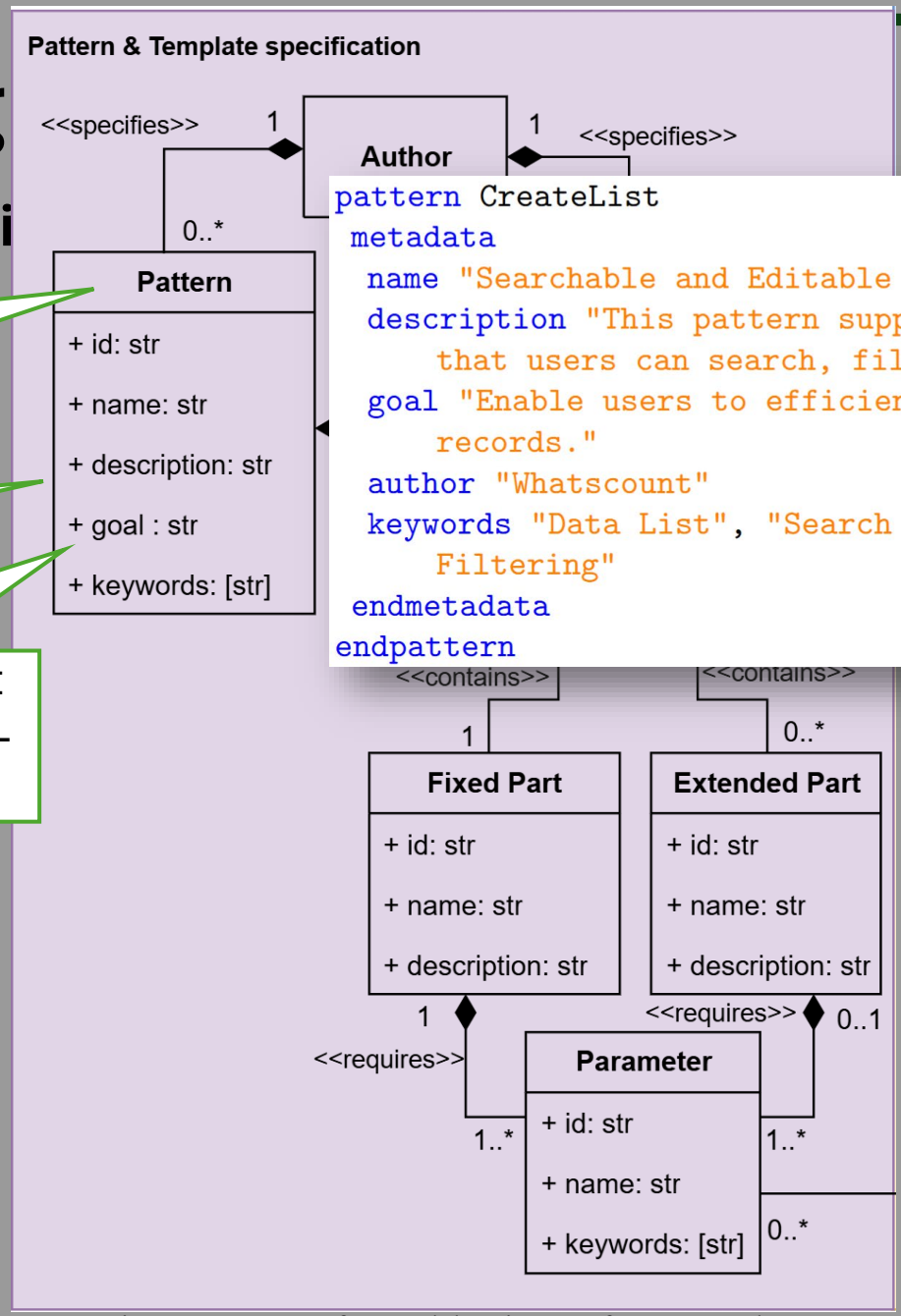
The LEMON (Lang

Pattern & Template Specifi

Named, self-contained language element

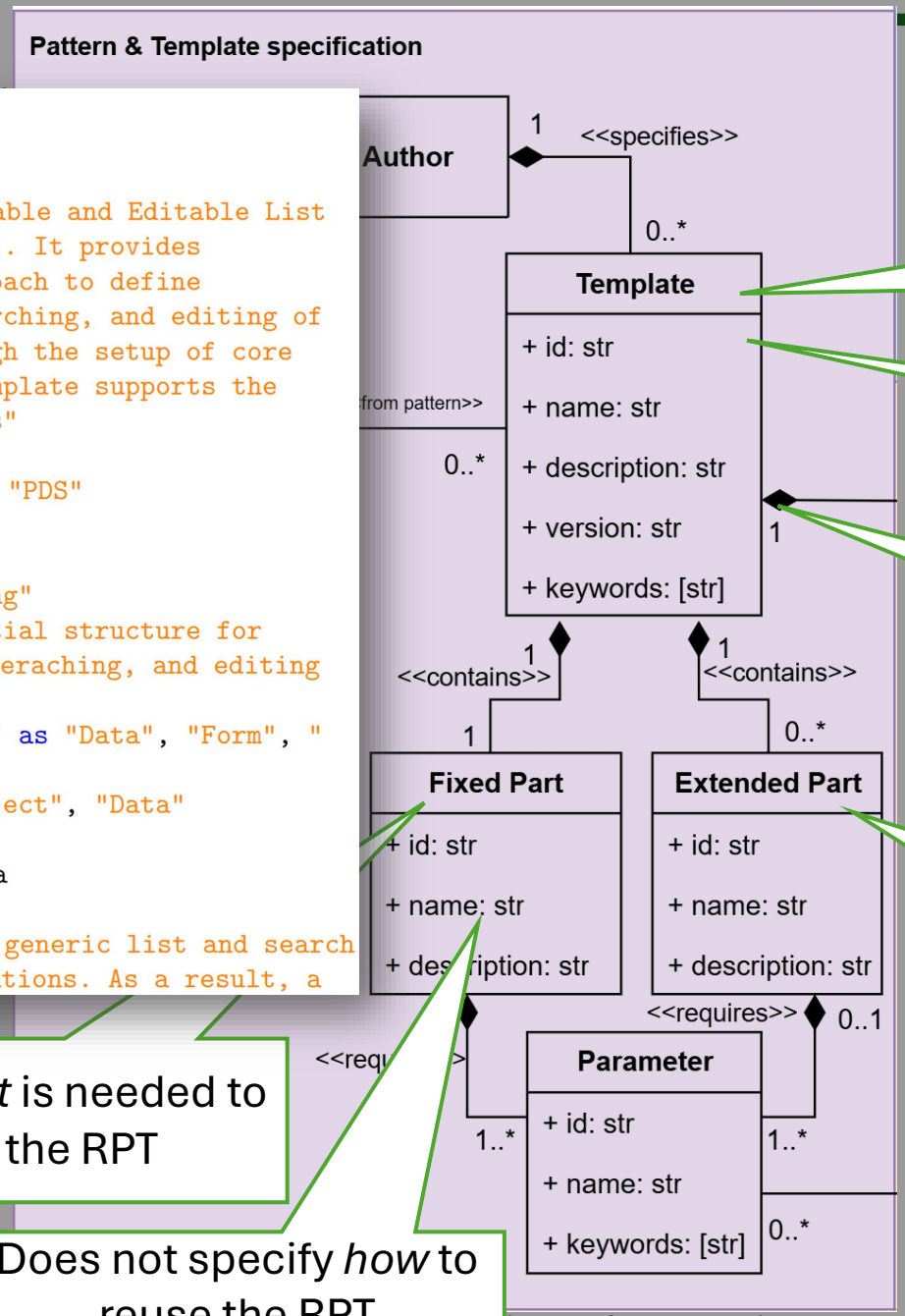
Abstract and tool-independent

Describe reusable requirement specification (functional or non-functional)




```
pattern CreateList
 metadata
  name "Searchable and Editable List"
  description "This pattern supports presenting structured data in a list
      that users can search, filter, and edit."
  goal "Enable users to efficiently view, search and edit structured data
      records."
  author "Whatscount"
  keywords "Data List", "Search functionality", "Editable Entries", "
      Filtering"
 endmetadata
endpattern
```

```
template F_PDSListToSearchAndEdit
  from pattern CreateList
  name "List and Search in PDS"
  description "This template implements the Searchable and Editable List
      pattern within the Posity Design Studio (PDS). It provides
      requirement engineers with a structured approach to define
      QueryModels that enable dynamic listing, searching, and editing of
      domain-specific data. By guiding users through the setup of core
      data tables and their relationships, this template supports the
      efficient visualisation of structured records"
 author "Whatscount and Posity AG"
 keywords "Visualization", "QueryModel", "Search", "PDS"
 version "1.0"
 fixedpart FP_PDSGenericListAndEdit
    name "Generic QueryModel for Listing and Editing"
    description "This fixed part defines the essential structure for
        creating a QueryModel in PDS for listing, seraching, and editing
        data"
    parameter I_QueryName "Name of the query model" as "Data", "Form", "
        Object"
    parameter I_Table "Primary data source" as "Object", "Data"
 endfixedpart
 extendedpart EP_DigitalHealthListAndSearchTestData
   name "Digital Health Test Data View"
   description "This extended part specialises the generic list and search
        pattern template for digital health applications. As a result, a
```



Share the same goal as its base pattern

Less abstract/more granular

Implemented in one or more MDSD tools

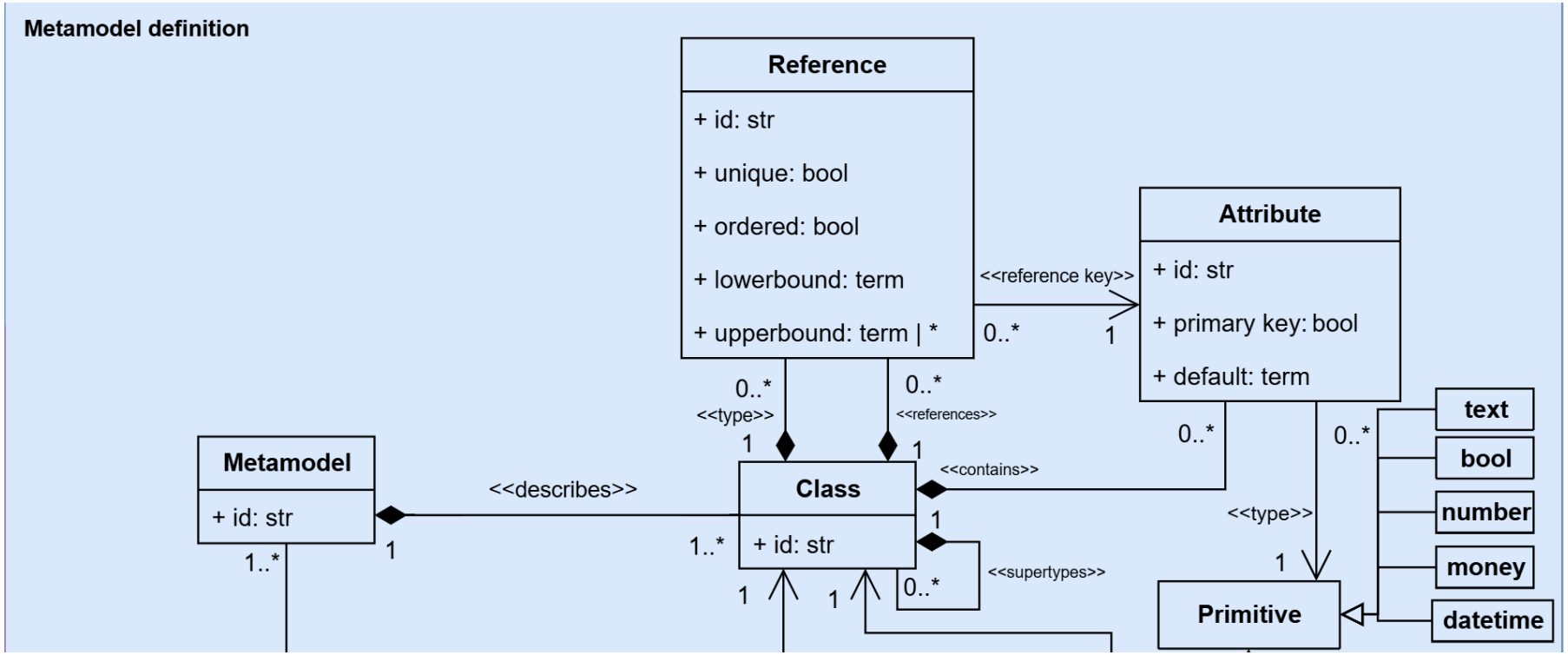
Extend the fixed part for specific domains, tools, or constraints

States *what* is needed to reuse the RPT

Does not specify *how* to reuse the RPT

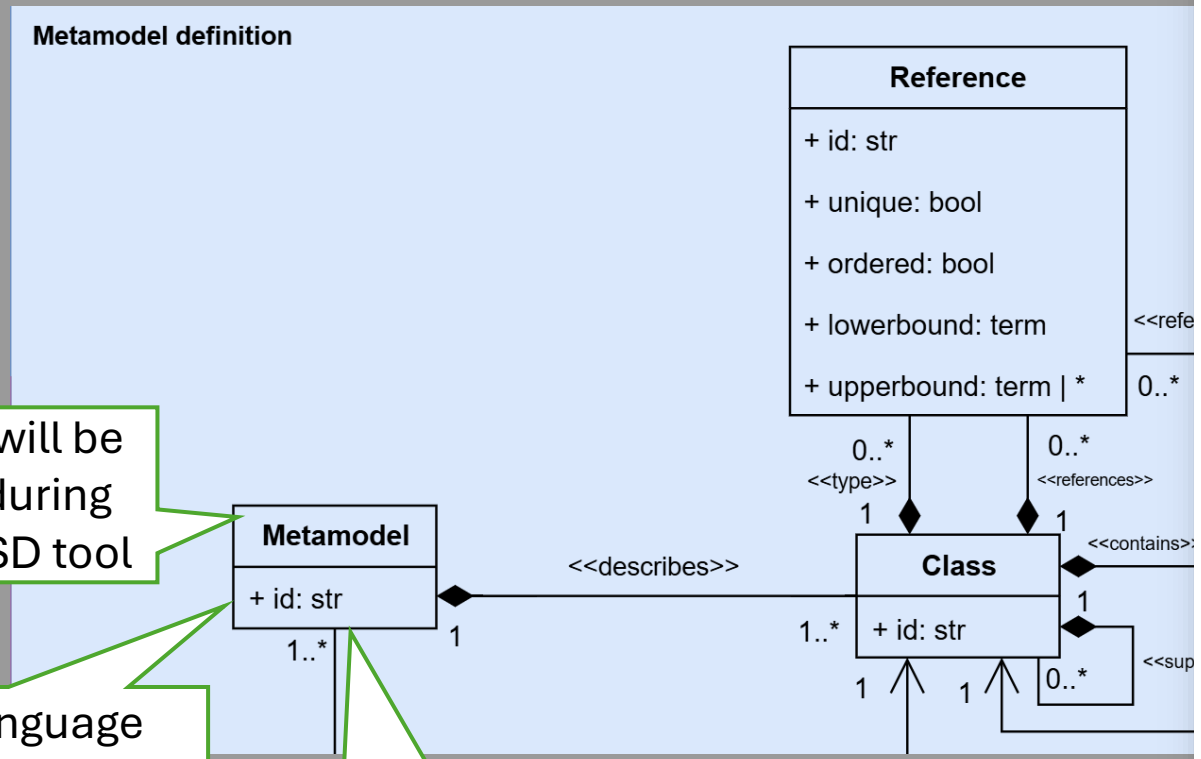
The LEMON (Language) Metamodel

Metamodel definition



The LEMON (Language) Metamodel

Metamodel definition



Defines what will be instantiated during reuse in a MDSD tool

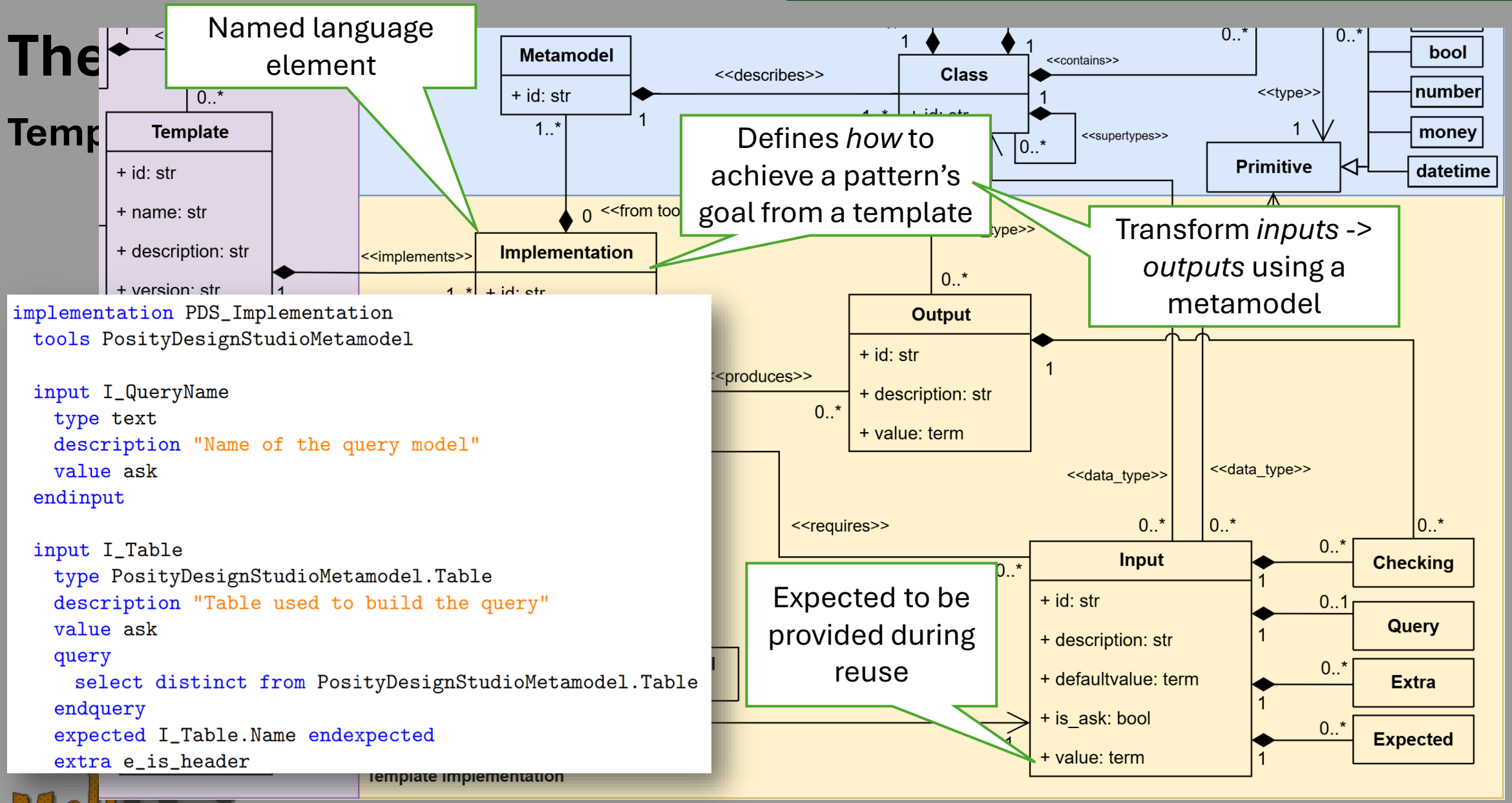
Named language element

Groups MDSD tool classes

```
metamodel PosityDesignStudioMetamodel
 class Table
   attribute Name
     type text
     default "Default table name"
     primarykey true
   endattribute
 endclass
 class QueryModel
   attribute Name
     type text
     default "Default Query model name"
     primarykey true
   endattribute
   reference FKTableOfQueryModel
     type Table
     ordered false
     unique false
     keys Table.Name
     lowerbound 0
     upperbound *
   endreference
 endclass
endmetamodel
```

The

Temp



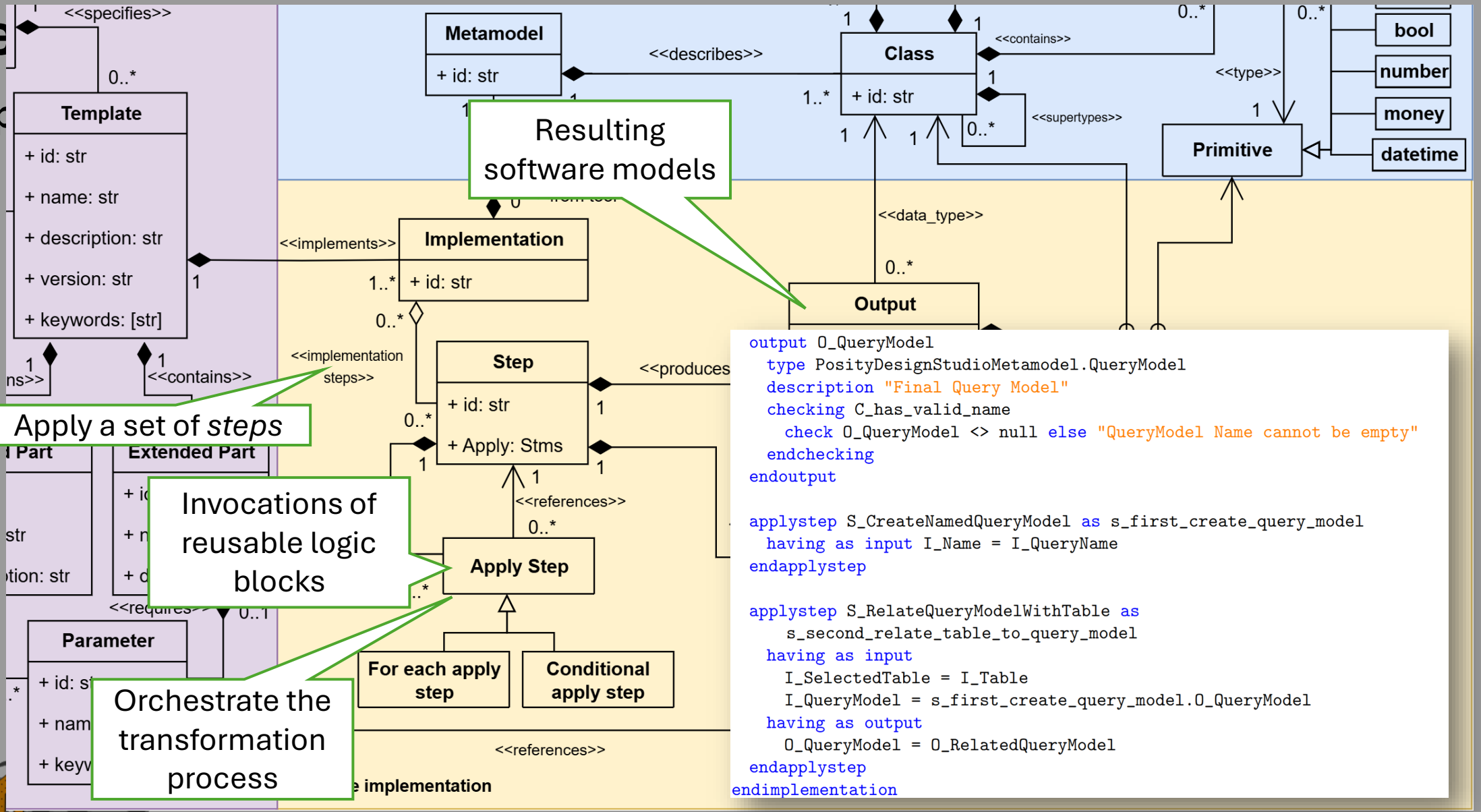

```
implementation PDS_Implementation
  tools PosityDesignStudioMetamodel

  input I_QueryName
    type text
    description "Name of the query model"
    value ask
  endinput

  input I_Table
    type PosityDesignStudioMetamodel.Table
    description "Table used to build the query"
    value ask
    query
      select distinct from PosityDesignStudioMetamodel.Table
    endquery
    expected I_Table.Name endexpected
    extra e_is_header
```

The

Temp

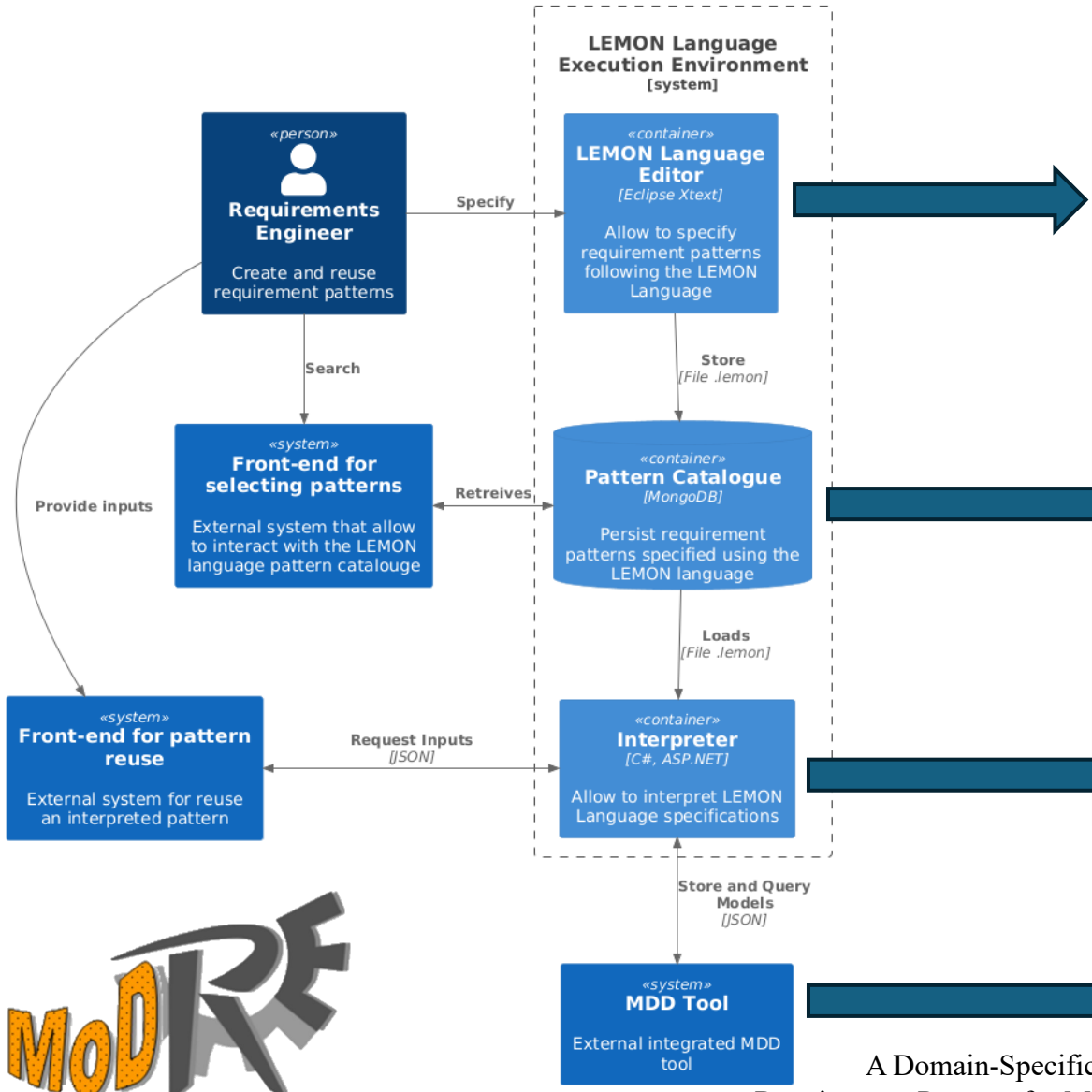


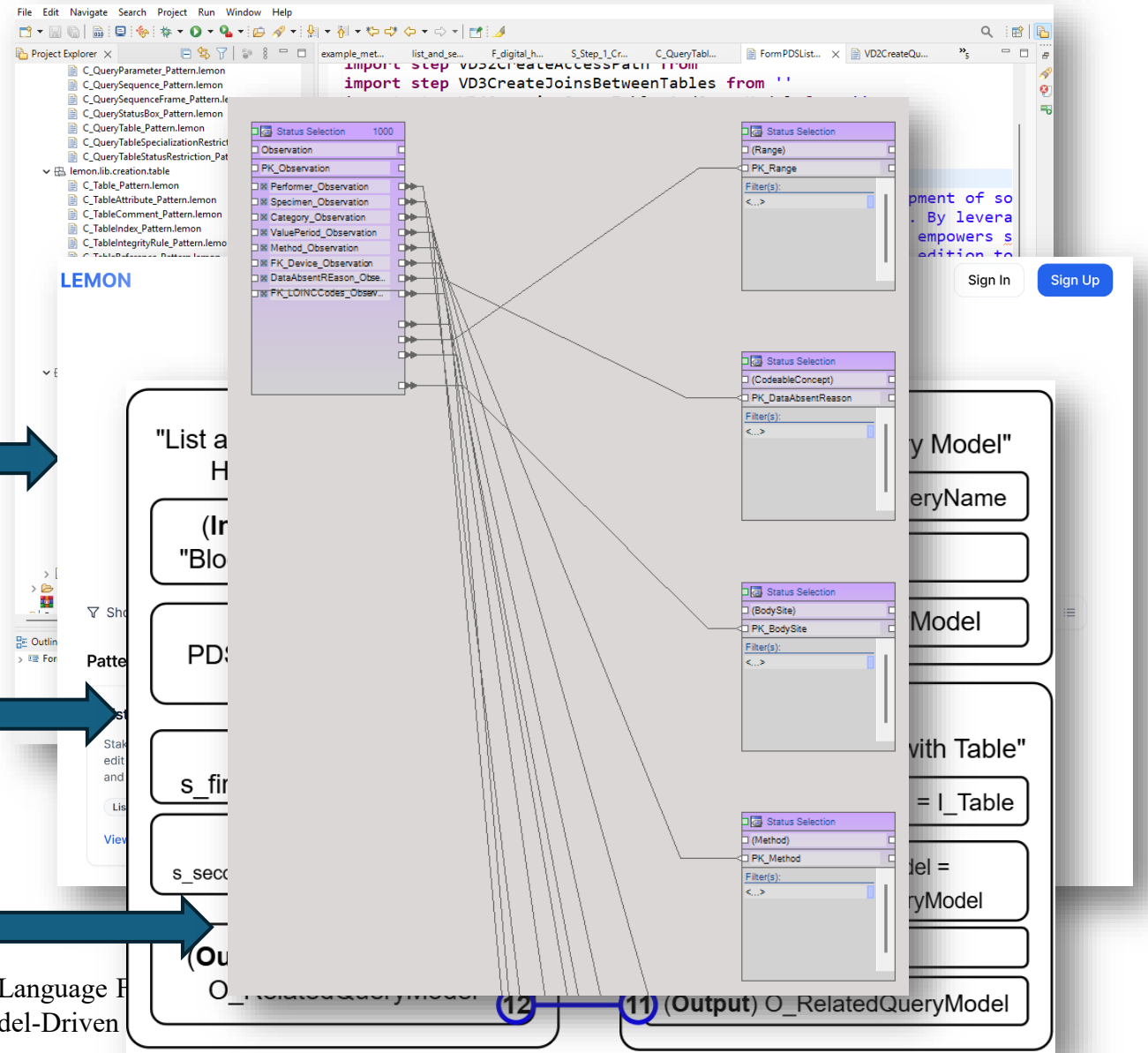

```
output O_QueryModel
  type PosityDesignStudioMetamodel.QueryModel
  description "Final Query Model"
  checking C_has_valid_name
    check O_QueryModel <> null else "QueryModel Name cannot be empty"
  endchecking
endoutput

applystep S_CreateNamedQueryModel as s_first_create_query_model
  having as input I_Name = I_QueryName
endapplystep

applystep S_RelateQueryModelWithTable as
    s_second_relate_table_to_query_model
  having as input
    I_SelectedTable = I_Table
    I_QueryModel = s_first_create_query_model.O_QueryModel
  having as output
    O_QueryModel = O_RelatedQueryModel
endapplystep
endimplementation
```

The LEMON Execution Environment

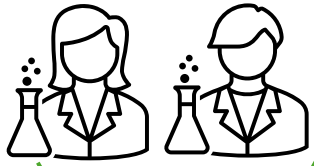




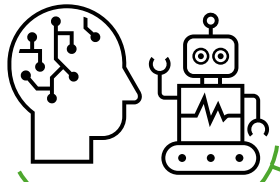
A Domain-Specific Language F
Requirement Patterns for Model-Driven

Conclusion and Open Research Questi

RQ1. How do requirements engineers **specify, modify, and reuse patterns** with LEMON?



RQ2. Can LEMON-based pattern specification be **automated**?



Contribution:

Designed LEMON, a domain-specific language bridging reusable requirement patterns and MDSD tools, enabling model-level reuse

RQ3. How do requirements engineers **interact** with the pattern **catalogue** & **reuse** interface?

References

[1] A. Gupta, G. Poels, and P. Bera, “Using Conceptual Models in Agile Software Development: A Possible Solution to Requirements Engineering Challenges in Agile Projects,” *IEEE Access*, vol. 10, pp. 119745–119766, 2022, doi: 10.1109/ACCESS.2022.3221428.
[2] T. N. Kudo, R. F. Bulcão-Neto, and A. M. R. Vincenzi, “Requirement patterns: a tertiary study and a research agenda,” *IET Software*, vol. 14, no. 1, pp. 18–26, Feb. 2020, doi: 10.1049/iet-sen.2019.0016.
[3] P. Mahendra and A. Ghazarian, “Patterns in the Requirements Engineering: A Survey and Analysis Study,” in *WSEAS Transactions of Information Science and Applications,* 2014, pp. 214–230.
[4] S. Renault, O. Mendez-Bonilla, X. Franch, and C. Quer, “PABRE: Pattern-based Requirements Elicitation,” in *2009 Third International Conference on Research Challenges in Information Science*, IEEE, Apr. 2009, pp. 81–92. doi: 10.1109/RCIS.2009.5089271.
[5] K. Kumar and Ra. K. Saravanaguru, “CONTEXT AWARE REQUIREMENT PATTERNS (CaRePa) METHODOLOGY AND ITS EVALUATION,” *Far East Journal of Electronics and Communications*, vol. 16, no. 1, pp. 101–117, Feb. 2016, doi: 10.17654/EC016010101.
[6] L. Sardi, A. Idri, L. Redman, H. Alami, and J. Fernández-Alemán, “A Reusable Catalog of Requirements for Gamified Mobile Health Applications,” in *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering*, SCITEPRESS - Science and Technology Publications, 2022, pp. 435–442. doi: 10.5220/0011071700003176.
[7] A. Sleimi, M. Ceci, M. Sabetzadeh, L. C. Briand, and J. Dann, “Automated Recommendation of Templates for Legal Requirements,” in *Proceedings of the IEEE International Conference on Requirements Engineering*, IEEE Computer Society, Aug. 2020, pp. 158–168. doi: 10.1109/RE48521.2020.00027.
[8] C. Denger, D. M. Berry, and E. Kamsties, “Higher quality requirements specifications through natural language patterns,” in *Proceedings 2003 Symposium on Security and Privacy*, IEEE, 2003, pp. 80–90. doi: 10.1109/SWSTE.2003.1245428.
[9] X. Franch, S. Gnesi, F. Paccosi, C. Quer, and L. Semini, “Leveraging Requirements Elicitation through Software Requirement Patterns and LLMs,” in REFSQ2025, 2025, pp. 261–276. doi: 10.1007/978-3-031-88531-0_19.
[10] I. Darif, G. El Boussaidi, S. Kpodjedo, and A. Paz, “UTL: A Unified Language for Requirements Templates,” in *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, 2025, pp. 1489–1506. doi: 10.1145/3672608.3707911.
[11] S. Robertson, “Requirements Patterns Via Events/Use Cases,” in PLoP, 1996, pp. 1-16.
[12] A. R. da Silva *et al.*, “A pattern language for use cases specification,” in *Proceedings of the 20th European Conference on Pattern Languages of Programs*, New York, NY, USA: ACM, Jul. 2015, pp. 1–18. doi: 10.1145/2855321.2855330.
[13] Mendix, “Mendix Sample and Starter App Catalogue.” Accessed: Mar. 30, 2025. [Online]. Available: https://marketplace.mendix.com/link/contenttype/102,109
[14] OutSystems, “OutSystems: Application Templates.” Accessed: Jul. 11, 2024. [Online]. Available: https://success.outsystems.com/documentation/11/building_apps/application_templates/
[15] D. Mosquera, O. Pastor, and J. Spielberger, “LEMON: A Tool for Enhancing Software Requirements Communication through Requirements Pattern-based Modelling Assistance,” in *Posters & Tools Track at REFSQ2024,* 2024.
[16] I. Darif, G. El Boussaidi, and S. Kpodjedo, “On the Automated Generation of UI for Template-based Requirements Specification,” in *MO2RE*, 2025.
[17] S. Sendall and W. Kozaczynski, “Model transformation: the heart and soul of model-driven software development,” *IEEE Softw*, vol. 20, no. 5, pp. 42–45, Sep. 2003, doi: 10.1109/MS.2003.1231150.
[18] Posity AG, “Posity Design Studio Homepage.” Accessed: May 31, 2025. [Online]. Available: https://posity.ch
[19] Wikipedia, “Extended Backus-Naur Form.” Accessed: May 31, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Extended_Backus–Naur_form
[20] Eclipse Foundation, “Ecore: A Metamodel for Models.” Accessed: Mar. 07, 2024. [Online]. Available: https://wiki.eclipse.org/Ecore
[21] D. Mosquera, M. Ruiz, and A. Martakos, “A Domain-Specific Language For Specifying Requirement Patterns for Model-Driven Development - EBNF and Execution Envrionment POC.” Accessed: Jun. 01, 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15601580
[22] XText, “XText Eclipse Homepage.” Accessed: May 31, 2025. [Online]. Available: https://eclipse.dev/Xtext
[23] Eclipse, “Eclipse Modelling Framework.” Accessed: May 31, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Eclipse_Modeling_Framework