ER'2008 Tutorial: Can Domain Modeling become automated?

Iris Reinhartz-Berger

Department of Management Infromation Systems

University of Haifa, Israel

Agenda

- **An overview of Domain Engineering**
  - Domain Engineering concepts
  - The two layer model and its related activities
- **Semi Automatic Creation of Domain Models**
  - The ADOM approach
  - Knowledge Elicitation in ADOM
  - The ADOM supporting tools
  - Experimenting with ADOM
- **Summary and Discussion**

Agenda

- **An overview of Domain Engineering**
  - **Domain Engineering concepts**
  - The two layer model and its related activities
- **Semi Automatic Creation of Domain Models**
  - The ADOM approach
  - Knowledge Elicitation in ADOM
  - The ADOM supporting tools
  - Experimenting with ADOM
- **Summary and Discussion**

Domain Engineering Concepts:
What is a Domain?

- A ***domain*** is an area of knowledge characterized by a set of concepts, their relationships, and constraints accepted by practitioners in that area.
  - a set of applications that use a common jargon for describing the concepts, problems, and solutions
  - a class of similar systems that share common features and operations
- Other names for the term 'domain': 'product line', 'product family', …
- Domain attributes: mature, stable, economically viable
- Domain examples: Telephone switches, Insurance portals, Customer Relation Management, Online banking

Domain Engineering Concepts:
What is Domain Engineering?

- ***Domain engineering*** is the development and evolution of domain specific knowledge and artifacts to support the development and evolution of systems in the domain.
  - The purpose of domain engineering is to identify, model, construct, catalog, and disseminate the ***commonalities*** and ***differences*** of particular domain applications
  - Domain engineering includes engineering of domain models, components, methods and tools
  - Domain engineering includes three main activities: domain analysis, domain design, and domain implementation

Domain Engineering Concepts:
What is Domain Analysis?

- ***Domain analysis*** (The Free Dictionary by Farlex)
  - Is the domain engineering activity in which domain knowledge is studied and formalized as a domain definition and a domain specification
    - Relates to non-implementation issues
  - Is the process of identifying, collecting, organizing, analyzing and representing a domain model and software architecture from the study of existing systems, underlying theory, emerging technology and development histories within the domain of interest
  - Is the analysis of systems within a domain to discover commonalities and differences among them
  - A common way for carrying out domain analysis is **modeling**

Domain Engineering Concepts:
What is Domain Analysis?

- ***Domain analysis*** includes:
  - **Set objectives:** identify stakeholders and their objectives
  - **Scope domain:** define selection criteria
  - **Define domain:** identify boundary conditions, examples, counter examples, main features
  - **Define relations:** define relations to other domains, divide the domain into sub-domains
  - **Acquire domain information** from experts, legacy systems, literature, prototyping
  - **Describe domain terminology:** lexicon of terms, commonality & variability, features
  - **Refine domain:** build overall domain, analyze trade offs, innovative feature combinations

Domain Engineering Concepts:
What are Domain Design and Domain Implementation?

- ***Domain design*** is the activity that takes the results of domain analysis to identify and generalize solutions for those common requirements in the form of a Domain-Specific Software Architecture (DSSA)

  (Carnegie-Mellon Software Engineering Institute)

  - It focuses on the problem space, not just on a particular system's requirements, to design a solution (solution space)
  - It supports developing a common architecture for the system in the domain and devising a production plan

- ***Domain implementation*** is the process of identifying reusable components based on the domain model and generic architecture

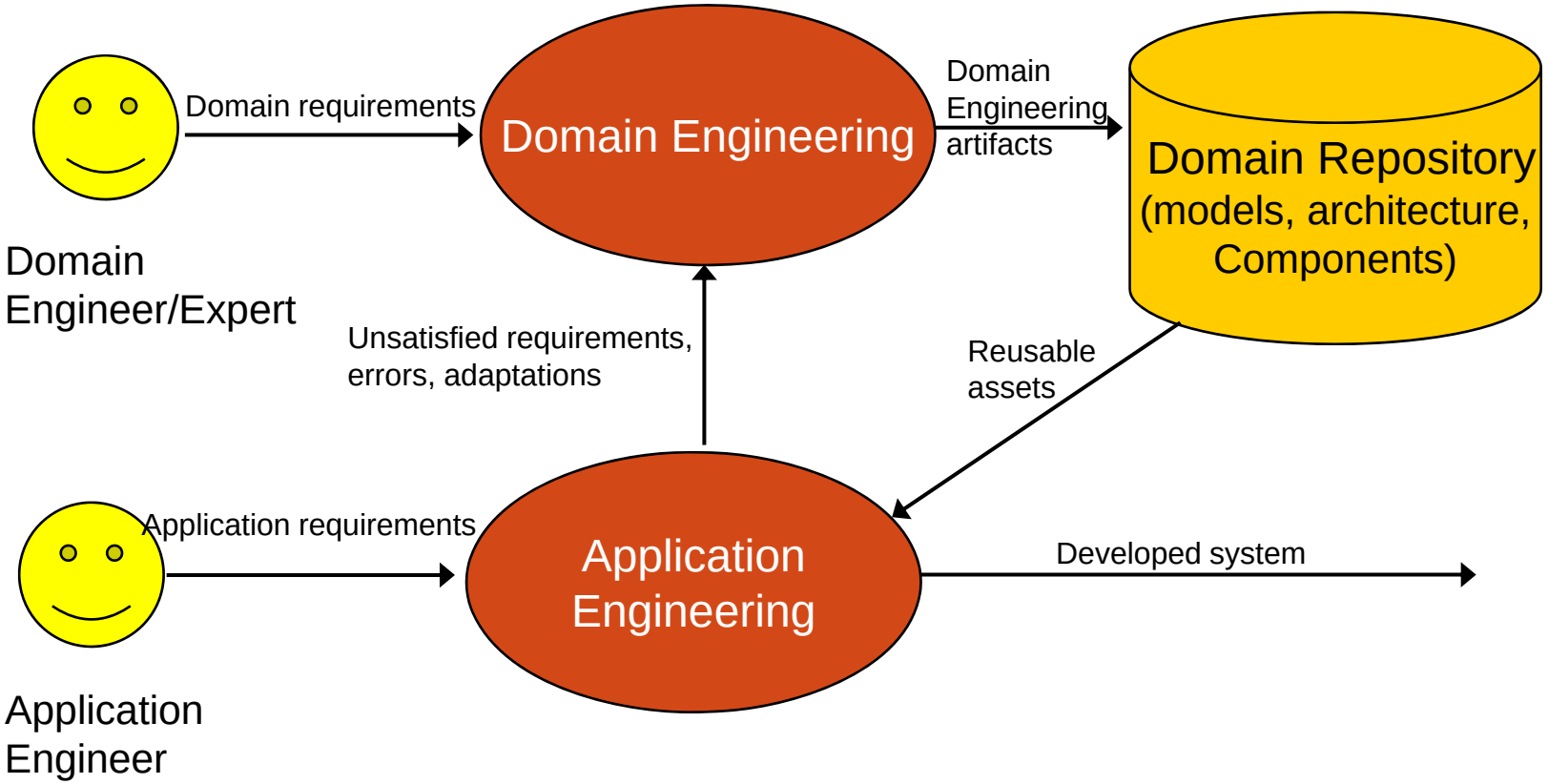
  (Carnegie-Mellon Software Engineering Institute)

Domain Engineering Concepts:

How does Domain Engineering differ from Application Engineering?

Domain Engineering	Application Engineering
a systematic approach to construct reusable assets in a given problem domain	uses the assets to build specialized software systems in the given domain
Define and scope domain	Do delta analysis and design relative to a domain model and architecture
Analyze examples, needs, trends	
Develop domain model and architecture	Use component systems as starting point
Structure commonality and variability	Find, specialize, and integrate components
Engineer reusable component systems, languages, and tools	Exploit variability mechanisms, languages, generators, ….

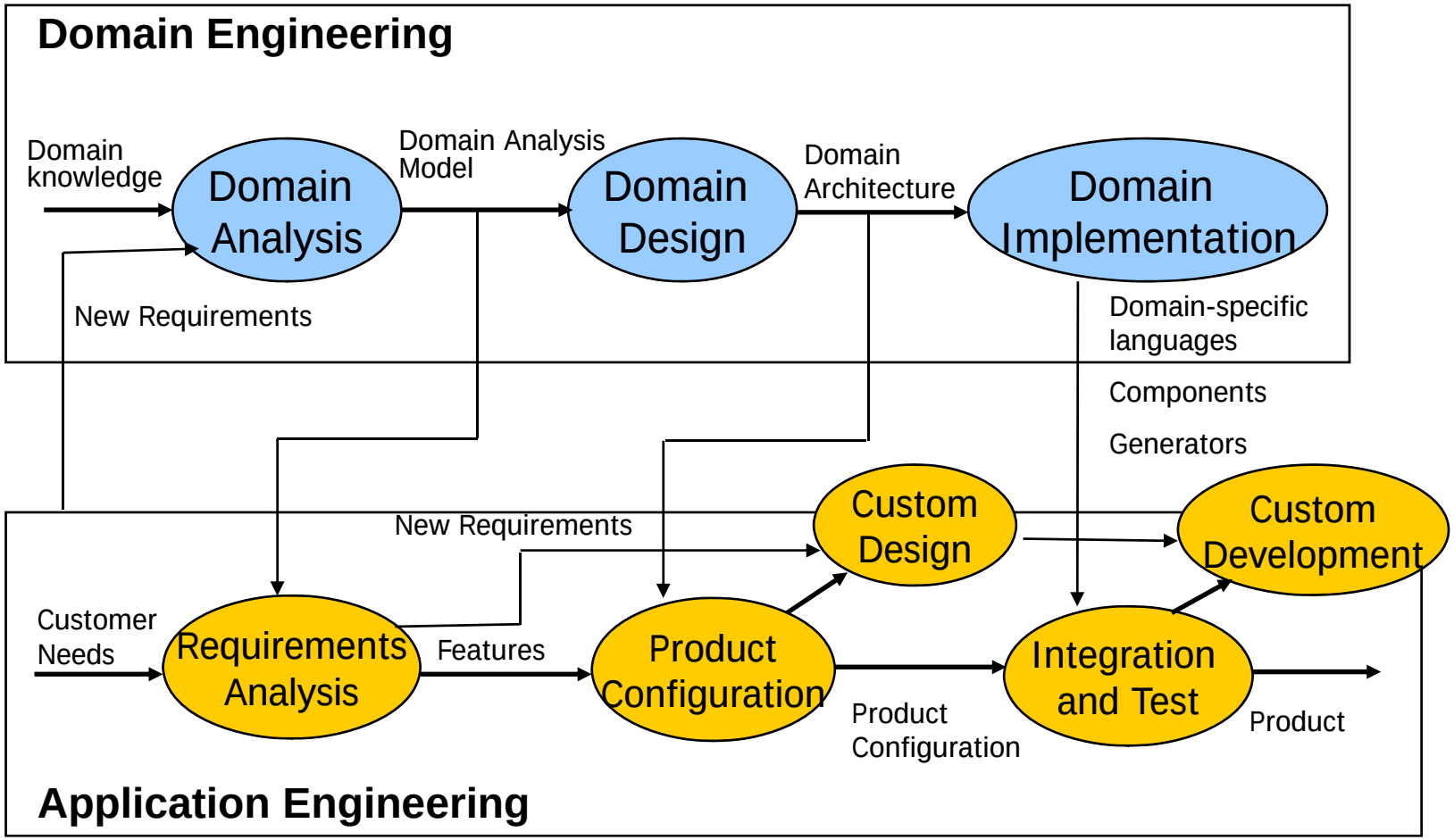
Domain Engineering Concepts:
How does Domain Engineering differ from Application Engineering?



Adapted from: Hassan Gomaa, Designing Software Product Lines with UML, Addison Wesley, 2004

Domain Engineering Concepts:

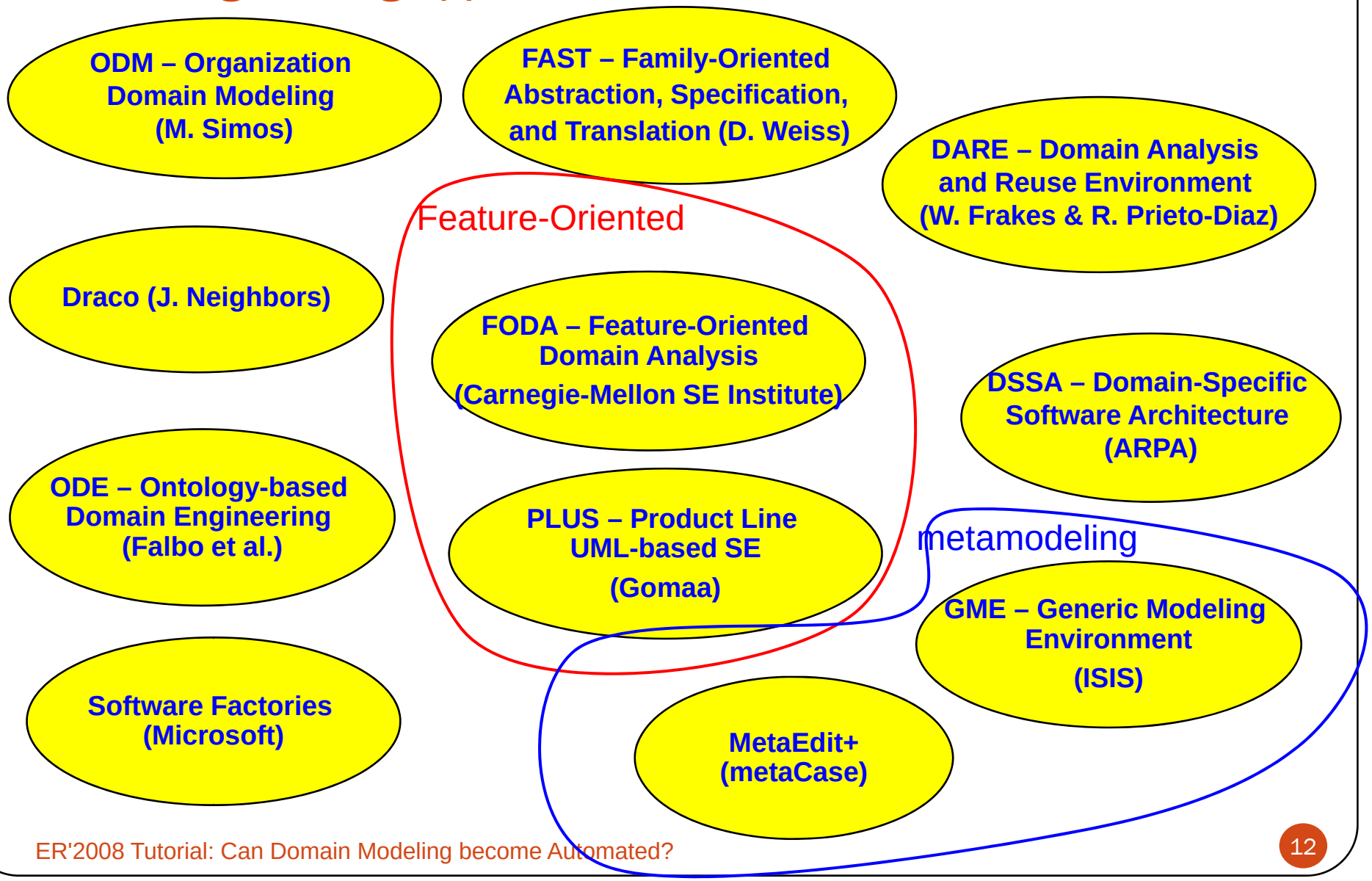
How does Domain Engineering differ from Application Engineering?



Adapted from: the Software Engineering Institute at Carnegie Mellon

Domain Engineering Concepts:

Domain Engineering Approaches – A Partial List



Domain Engineering Concepts:
What are the advantages of Domain Engineering?

- Provide means for gathering and organizing domain related information and **knowledge**.
- Provide libraries of assets to be **instantiated (reused)** in particular applications.
- Provide **validation** templates for particular applications in order to avoid semantic errors in early development stages.

Domain Engineering Concepts:
What are the limitations of Domain Engineering?

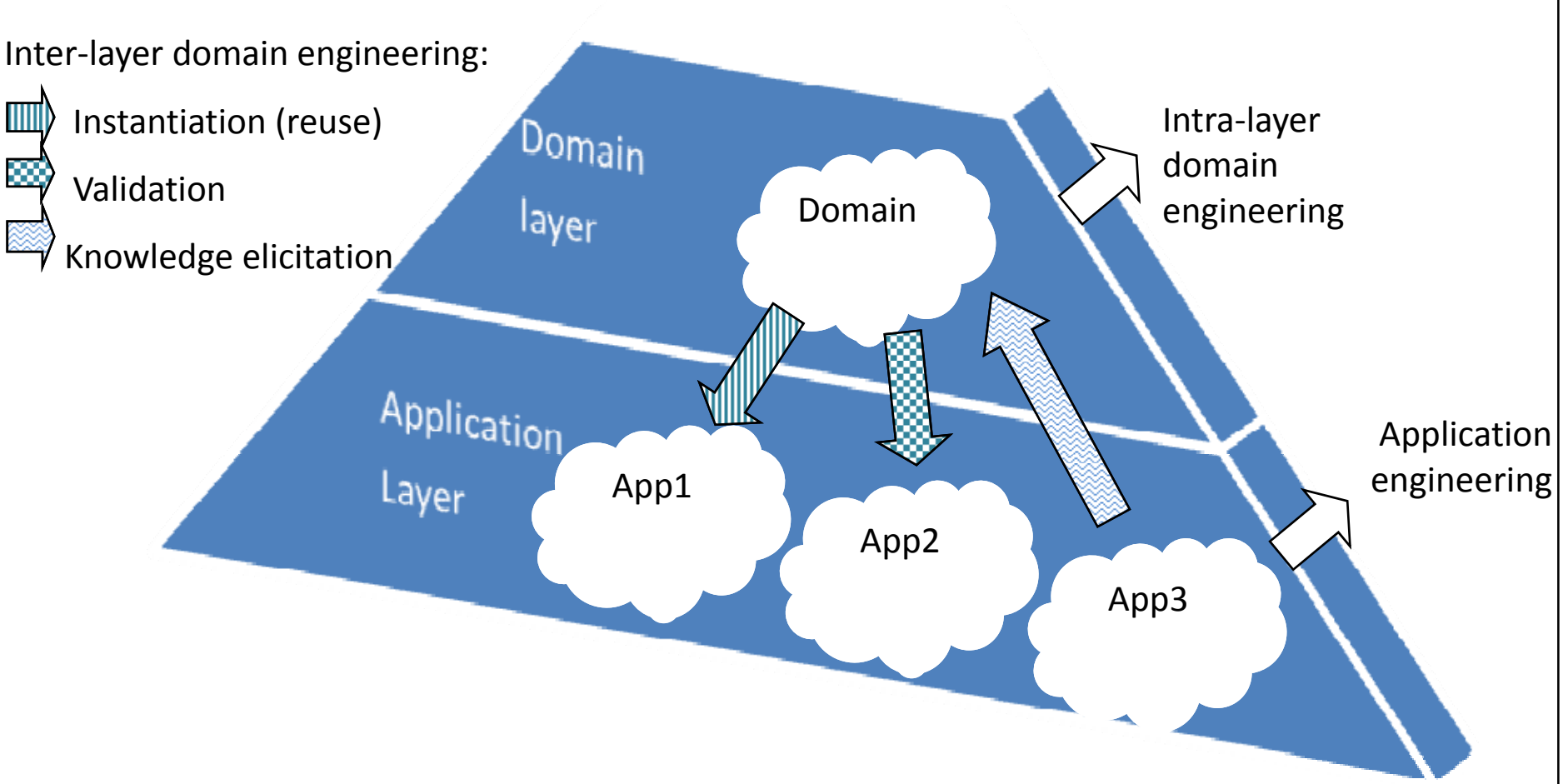
- Deal with too broad areas (domains) which are usually understood only during the development process.
- Require expertise in the domain, reaching a very high level of abstraction, and providing flexible, yet formal, artifacts.
- Deal with two different abstraction levels: the domain and the application.
  - Each level is accompanied by different notions and notations
  - The transition between the levels remains sometimes vague

Agenda

- **An overview of Domain Engineering**
  - Domain Engineering concepts
  - **The two layer model and its related activities**
- **Semi Automatic Creation of Domain Models**
  - The ADOM approach
  - Knowledge Elicitation in ADOM
  - The ADOM supporting tools
  - Experimenting with ADOM
- **Summary and Discussion**

The Two Layer Domain Engineering Model

Presenting the Model



The Two Layer Domain Engineering Model

Presenting the Model

- The ***domain layer*** is the layer in which the domain artifacts are handled.
- The ***application layer*** is the layer in which the applications (systems) are developed
- Examples:

Field	Domain layer	Application layer
Software Engineering	Domain knowledge	Programs, applications, or systems
Business Process Design	Reference models	Particular business processes
Method Engineering	Standards, such as ISO/IEC 24744 and SPEM	Different development methodologies (e.g., RUP, XP, and so on)

The Two Layer Domain Engineering Model
Presenting the Model

- Domain engineering supports:
  - ***Intra layer activities***, namely domain analysis, domain design, and domain implementation
  - ***Inter-layer activities***, i.e., forces that exist between the domain and application layers:
    - The domain layer artifacts may be instantiated (reused) in the application layer.
    - The domain layer artifacts may be used for validation purposes in the application layer.
    - The applications may be generalized into domain artifacts in a process of knowledge elicitation

Agenda

- **An overview of Domain Engineering**
  - Domain Engineering concepts
  - The two layer model and its related activities
- **Semi Automatic Creation of Domain Models**
  - **The ADOM approach**
  - Knowledge Elicitation in ADOM
  - The ADOM supporting tools
  - Experimenting with ADOM
- **Summary and Discussion**

The ADOM Approach
Motivation

- An **A**pplication-based **DO**main **M**odeling approach
- Application and domain models are **similar**:
  - They both define structure
  - They both exhibit functionality (behavior)
  - They both introduce structural and behavioral constraints
- However they also **differ** in their:
  - Abstraction level
  - Flexibility level

The ADOM Approach
Motivation

- ADOM treats domains with regular application engineering tools, methods, and techniques.
  - Furthermore, it enables its adopt to different fields due to its independence in the modeling language
- ADOM defines three layers of abstraction: application, domain, and language.
  - In all layers the same approaches (notions and notations) are applied.
  - Each layer defines constraints on the more concrete layers.

The ADOM Approach
Examples of three Common Usages

- **Software Engineering**
  - **Goal:** Guiding and validating system design using domain artifacts
  - **Language:** UML, mainly use case, class, sequence, and state diagrams
  - **Deliverables:** Application and domain models
- **Business Process Design**
  - **Goal:** Guiding and validating business process design using reference models
  - **Languages:** EPC, BPMN, and UML activity diagrams
  - **Deliverables:** Business process and reference models

The ADOM Approach

Examples of three Common Usages

- **Method Engineering**
  - **Goal:** Guiding the specification of method components and the creation of situational methodologies
  - **Language:** Mainly OPM
  - **Deliverables:** Method component type, method component, and situational methodology models
- **Requirements from the associated modeling languages:**
  - A multiplicity indicator mechanism
  - A domain classifier mechanism
  - UML's stereotype mechanism was employed for these purposes. Minor changes to the other languages metamodels were introduced

The ADOM Approach for SE

Multiplicity Indicators in ADOM-UML

Abbreviated Notation	Full Notation	Meaning
<<optional many>>	<<multiplicity min = 0 max = ∞>>	Any number (including 0) of application elements can be classified (stereotyped) as this domain element
<<optional single>>	<<multiplicity min = 0 max = 1>>	At most one application element can be classified (stereotyped) as this domain element
<<mandatory many>>	<<multiplicity min = 1 max = ∞>>	At least one application element can be classified (stereotyped) as this domain element
<<mandatory single>>	<<multiplicity min = 1 max = 1>>	Exactly one application element can be classified (stereotyped) as this domain element
	<<multiplicity min = n max = m>>	Between n to m application elements can be classified (stereotyped) as this domain element

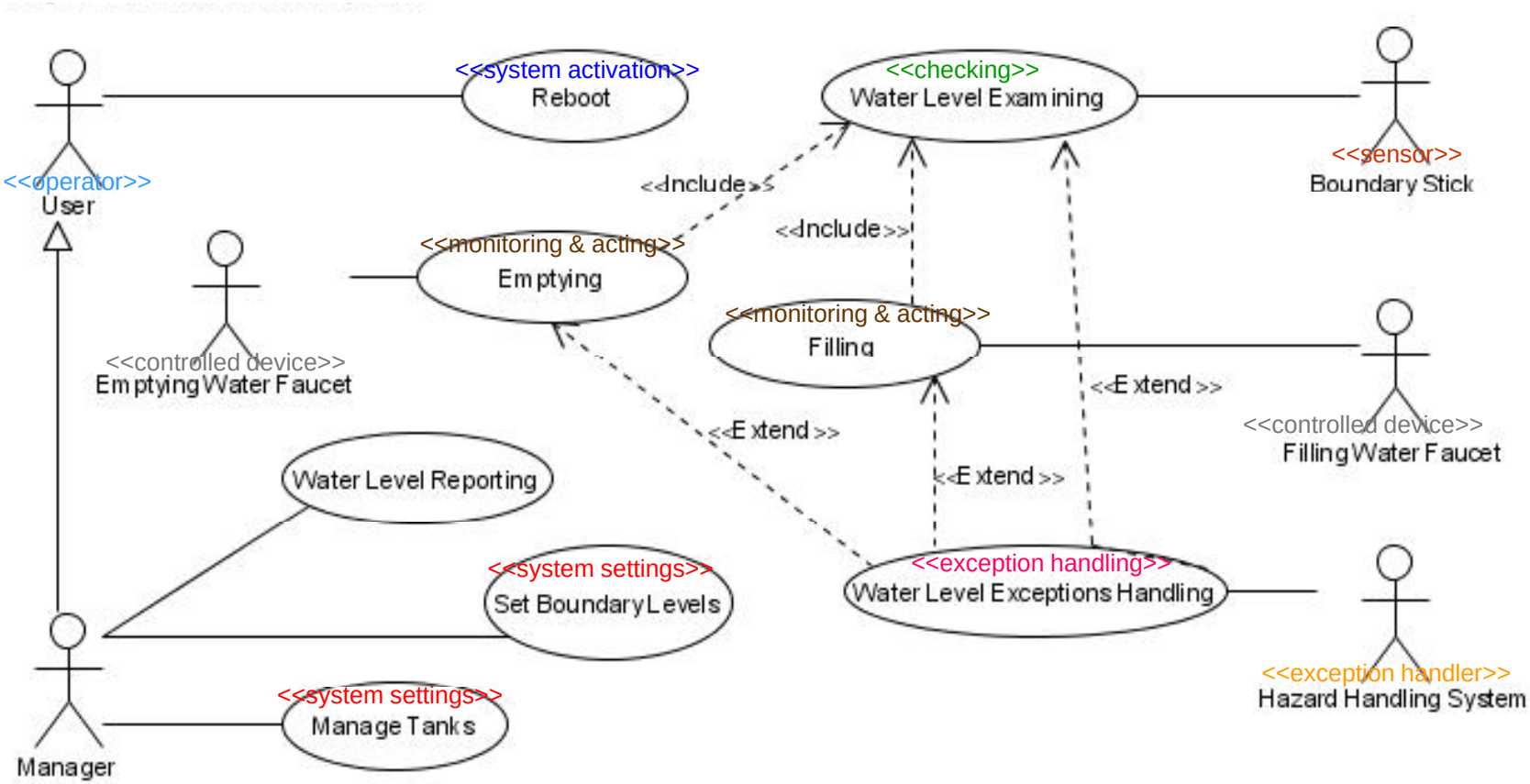
The ADOM Approach for SE

An Example of Process Control Systems

- Applications in the **Process Control System (PCS)** domain monitor and control the values of certain variables through a set of components that work together to achieve a common objective or purpose.
- Two specific examples in the domain:
  - A **water level control system (WLC)** for monitoring and controlling the water level in tanks
  - A **home climate control system (HCC)** for monitoring and controlling room temperature and humidity

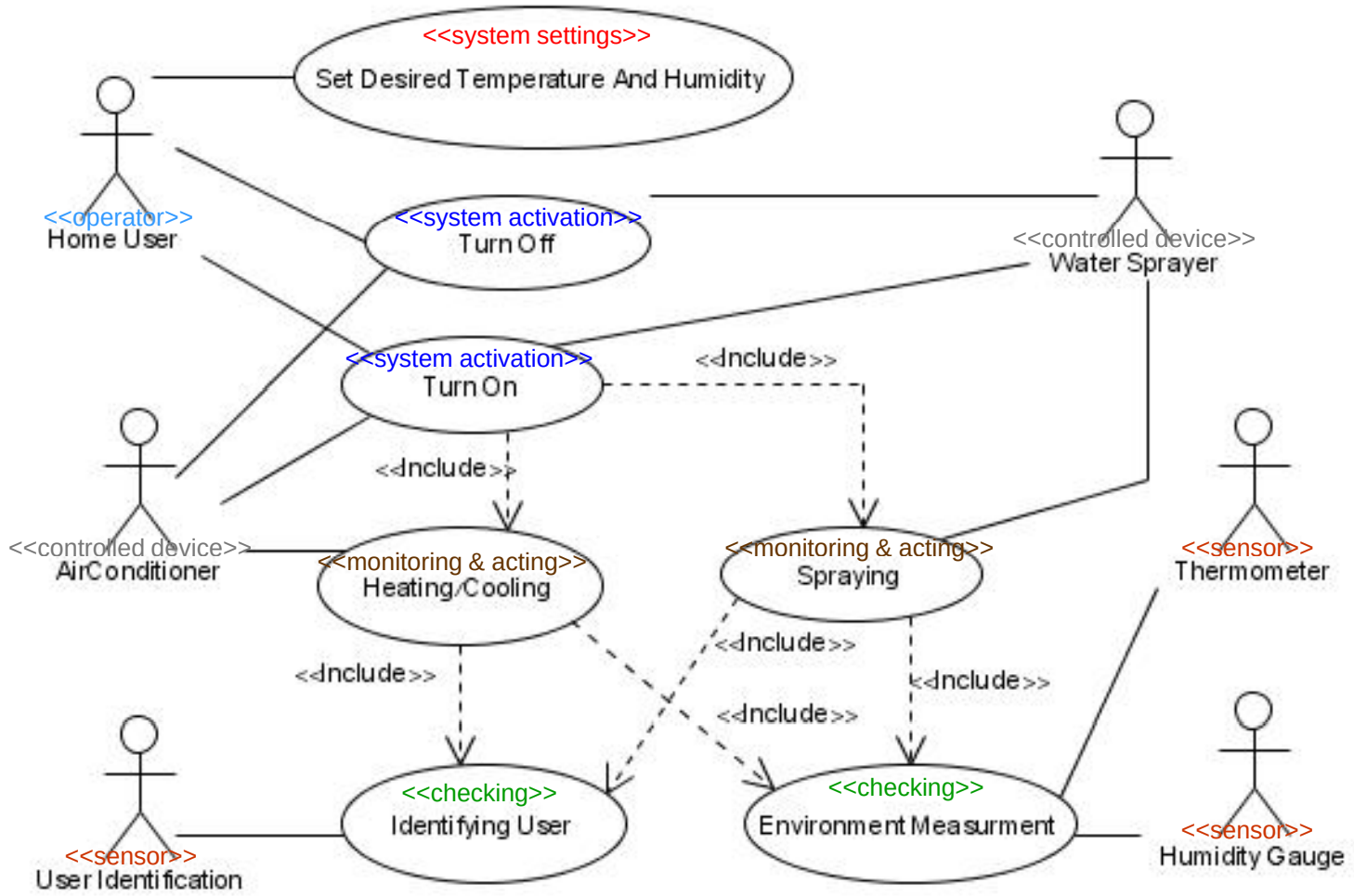
The ADOM Approach for SE

The WLC application model – a UC diagram



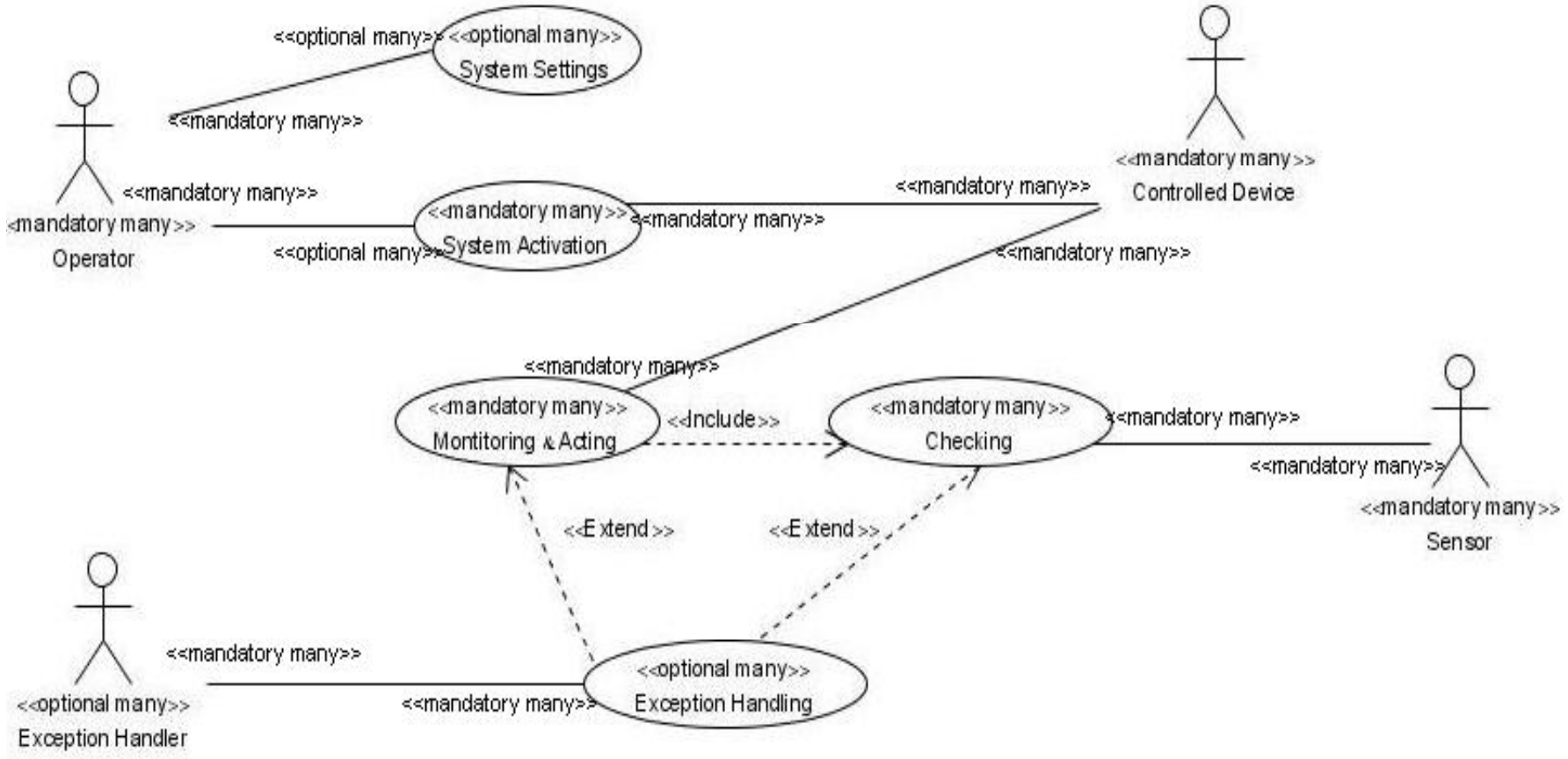
The ADOM Approach for SE

The HCC application model – a UC diagram



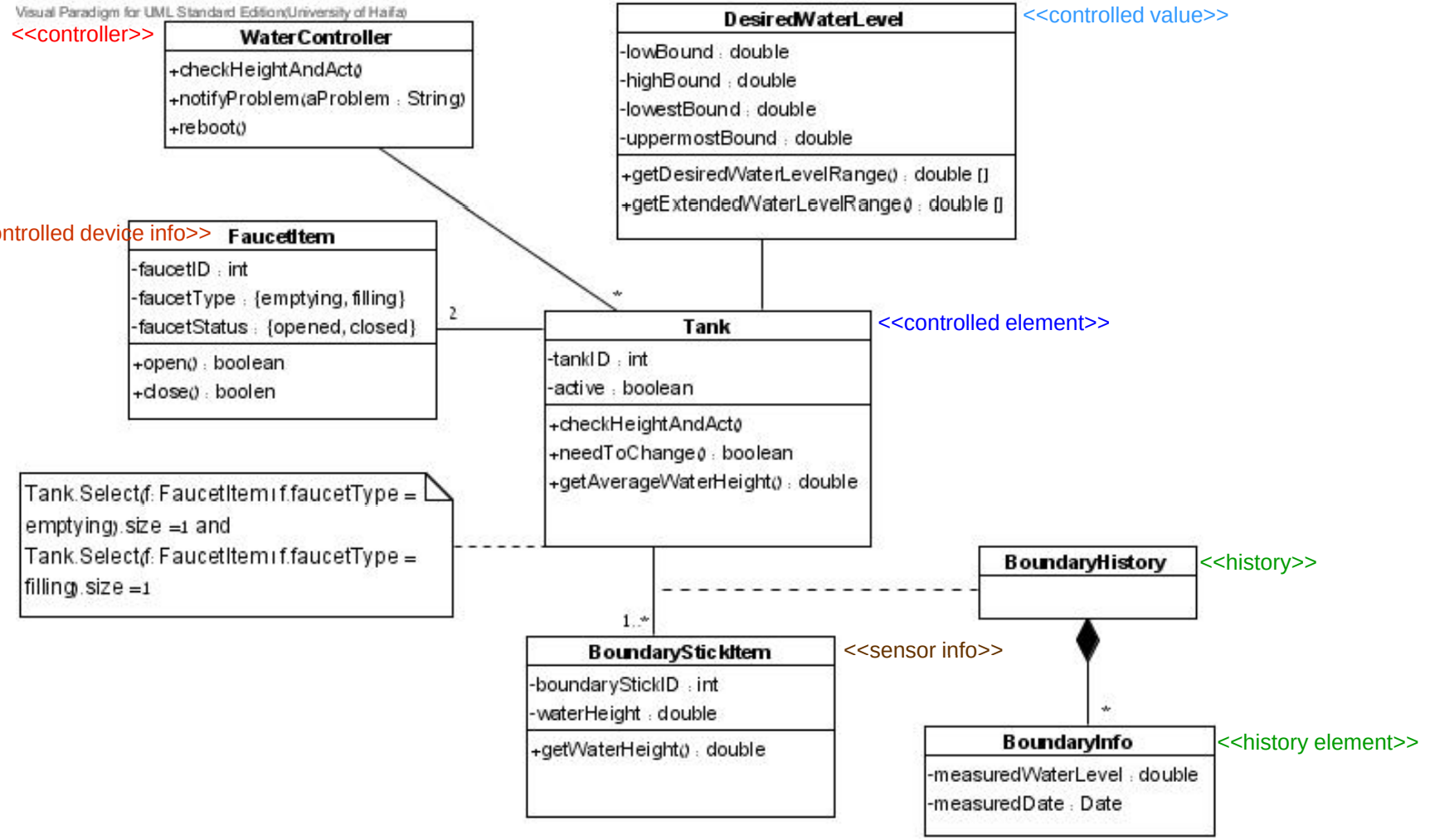
The ADOM Approach for SE

The PCS domain model – a UC diagram



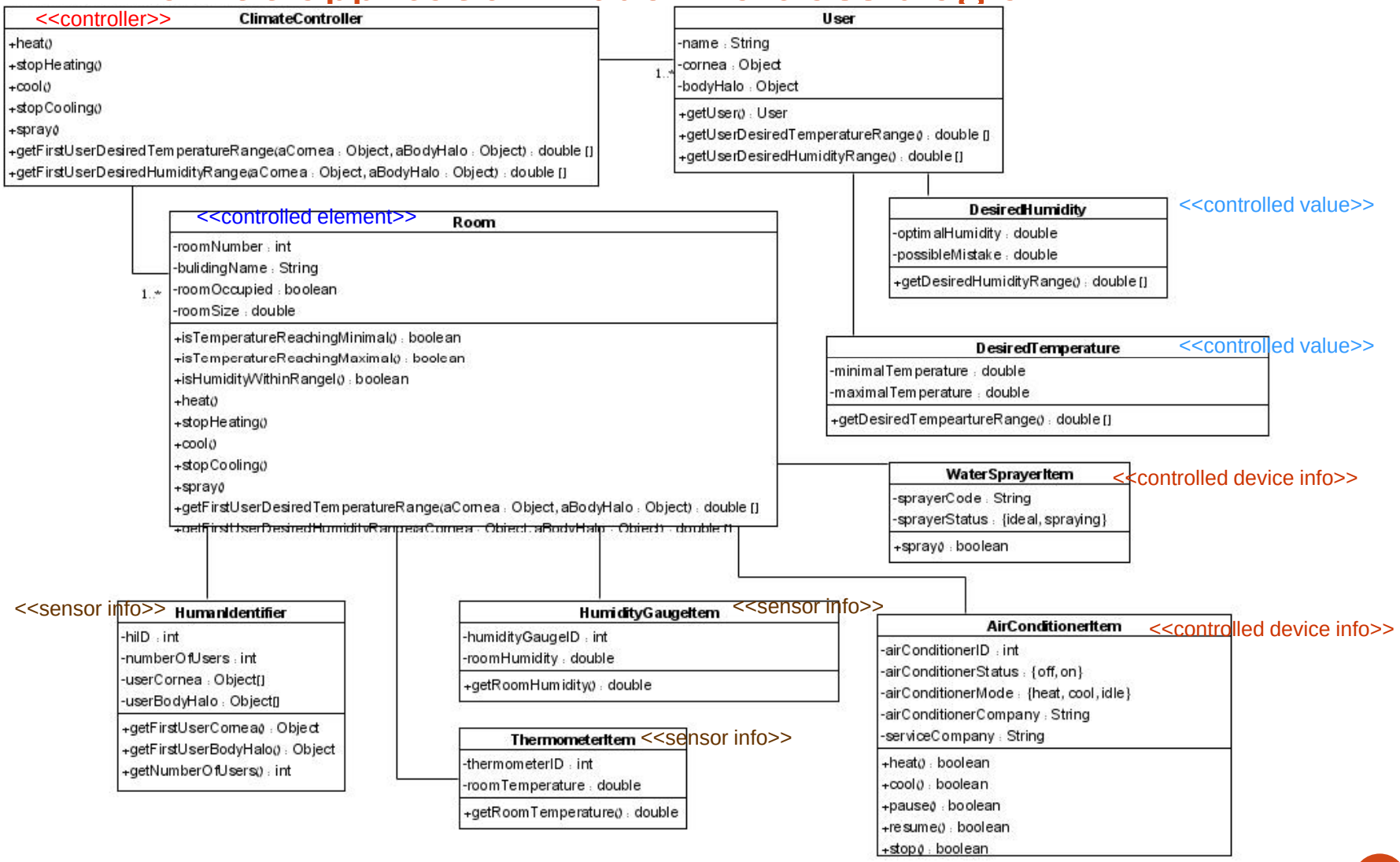
The ADOM Approach for SE

The WLC application model – a class diagram



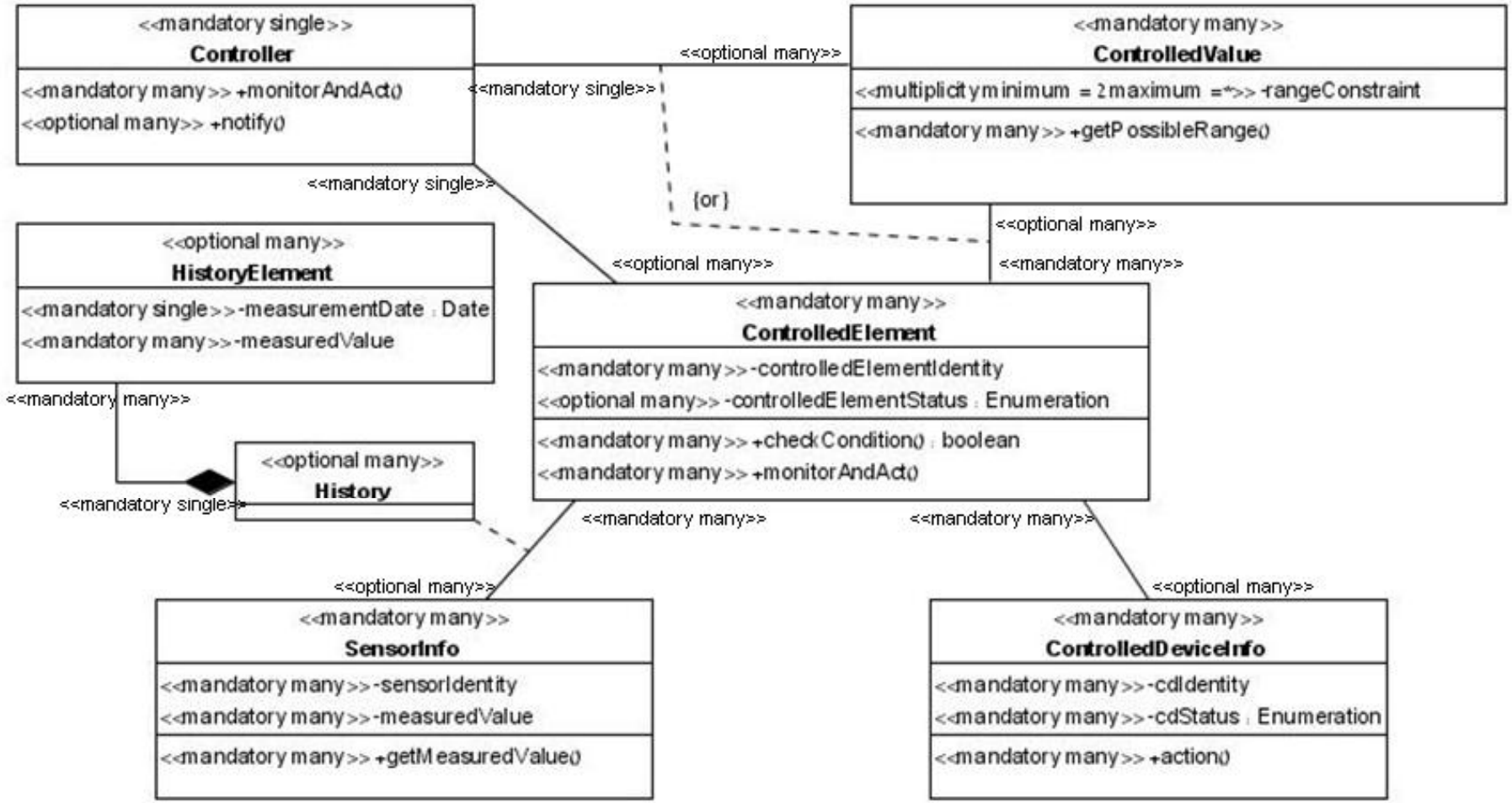
The ADOM Approach for SE

The HCC application model – a class diagram



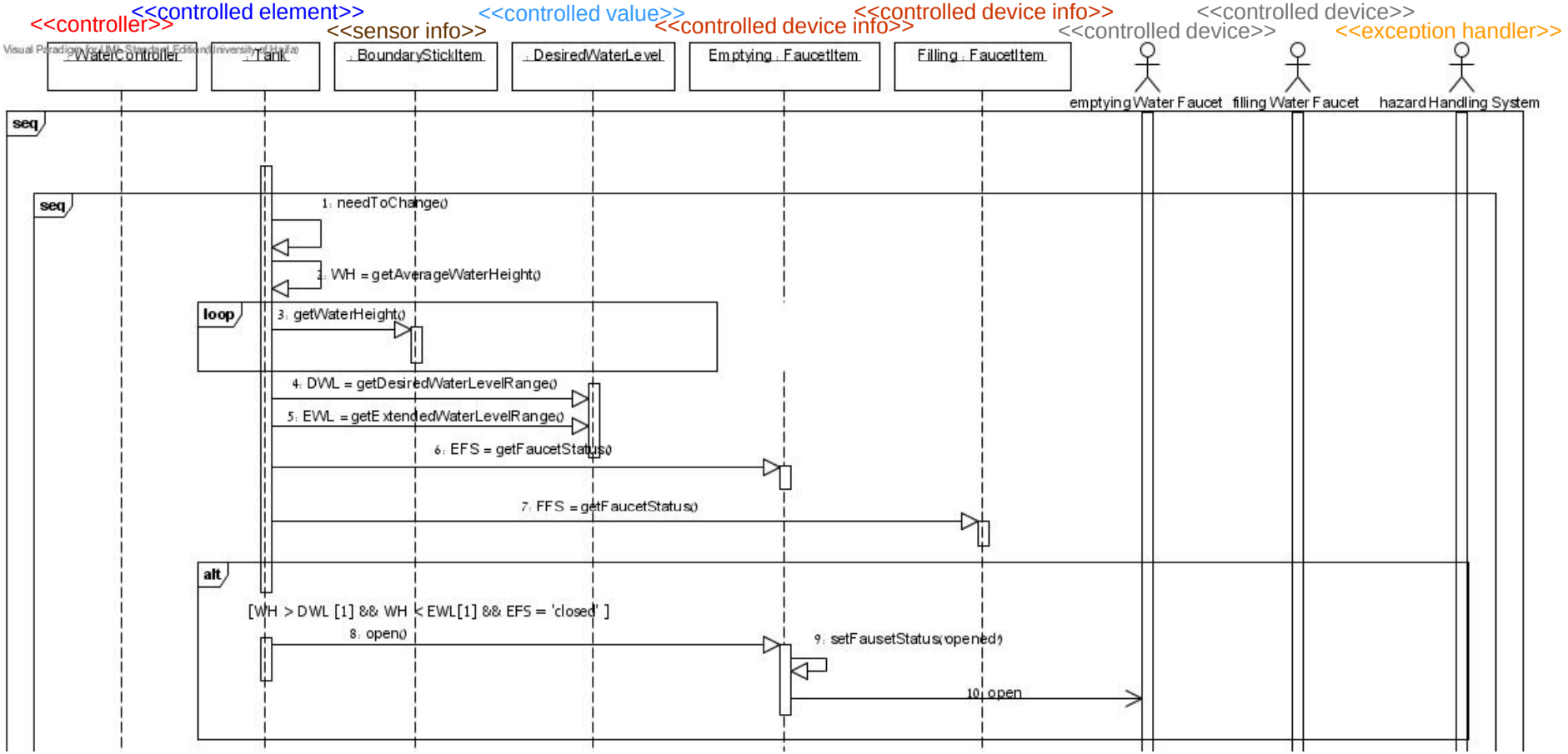
The ADOM Approach for SE

The PCS domain model – a class diagram



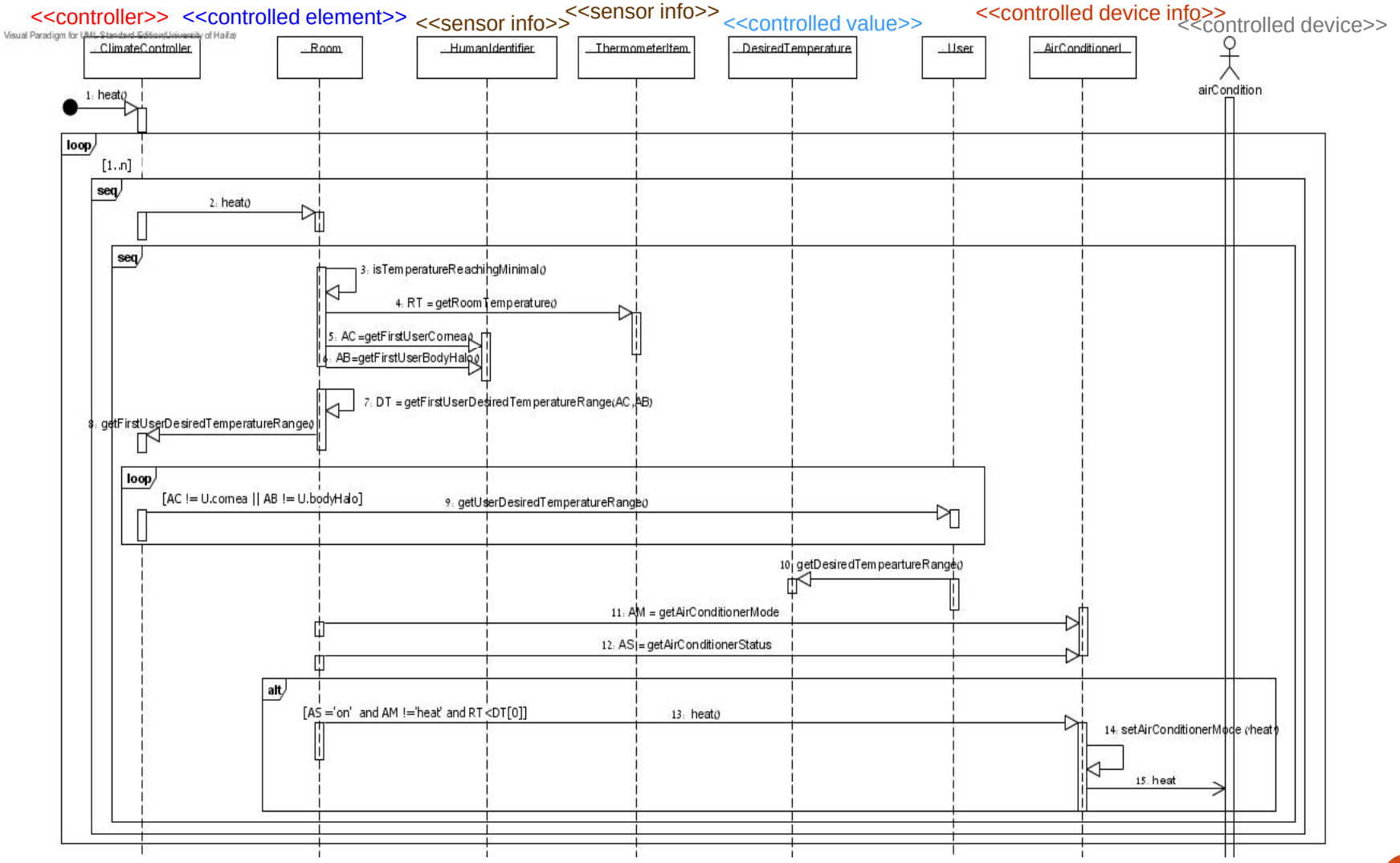
The ADOM Approach for SE

The WLC application model – a sequence diagram



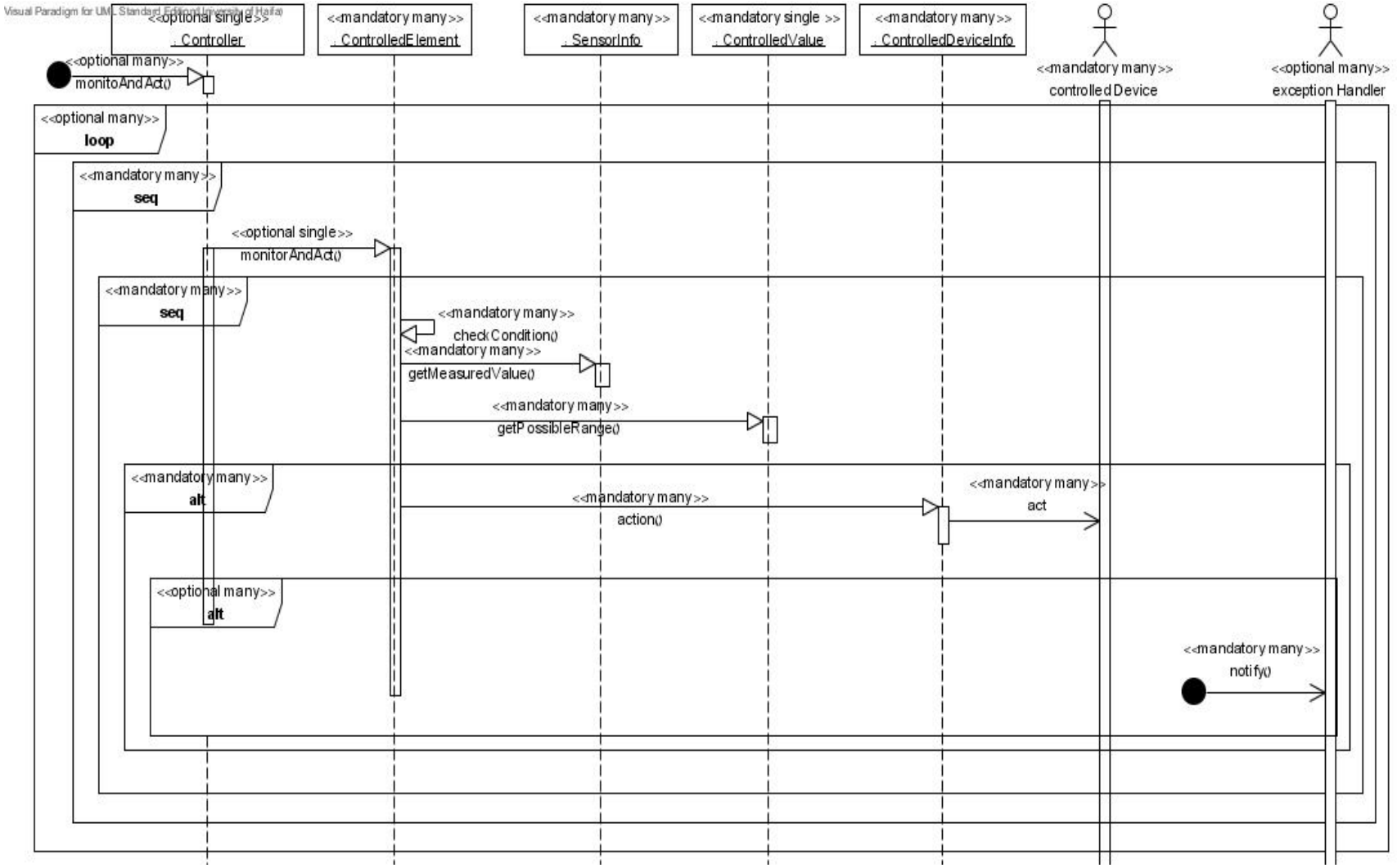
The ADOM Approach for SE

The HCC application model – a sequence diagram



The ADOM Approach for SE

The PCS domain model – a sequence diagram



Agenda

- **An overview of Domain Engineering**
  - Domain Engineering concepts
  - The two layer model and its related activities
- **Semi Automatic Creation of Domain Models**
  - The ADOM approach
  - **Knowledge Elicitation in ADOM**
  - The ADOM supporting tools
  - Experimenting with ADOM
- **Summary and Discussion**

Knowledge Elicitation in ADOM
Basic Terminology

- A ***model M*** is a representation of certain aspects of the world of interest.

  The basic building blocks of models are *elements*.

- A ***relational element re*** in a model M is a binary directional relationship between two other elements in M.

  Notation: $re_M = (s, t)$, where s is the *source* of re_M and t is its *destination*.

  - Bidirectional relationships are represented as two relational elements $re_1 = (s, t)$ and $re_2 = (t, s)$.
  - n-ary relationships are split into binary relationships.
  - Messages & Associations are examples of relational elements in UML.

Knowledge Elicitation in ADOM
Basic Terminology

- A ***dependent element*** d in a model M is an element which has an implicit binary directional relationship with another element e in a model M, such that the omission of e from the model implies the omission of d.

  Notation: d $\hookleftarrow_M$ e, where e is termed the ***dependee*** of d in model M.

  - Attributes & Operations are examples of dependent elements.

- A ***first order element*** in a model M is an element which is not relational neither dependent in model M.

  - Top-level packages are examples of first order elements in UML.

Knowledge Elicitation in ADOM
Element Similarity Types

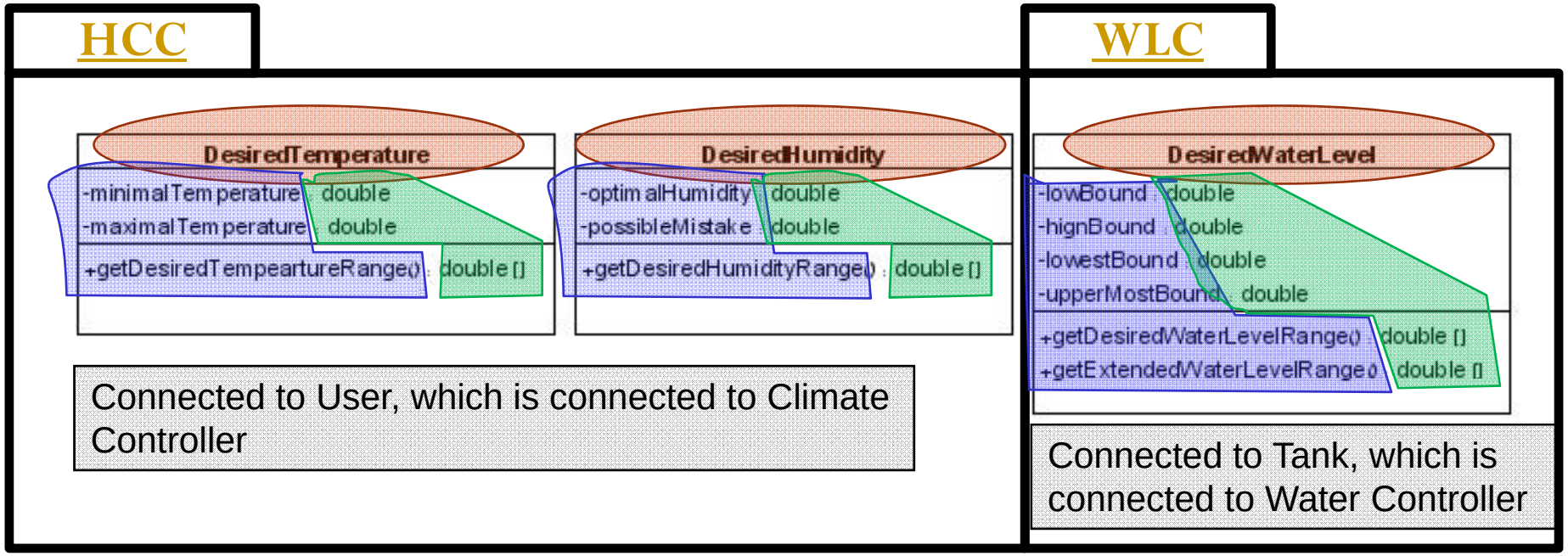
- ***Meta-information similarity***
  - Using meta-classes, types, and other kinds of meta-information in order to determine whether two elements are similar
- ***Linguistic (syntactic) similarity***
  - Adopting Dao and Simpson similarity measurement between two sentences
  - Using WordNet - a large, online lexical base for English language, freely distributed by the Princeton University
    - Domain-specific vs. general purpose lexical bases

Knowledge Elicitation in ADOM
Element Similarity Types

- *Contextual (semantic) similarity*
  - *Dependee similarity* takes into consideration the dependent elements of certain elements
    - The structure of compound elements is used in order to measure their similarity
    - In class diagrams, for example, we can assume, to some degree of confidence, that classes that exhibit similar attributes and operations are similar too.
  - *Relational similarity* takes into consideration the relationships in which two elements participate as sources or destinations
    - Relational similarity takes into consideration the type of the relationship
    - Elements that are connected to similar elements are more similar

Knowledge Elicitation in ADOM

Examples of Similarity Measurements



Linguistic (syntactic) similarity

Contextual (semantic) dependee similarity

Meta-information similarity

Contextual relational similarity

Knowledge Elicitation in ADOM
Similarity Measurements of Behavior: UML Sequence Diag.

- The approach refers similarity to both structural and behavioral aspects of the modeled systems
  - Models are considered hierarchical graphs with nodes (first order and dependent elements) and arcs (relational elements)
  - A sequence diagram, for example, is the dependee of its elements
  - Since sequence diagrams rely on the system structure as expressed in the class diagrams, matches established between elements in the class diagram are used for matching behavioral elements in the sequence diagram
  - In order to refer to the order of messages in a sequence diagram, we implicitly add relational elements $re=(m_1, m_2)$ between any two messages m_1 and m_2 such that m_1 precedes m_2

Knowledge Elicitation in ADOM

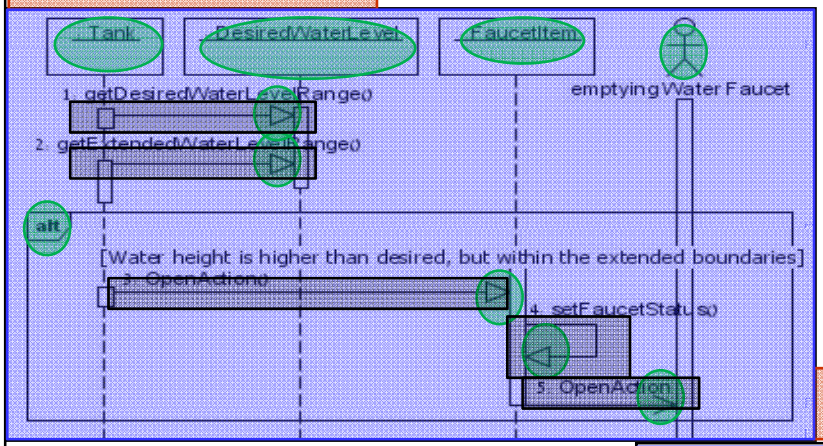
Similarity Measurements of Behavior: UML Sequence Diag.

- The similarity of two sequence diagrams, seq_1 and seq_2, is measured as follows:
  - **Linguistic similarity** takes into consideration the sequence diagrams names
  - **Dependee similarity** refers to the similarities of the different combined fragments, messages, and lifelines that constitute the sequence diagram.
  - **Relational similarity** refers to the message sources, destinations, and order.
  - **Meta-information similarity** takes into consideration the element types (e.g., the message types).

Knowledge Elicitation in ADOM

Similarity Measurements of Behavior: UML Sequence Diag.

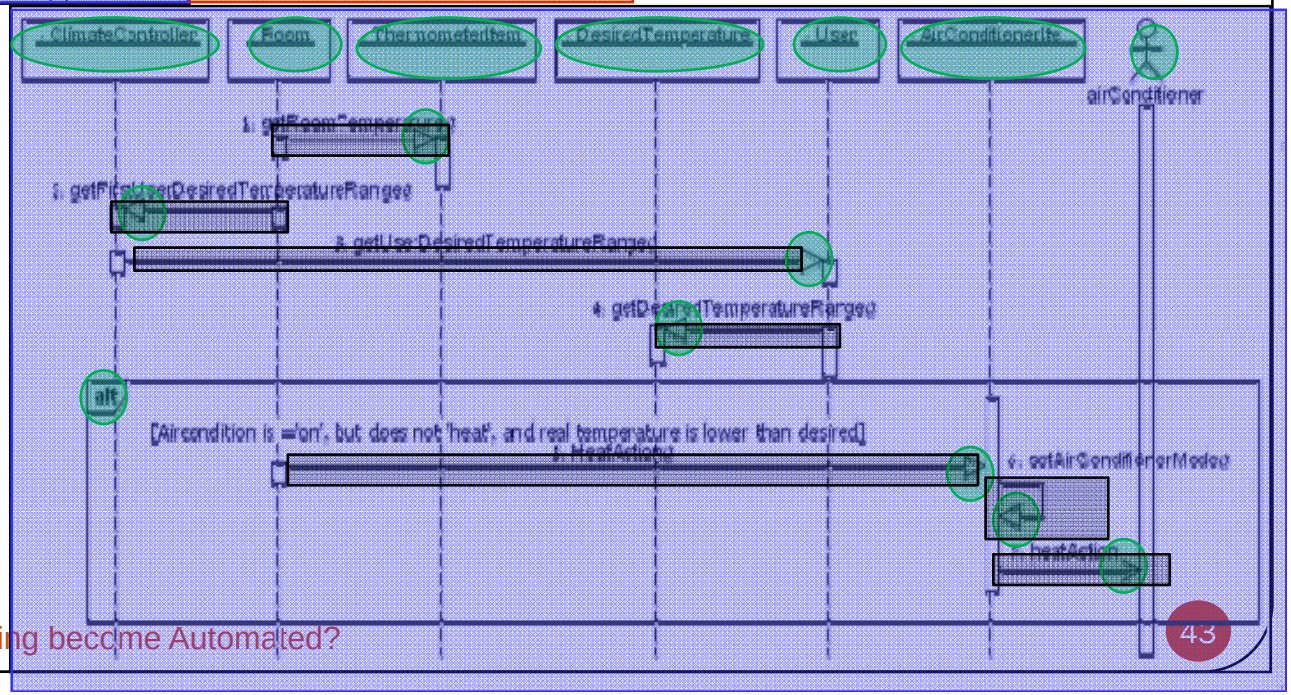
WLC: Tank Fill



Linguistic (syntactic) similarity

Contextual (semantic) dependee similarity

HCC: Room Heat



Meta-information similarity

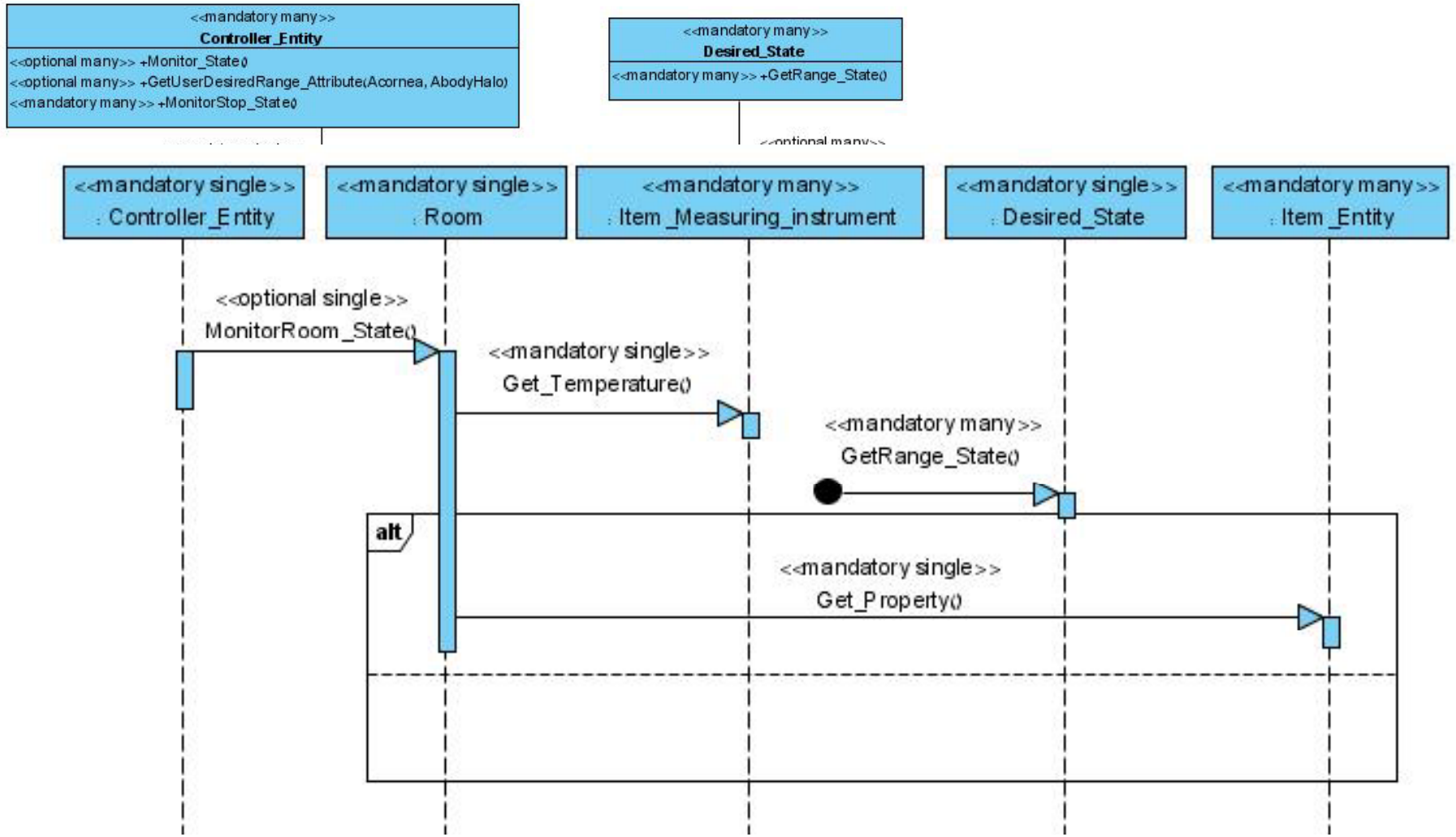
Contextual (conceptual) relational similarity

Knowledge Elicitation in ADOM
Merging and Generalizing

- Thresholds are defined in order to consider only similar enough pairs of elements for the model merging phase
- Similarity groups are defined
  - *A **similarity group*** of an element e, SG(e), is defined as a group of elements whose general similarity with e passes a certain similarity threshold
    - Note that $e \in SG(e') \Leftrightarrow e' \in SG(e)$
- All elements in the same similarity group are generalized into one domain element. WordNet is used for giving names to the domain elements. We distinguish between:
  - Mandatory elements that appear in all applications in the domain
  - Optional elements that appear in several applications in the domain
  - Application-specific elements that should not be part of the domain

Knowledge Elicitation in ADOM

The PCS Example



Agenda

- **An overview of Domain Engineering**
  - Domain Engineering concepts
  - The two layer model and its related activities
- **Semi Automatic Creation of Domain Models**
  - The ADOM approach
  - Knowledge Elicitation in ADOM
  - **The ADOM supporting tools**
  - Experimenting with ADOM
- **Summary and Discussion**

The ADOM Supporting Tools

- **Plugs-in to different Modeling languages:**
  - TOPCASED – UML CASE tool
  - OPCAT – OPM CASE tool
- The plugs-in support:
  - Domain model creation
  - Domain model instantiation guidance
  - Application model validation
- **A semi-automatic domain modeling (SDM) tool:**
  - Gets a set of UML models in XMI format and creates a domain model (in XMI format)
  - Currently supports class and sequence diagrams only
  - More on the tool can be found at http://mis.hevra.haifa.ac.il/~iris/research/SDM/index.htm

Agenda

- **An overview of Domain Engineering**
  - Domain Engineering concepts
  - The two layer model and its related activities
- **Semi Automatic Creation of Domain Models**
  - The ADOM approach
  - Knowledge Elicitation in ADOM
  - The ADOM supporting tools
  - **Experimenting with ADOM**
- **Summary and Discussion**

Experimenting with ADOM

Experiment #1: SDM vs. Manual Domain Modeling

- **Goal:** checking model comprehension, correctness, and completeness when using SDM
- **Subjects:** third year undergraduate students (HU) who had previous knowledge or experience in system modeling and specification and took an advance software engineering course
- **Forms:** the forms included three models of a project management domain
  - SDM domain model generated on 4 applications
  - Human made domain model created after carrying out literature review
  - Human made domain model created after carrying out reverse engineering of systems in the domain

Experimenting with ADOM

Experiment #1: SDM vs. Manual Domain Modeling

- **Tasks:**
  - Mapping the SDM domain model terminology to concepts in the two human made domain models
  - Grading 9 different aspects in the SDM domain model
    - The aspects referred to correctness, completeness, redundancy, and consistency of the SDM model
  - Correcting the SDM domain model to reflect their knowledge in the domain

Experimenting with ADOM

Experiment #1: SDM vs. Manual Domain Modeling

- **Results:**
  - Above 98% of the participants mapped the structure correctly
  - Above 70% of the participants succeeded in correctly mapping the behavior
  - The main criticism:
    - Too abstract names or names that are not compatible with their expected roles in the domain

To Raw Data …

Experimenting with ADOM
Experiment #1: SDM vs. Manual Domain Modeling

- **Results (cont.):**
  - The participants graded the SDM model as follows (out of 5):
    - Correctness – 3.9
    - Completeness – 3.7
    - Redundancy – 4.5 (i.e., the participants thought that the model did not include too many redundant elements)
    - Consistency – 4.0
    - Abstraction level – 3.5

To Raw Data …

Experimenting with ADOM
Experiment #2: SDM vs. Ontologies

- **Goal:** checking SDM model comprehension, correctness, and completeness in comparison to those of ontologies
- **Subjects:** third year undergraduate students (BGU) who had previous knowledge or experience in system modeling and specification and took an advance software engineering course
- **Forms:** the forms referred to a scheduling domain and included:
  - SDM domain model generated on 13 applications
  - A known ontology in the domain (Ozone)

Experimenting with ADOM
Experiment #2: SDM vs. Ontologies

- **Tasks:**
  - For each element in the SDM model, determining whether it is relevant to domain and how, and giving it a more meaningful name.
  - Mapping the SDM model terminology to that of the ontology and vice versa
  - Grading 7 different aspects in the SDM domain model
    - The aspects referred to correctness, completeness, and redundancy of the SDM model
    - The aspects referred only to structure and not to behavior

Experimenting with ADOM
Experiment #2: SDM vs. Ontologies

- **Results:**
  - 75% of the SDM domain elements were evaluated as relevant to the domain
  - 73% of the participant answers correctly mapped the SDM model terminology to that of the ontology
  - 65% of the participant answers correctly mapped the ontology terminology to that of the SDM model

To Raw Data . . .

Experimenting with ADOM
Experiment #2: SDM vs. Ontologies

- **Results (cont.):**
  - The participants graded the SDM model as follows (out of 5):
    - Correctness – 3.9
    - Completeness – 3.7
    - Redundancy – 3.9
    - Abstraction level – 2.3

To Raw Data …

Agenda

- **An overview of Domain Engineering**
  - Domain Engineering concepts
  - The two layer model and its related activities
- **Semi Automatic Creation of Domain Models**
  - The ADOM approach
  - Knowledge Elicitation in ADOM
  - The ADOM supporting tools
  - Experimenting with ADOM
- **Summary and Discussion**

Summary & Discussion
The ADOM Approach Perspectives

- **The knowledge engineering perspective**
  - ADOM provides a means for knowledge representation and elicitation
- **The instantiation perspective**
  - ADOM supports different types of reuse (customization, specialization, open and closed reuse) by incorporating domain elements into application models
- **The validation and verification perspective**
  - ADOM provides means for relaxing application artifacts to domain artifacts in order to check the validity of the application artifacts in the context of the given domain.

Summary & Discussion
ADOM's Advantages

- Treating domains similarly to applications enables the specification of behavioral constraints and not just structural ones.
- Treating domains in a separate layer (and not in the language layer) enables adjustment of ADOM to different modeling languages.
- The usage of the same modeling language for both the application and domain layers can reduce the ontological gap and the communication problems between the different stakeholders in the system development process.

Summary & Discussion
ADOM's Advantages

- ADOM supports semi-automatic domain knowledge elicitation
  - This process is supported by a tool
  - This process outputs reasonable models even on small numbers of applications
- ADOM supports other inter-layer domain engineering activities:
  - Validating application models throughout gradual system development stages for reducing the development efforts and costs as errors are detected in early stages
  - Guiding the instantiation of application models from domain models

Summary & Discussion
ADOM's Limitations & Future Work

- The completeness and expressiveness of the constraints in ADOM have not been checked (yet)
  - In particular, classifying different types of variability and supporting them in ADOM
- As many domain engineering techniques and methods, the ADOM approach can be criticized as dealing with too broad areas (domains)
  - The results of SDM depend on the broadness of the domain
  - The results of SDM depend on the chosen applications and their models
  - The results of SDM depend on the lexicon used (WordNet is too general purpose)

So… Can Domain Modeling become Automated?

Answers can also be sent to: iris@mis.haifa.ac.il

Thank you for your attention.

Questions?

The ADOM Approach for BP
Event-driven Process Chains & Multiplicity Indicators

- EPC was originally designed for business process modeling
- There are attempts to extend EPC to reference models as well
  - The most notably is Configurable EPC (C-EPC)
- EPC has no multiplicity indicators, neither domain classifiers
  - We added multiplicity indicators of the form <m, n> to the domain layer and domain classifiers of the form <*domain-element name*> to the application layer
  - <0, 0> is a special multiplicity indicator which can be associated to connectors and denotes a design-time decision

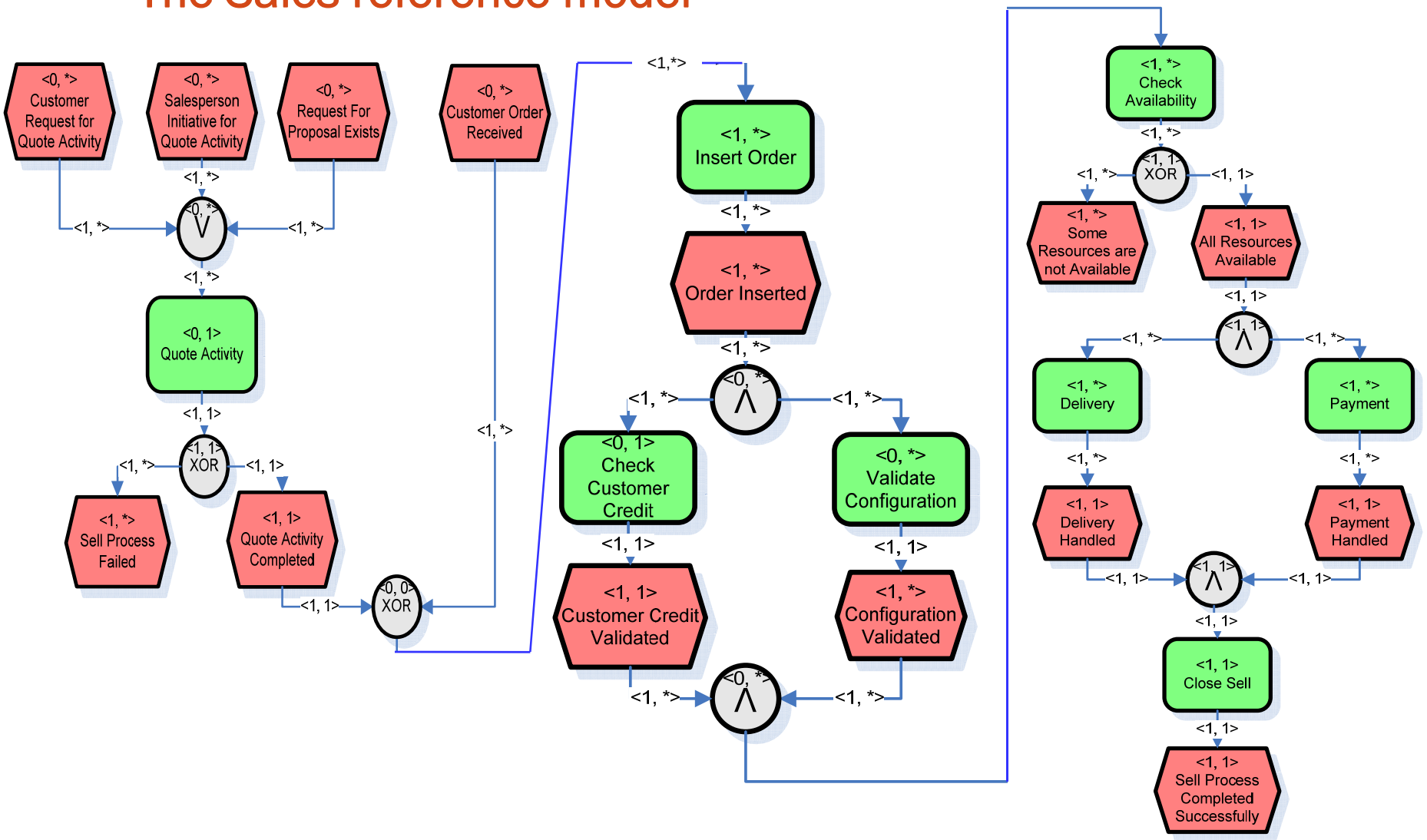
The ADOM Approach for BP

An Example of Sales processes

- The sales reference model integrates knowledge about selling products
- Deferent topologies for selling/producing are known:
  - make-to-stock: production is made for stock and not for a specific order. E.g., a chocolate manufacturer
  - assemble-to-order: off-the-shelf parts are composed for a specific order. E.g., a computer store
  - engineer-to-order: production is made for a specific order. E.g., a software development company
  - …

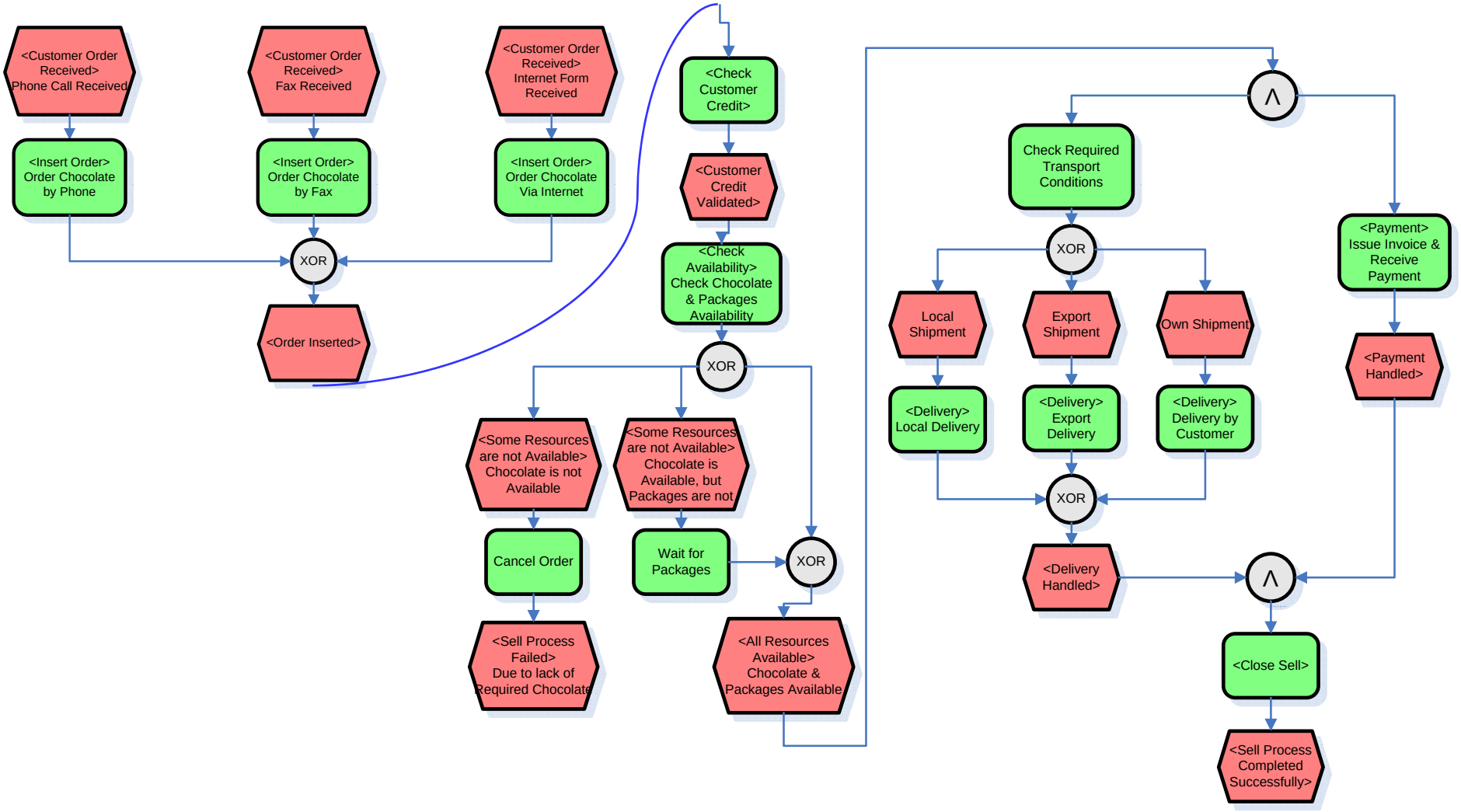
The ADOM Approach for BP

The Sales reference model



The ADOM Approach for BP

The Sales process of a Chocolate Manufacturer



The ADOM Approach for ME
ISO/IEC 24744

- **ISO/IEC 24744** (Feb. 2007) is an international standard regarding software engineering metamodel for development methodologies
- ISO/IEC 24744 refers to five aspects:
  - ***Work Units*** – the process aspects of methodologies
  - ***Work Products*** – the product aspects of methodologies
  - ***Producers*** – the people aspect of methodologies
  - ***Stages*** – the temporal aspect of methodologies
  - ***Model Units*** – the modeling language aspect of methodologies

The ADOM Approach for ME
Object-Process Methodology (OPM)

- OPM is an integrated approach to the development of systems in general and software systems in particular
- OPM unifies the system's *structure* and *behavior* throughout the analysis, design and implementation of the system within one frame of reference
  - Objects and processes are two types of equally important things (entities) required to describe a system in a single, unifying model
  - At any point in time, each object is at some state. Object states are transformed through the occurrence of a process
  - Object and Processes are connected via structural and procedural links
  - Complexity is controlled through recursive and selective scaling (zooming) of objects and/or processes to any desired level of details

The ADOM Approach for ME
OPM and ISO/IEC 24744

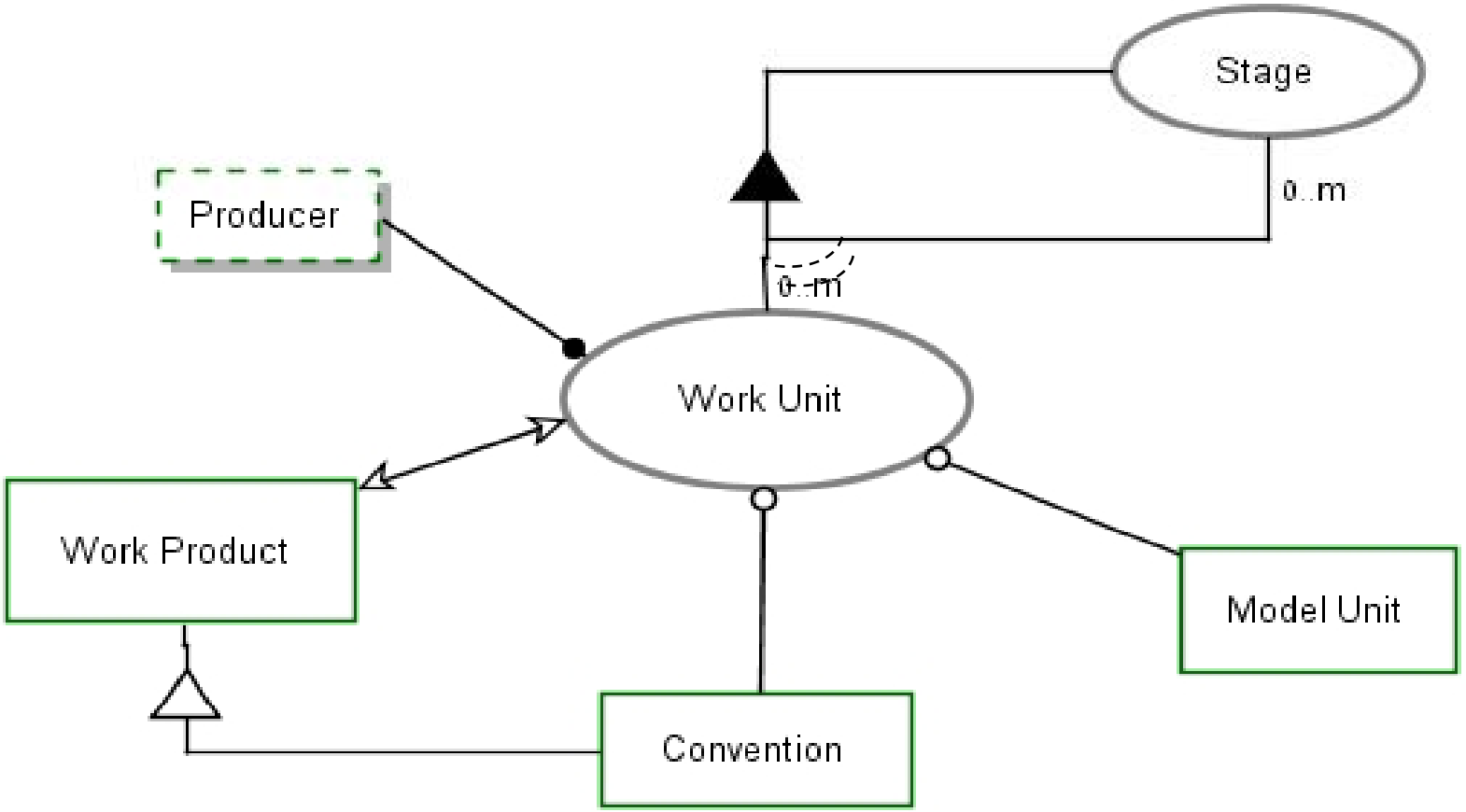
- A ***work unit*** is represented by an OPM process.
- A ***work product*** is represented by an OPM object which may be an input and/or an output for a work unit.
- A ***producer*** is a physical and environmental object that represents a human (or a team of humans) who should be involved somehow in the work unit.
- A ***stage*** is a wrapping process which may comprise of one or more work units.
- ***Model units*** are objects which are required for carrying out work units and producing work products.

The ADOM Approach for ME
OPM and ADOM

- OPM has multiplicities only to structural links
  - Domain model roles are used for specifying multiplicity indicators of objects, processes, and states
  - Link's cardinalities/multiplicities are used for specifying multiplicity indicators of structural links
  - The default multiplicity indicators of entities (objects, processes, and states) are assumed to be 'optional many'
  - The default multiplicity indicators of links are assumed to be 'mandatory single'
- OPM has roles as domain classifiers
  - A special kind of role (in the domain layer) is meta-attribute

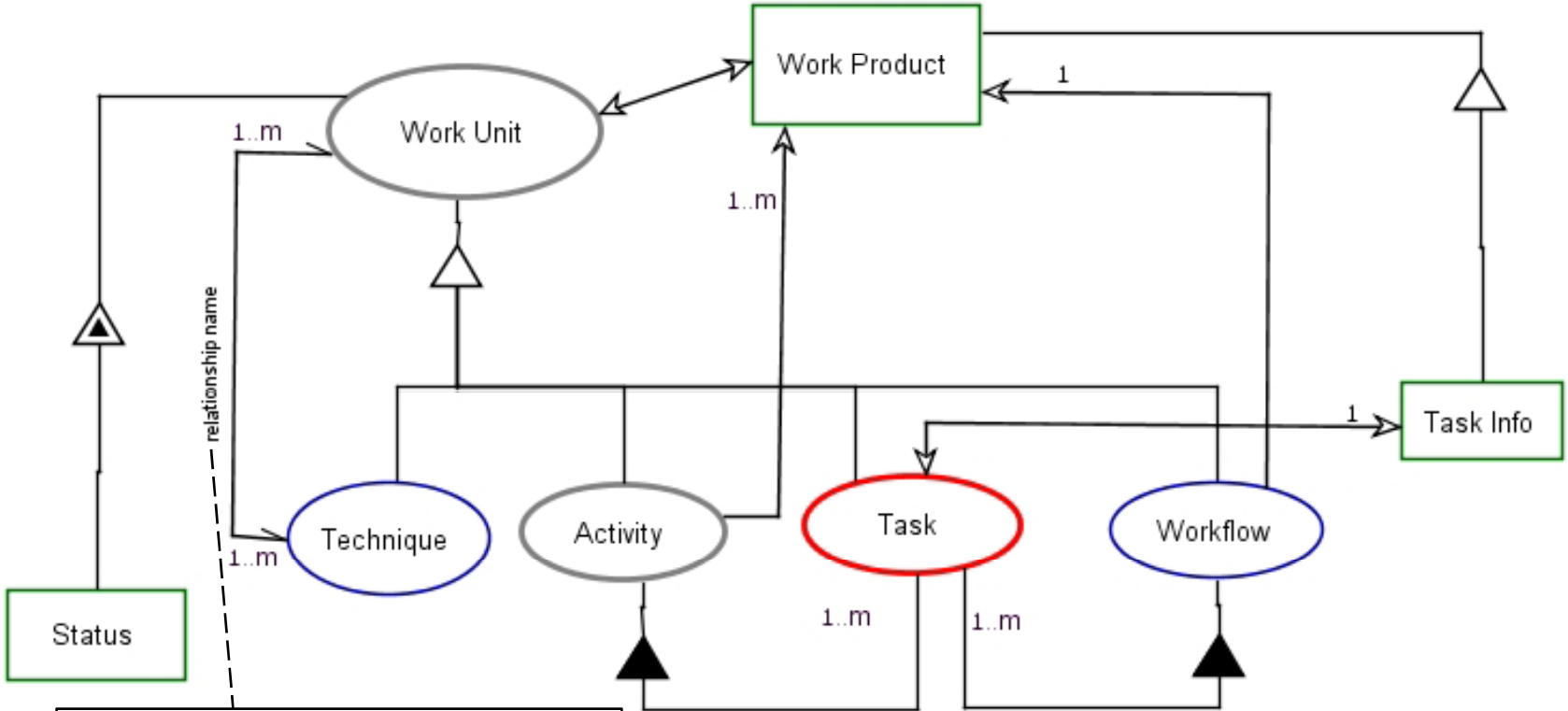
The ADOM Approach for ME

The top level domain model



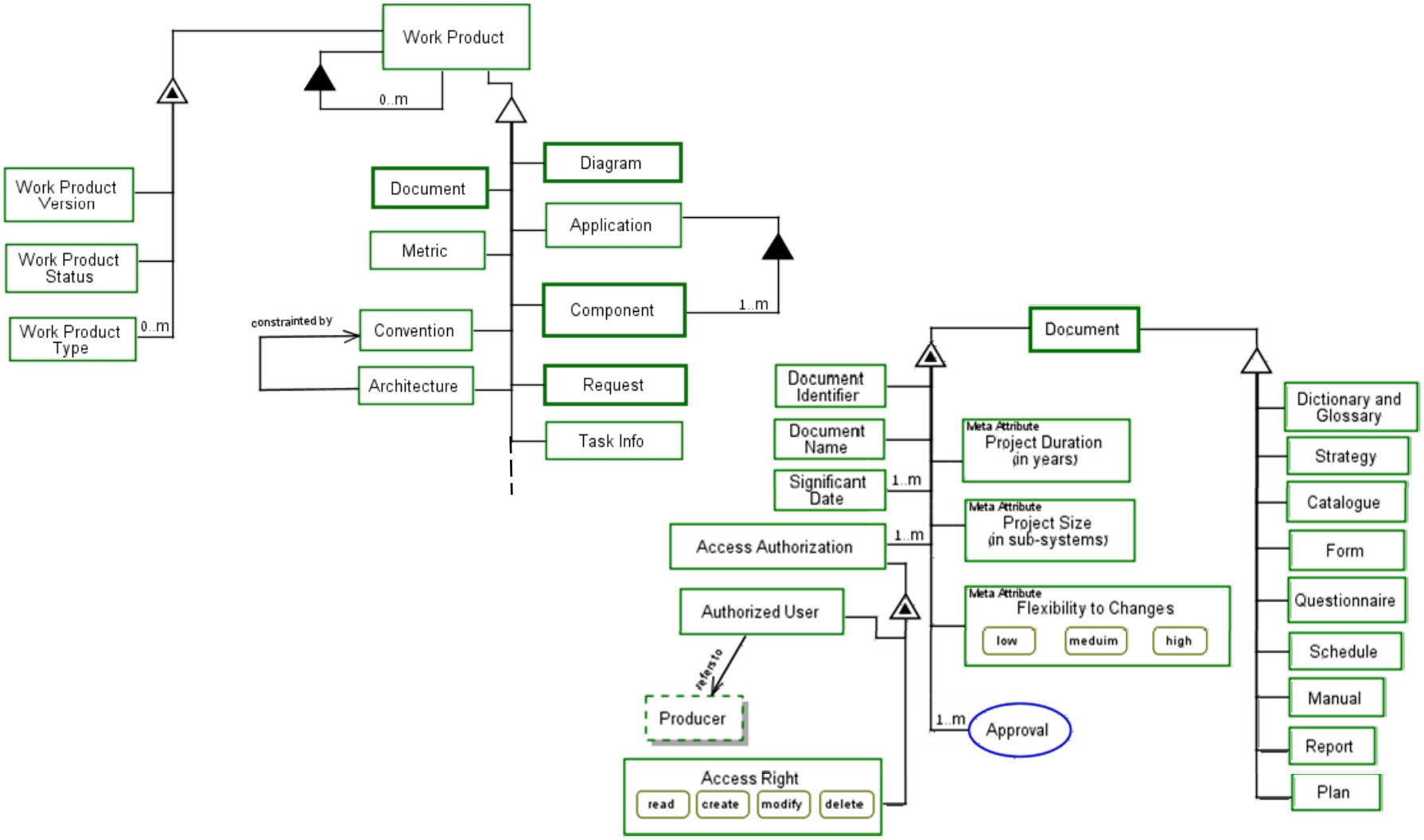
The ADOM Approach for ME

Unfolding Work Unit



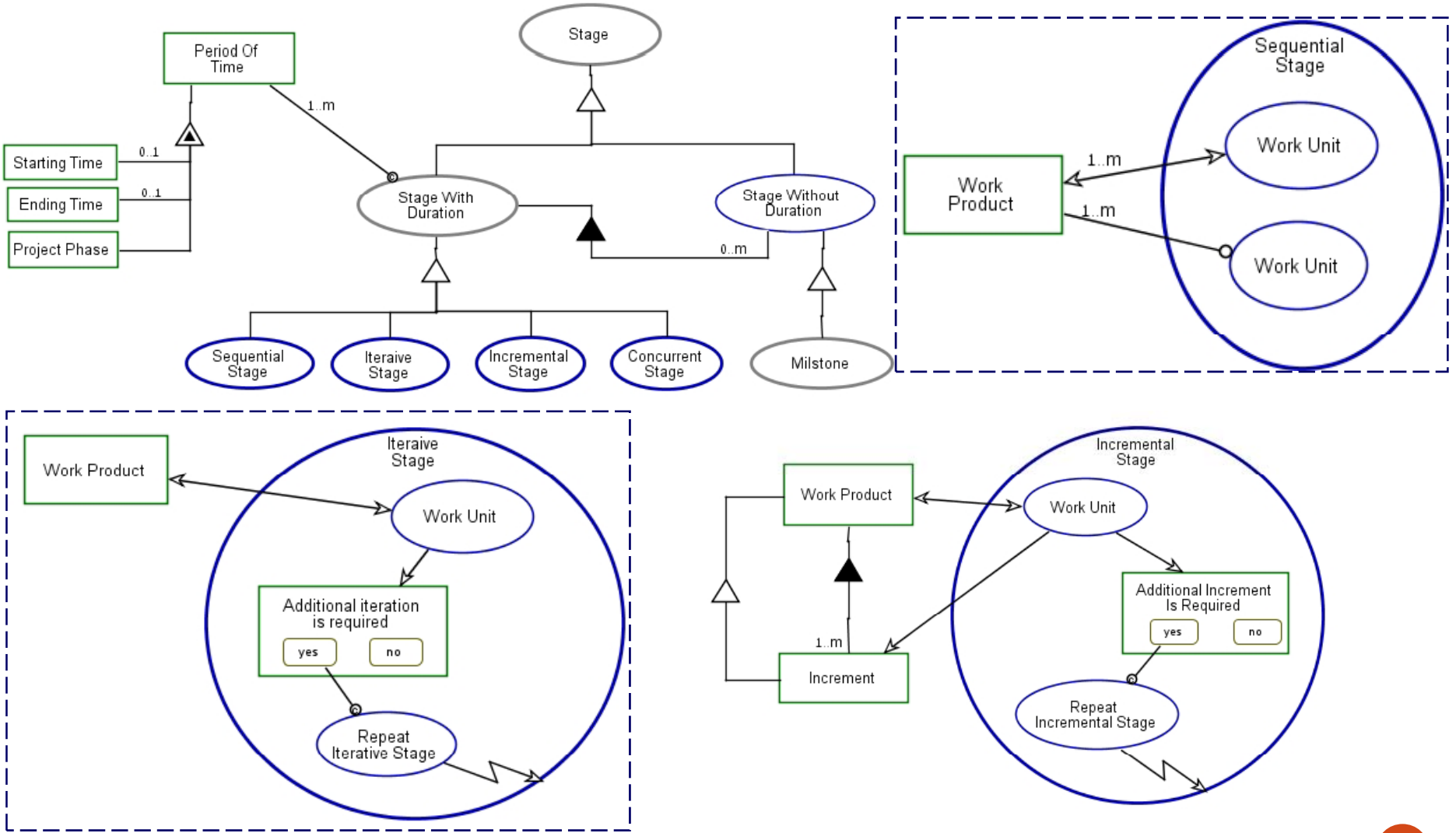
The ADOM Approach for ME

Unfolding Work Product and Document



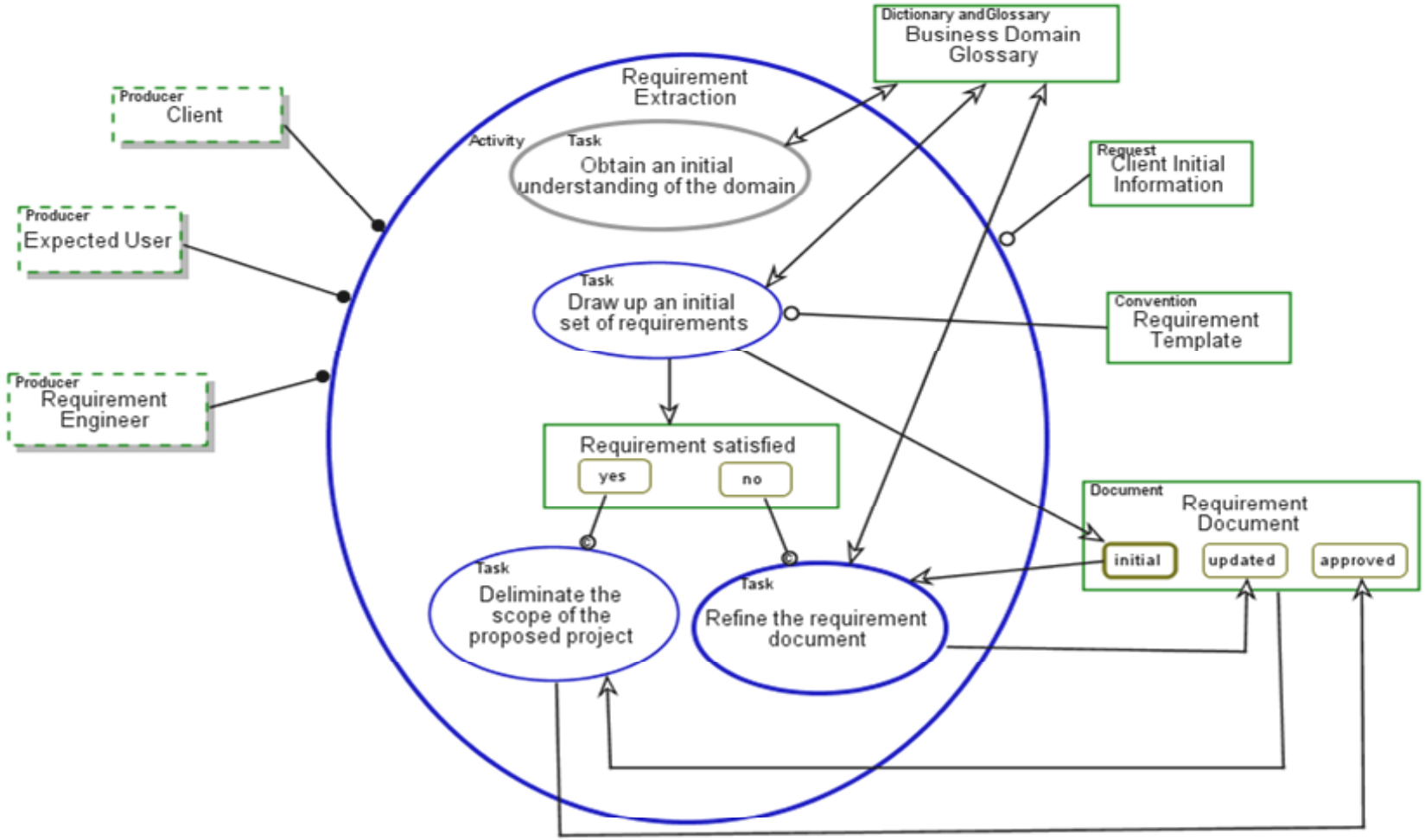
The ADOM Approach for ME

Unfolding Stage



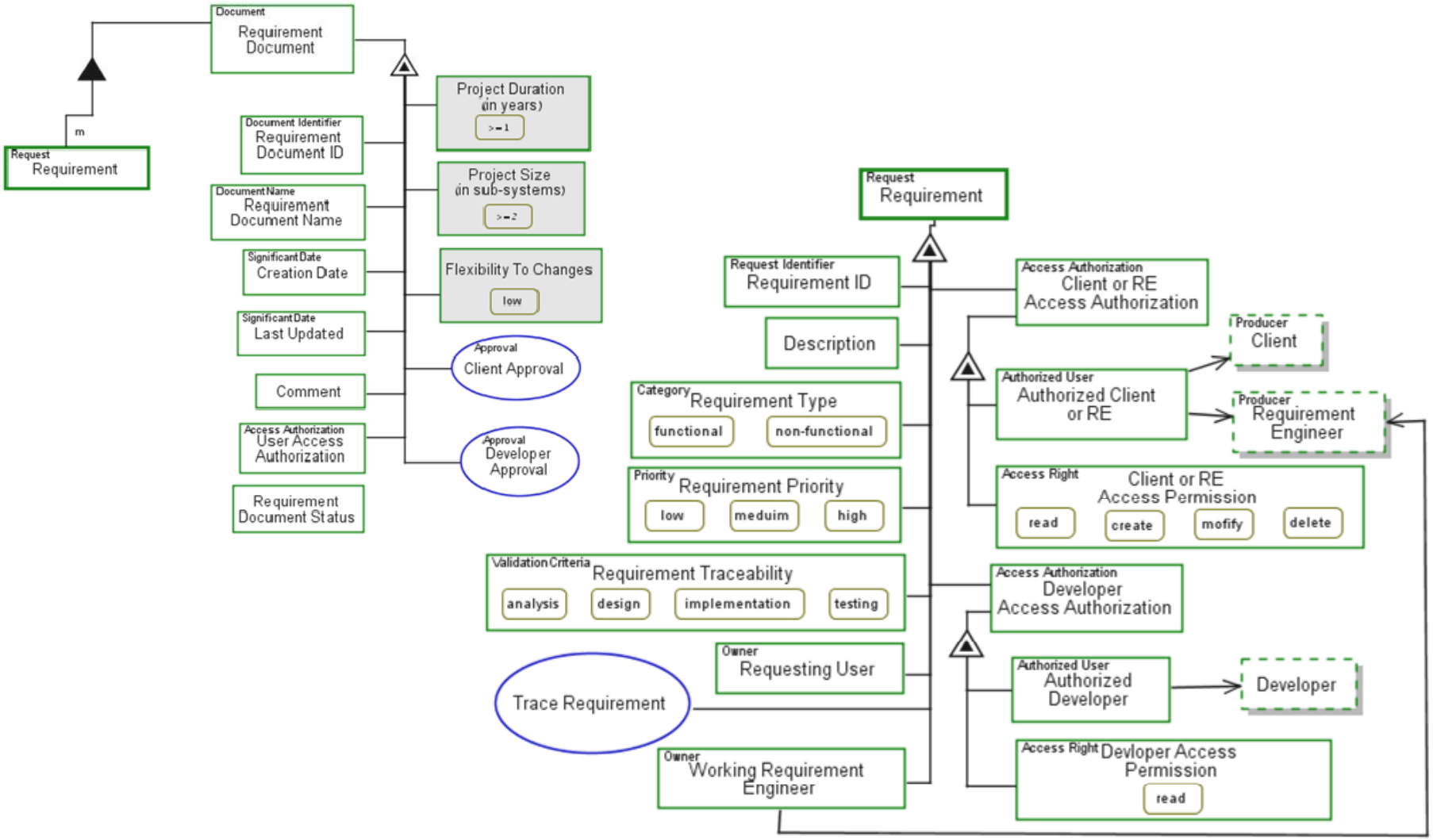
The ADOM Approach for ME

“Extract Requirements" activity (from RUP)



The ADOM Approach for ME

“Requirement Document" work product from RUP



Experimenting with ADOM

Experiment #1 Raw Results

Table 1. Performance achieved when mapping SDM elements to the human-made model elements

		Literature review			Reverse engineering		
		PM	Others	Overall	PM	Others	Overall
Class diagram	classes	100.0%	100.0%	**100.0%**	100.0%	98.9%	**99.2%**
Sequence diagrams	objects	87.5%	85.0%	**85.7%**	87.5%	96.7%	**94.0%**
	messages	94.4%	64.4%	**73.0%**	94.4%	75.0%	**80.6%**

Table 2. Grading the SDM model – experiment 1

	Model Correctness			Model Completeness			Redundancy			Diagrams consistency			Correct Stereotype Recognition			Abstraction level properness			General grade			Overall Average
	CD	SD	Overall	CD	SD	Overall	CD	SD	Overall	CD	SD	Overall	CD	SD	Overall	CD	SD	Overall	CD	SD	Overall	
PM AVG	3.5	3.9	3.7	3.3	3.7	3.5	4.2	4.5	4.3	4.4	4.6	4.5	3.1	3.8	3.5	3.8	3.7	3.8	3.4	3.6	3.5	3.82
Others AVG	3.7	4.2	3.9	3.6	3.9	3.8	4.4	4.8	4.6	3.8	3.8	3.8	3.3	4.1	3.7	3.2	3.5	3.3	3.4	3.7	3.6	3.80
Overall AVG	3.6	4.1	3.9	3.5	3.8	3.7	4.3	4.7	4.5	4.0	4.0	4.0	3.2	4.0	3.6	3.4	3.5	3.5	3.4	3.7	3.6	3.81

Legend: CD – Class Diagram, SD – Sequence Diagram

Experimenting with ADOM

Experiment #2 Raw Results

Table 3. Evaluating the relevancy of the main SDM model elements to the Scheduling domain

Part A — **Total Participants: 15**

SDM Element Name	Relevancy to Domain		
	Relevant	Not Relevant	Relevancy Percentage
ClassificationCar_Legal_document	12	3	80%
Customer_Device	11	4	73%
Customer_Person	8	7	53%
EmployeeWorker_Person	13	2	87%
Schedule_Activity	15	0	100%
Status_Statement	13	2	87%
System_Psychological_feature	13	2	87%
Ticket_Equipment	7	8	47%
WorkersClassificationDriver_Message	4	8	33%
Update_Driver_Act	3	0	100%

Experimenting with ADOM

Experiment #2 Raw Results

Table 4. Mapping the SDM elements to the Ozone ontology

SDM Elements	Equivalent Ozone Element	Correct Answers	Wrong Answers	Correct Percentage
ClassificationCar_Legal_document	Constraint	7	6	54%
Customer_Device	Product	7	6	54%
Customer_Person	Demand	6	5	55%
EmployeeWorker_Person	Resource	14	0	100%
Schedule_Activity	Activity	14	1	93%
Status_Statement	Activity	5	7	42%
System_Psychological_feature	Constraint	7	6	54%
Ticket_Equipment	Resource	3	6	33%
WorkersClassificationDriver_Message	Constraint	5	4	56%
Customer_Device - System_Psychological_feature	Imposes	2	1	67%
Ticket_Equipment - System_Psychological_feature	Imposes	2	0	100%
Customer_Person - System_Psychological_feature	Imposes	2	1	67%
EmployeeWorker_Person - System_Psychological_feature	Imposes	2	0	100%
Schedule_Activity - EmployeeWorker_Person	Requires	3	0	100%
Schedule_Activity - Ticket_Equipment	Requires	3	0	100%
Schedule_Activity - ClassificationCar_Legal_document	Restricts	2	1	67%
Schedule_Activity - System_Psychological_feature	Restricts	2	1	67%
ClassificationCar_Legal_document - System_Psychological_feature	Imposes	2	0	100%
		Average:		**73%**

Back

Experimenting with ADOM

Experiment #2 Raw Results

Table 5. Mapping the Ozone ontology to the SDM elements

Ozone Elements	Equivalent SDM Element	Correct Answers	Wrong Answers	Correct Percentage
Activity	Schedule_Activity	11	2	85%
Product	Customer_Device	8	6	57%
Demand	ClassificationCar_Legal_document	9	6	60%
Resource	EmployeeWorker_Person	10	5	67%
Constraint	System_Psychological_feature	8	6	57%
		Average:		**65%**

Table 6. Grading the SDM model – experiment 2

Model Correctness	Model Completeness	Redundancy	Correct Stereotype Recognition	Abstraction level properness	General grade	Overall Average
3.9	3.7	3.9	2.3	2.3	2.5	3.1

Domain Analysis Techniques

References (1)

- Arango, G. "Domain analysis: from art form to engineering discipline", Proceedings of the Fifth International Workshop on Software Specification and Design, p.152-159, 1989.
- Carnegie, M. "Domain Engineering: A Model-Based Approach", Software Engineering Institute, http://www.sei.cmu.edu/domain-engineering/, 2002.
- Champeaux, D. de, Lea, D., and Faure, P. Object-Oriented System Development, Addison Wesley, 1993.
- Cleaveland, C. "Domain Engineering", http://craigc.com/cs/de.html, 2002.
- Clauss, M. "Generic Modeling using UML extensions for variability", Workshop on Domain Specific Visual Languages, Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'01), 2001.
- Davis, J. "Model Integrated Computing: A Framework for Creating Domain Specific Design Environments", The Sixth World Multiconference on Systems, Cybernetics, and Informatics (SCI), 2002.
- Gomma, H., Designing Software Product Lines with UML, 2004.

Domain Analysis Techniques

References (2)

- Gomma, H. and Eonsuk-Shin, M. "Multiple-View Meta-Modeling of Software Product Lines", Proceedings of the Eighth IEEE International Confrerence on Engineering of Complex Computer Systems, 2002.
- Gomaa, E. and Kerschberg, L. "Domain Modeling for Software Reuse and Evolution", Proceedings of Computer Assisted Software Engineering Workshop (CASE 95), 1995.
- Harel, D. Statecharts: a Visual Formalism for Complex Systems. Science of Computer Programming 8: 231-274, 1987.
- Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A.,"Feature-Oriented Domain Analysis (FODA) Feasibility Study", CMU/SEI-90-TR-021 ADA235785, 1990.
- Morisio, M., Travassos, G. H., and Stark, M. "Extending UML to Support Domain Analysis", Proceedings of the Fifth IEEE International Conference on Automated Software Engineering, pp. 321-324, 2000.
- Nordstrom, G., Sztipanovits, J., Karsai, G., and Ledeczi, A. "Metamodeling - Rapid Design and Evolution of Domain-Specific Modeling Environments", Proceedings of the IEEE Sixth Symposium on Engineering Computer-Based Systems (ECBS), pp. 68-74, 1999.
- Petro, J. J., Peterson, A. S., and Ruby, W. F. "In-Transit Visibility Modernization Domain Modeling Report Comprehensive Approach to Reusable Defense Software" (STARS-VC-H002a/001/00).

The ADOM Approach
References (1)

- **For SE:**
  - I. Reinhartz-Berger and A. Sturm, Behavioral Domain Analysis – The Application-based Domain Modeling Approach, the 7th International Conference on the Unified Modeling Language (UML'2004), Lecture Notes in Computer Science 3273, pp. 410-424, 2004.
  - A. Sturm and I. Reinhartz-Berger, Applying the Application-based Domain Modeling Approach to UML Structural Views, the 23rd International Conference on Conceptual Modeling (ER'2004), Lecture Notes in Computer Science 3288, pp. 766-779, 2004.
  - I. Reinhartz-Berger and A. Sturm. Enhancing UML Models: A Domain Analysis Approach, Journal on Database Management (JDM) 19 (1), special issue on UML Topics, pp. 74-94, 2007.

The ADOM Approach
References (2)

- **For BP:**
  - I. Reinhartz-Berger, P. Soffer, and A. Sturm, A Domain Engineering Approach to Specifying and Applying Reference Models, Enterprise Modeling and Information Systems Architectures (EMISA'05), pp. 50-63, 2005.
  - P. Soffer, I. Reinhartz-Berger, and A. Sturm. Facilitating Reuse by Specialization of Reference Models for Business Process Design, the 8th Workshop on Business Process Modeling, Development, and Support (BPMDS'07), in conjunction with CAiSE'07.
  - P. Soffer, I. Reinhartz-Berger, and A. Sturm, Matching Models of Different Abstraction Levels: A Refinement Equivalence Approach. In Keng Siau (Ed.): Contemporary Issues in Database Design and Information Systems Development. Idea Group, pp. 89-122, 2007.

The ADOM Approach
References (3)

- **For ME:**
  - I. Reinhartz-Berger and A. Aharoni. Representation of Method Fragments: A Domain Engineering Approach, the 12th Workshop on Exploring Modeling Methods for Information Systems Analysis and Design (EMMSAD'07), in conjunction with CAiSE'07.
  - A. Aharoni and I. Reinhartz-Berger. Representation of Method Fragments: A Comparative Study, IFIP WG8.1 Working Conference on Situational Method Engineering: Fundamentals and Experiences (ME'07).
  - A. Aharoni and I. Reinhartz-Berger. A Domain Engineering-based Approach for Situational Method Engineering, the 27th International Conference on Conceptual Modeling (ER'2008), 2008