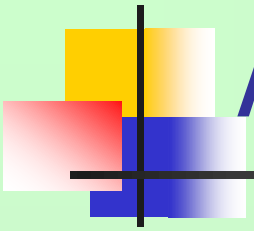
# Tutorial
# on
# Meta Models and Meta Execution Models

ICSE 2014

June 3, 2014

Kamalakar Karlapalem, Centre for Data Engineering, IIIT-Hyderabad, India

P. Radha Krishna, Infosys Labs, Infosys Limited, India



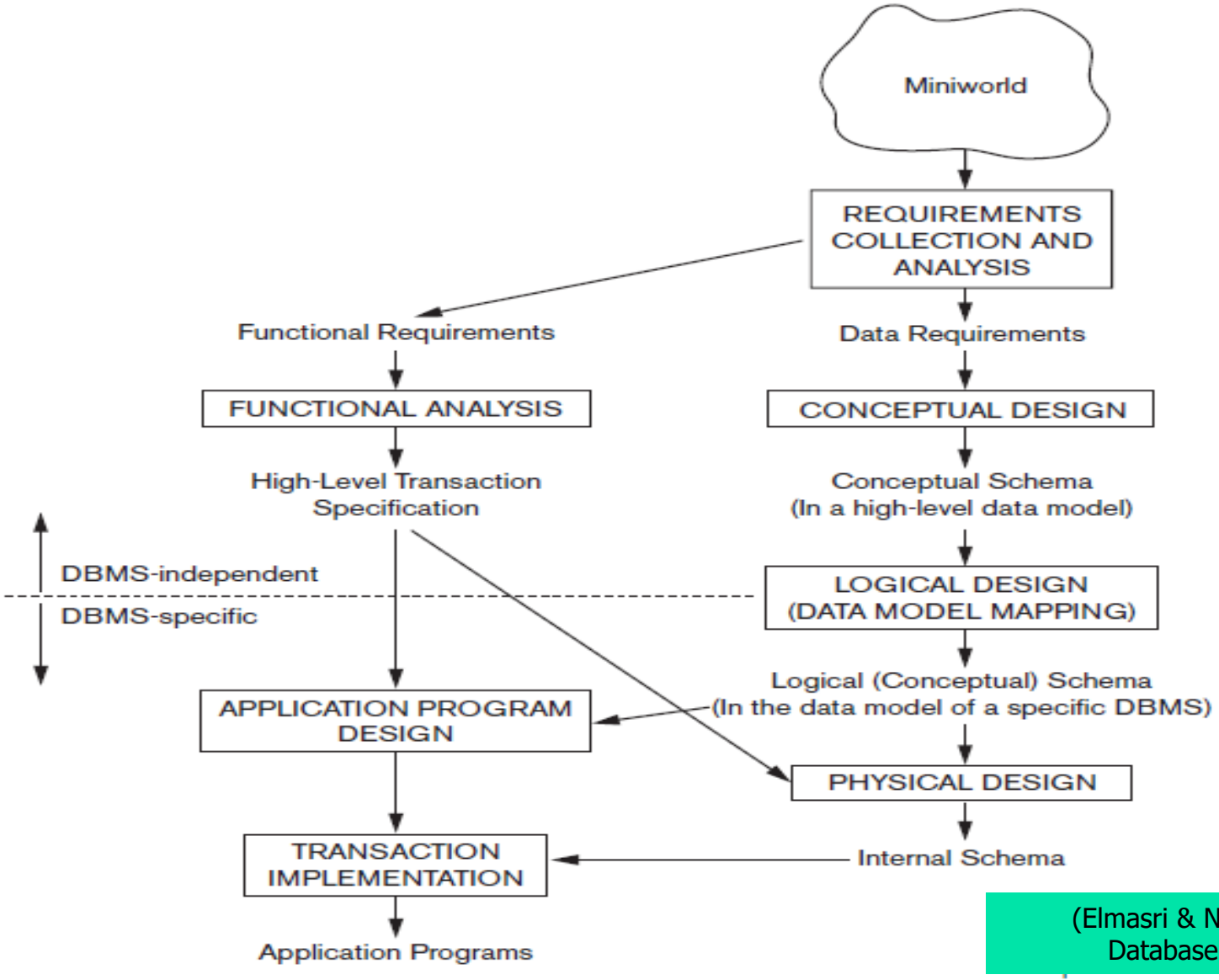
# Agenda

1. Introduction [20 Minutes]
    1. Understanding Conceptual Modeling, Meta Models and Meta Execution Models
2. Motivation [10 minutes]
    1. Issues with Complex Information Systems,
    2. Run-time change and design time change, etc.
3. Basic Notions [ 20 minutes]
4. Meta Models [30 minutes]

   Break
5. Meta Execution Models [30 minutes]
6. Implementation and Deployment Perspectives [20 minutes]
7. Related Software Engineering Artifacts [ 20 minutes]
8. Summary [10 minutes]
9. Open Problems and Discussion [20 Minutes]

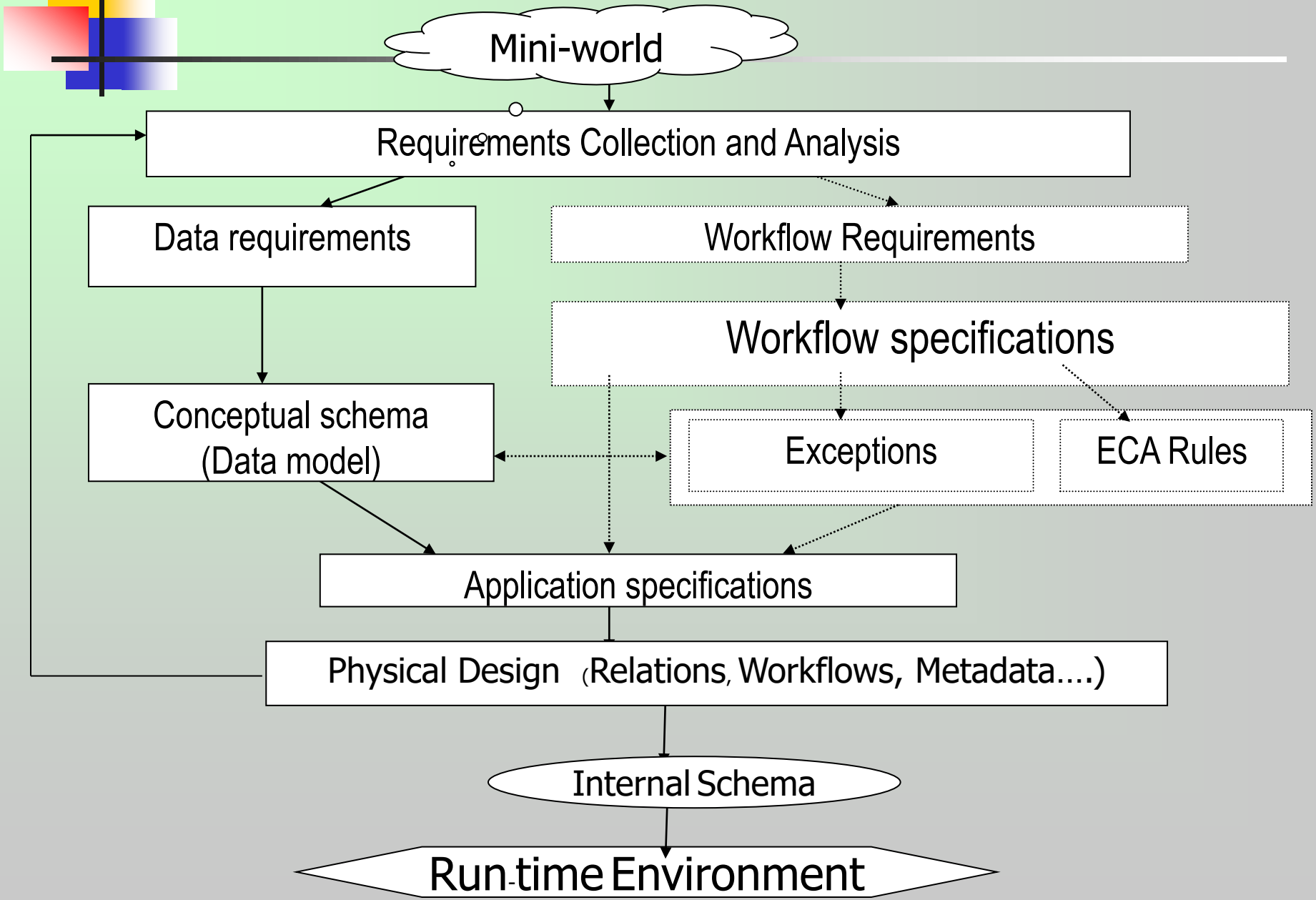
# Background

# Database Design for Applications

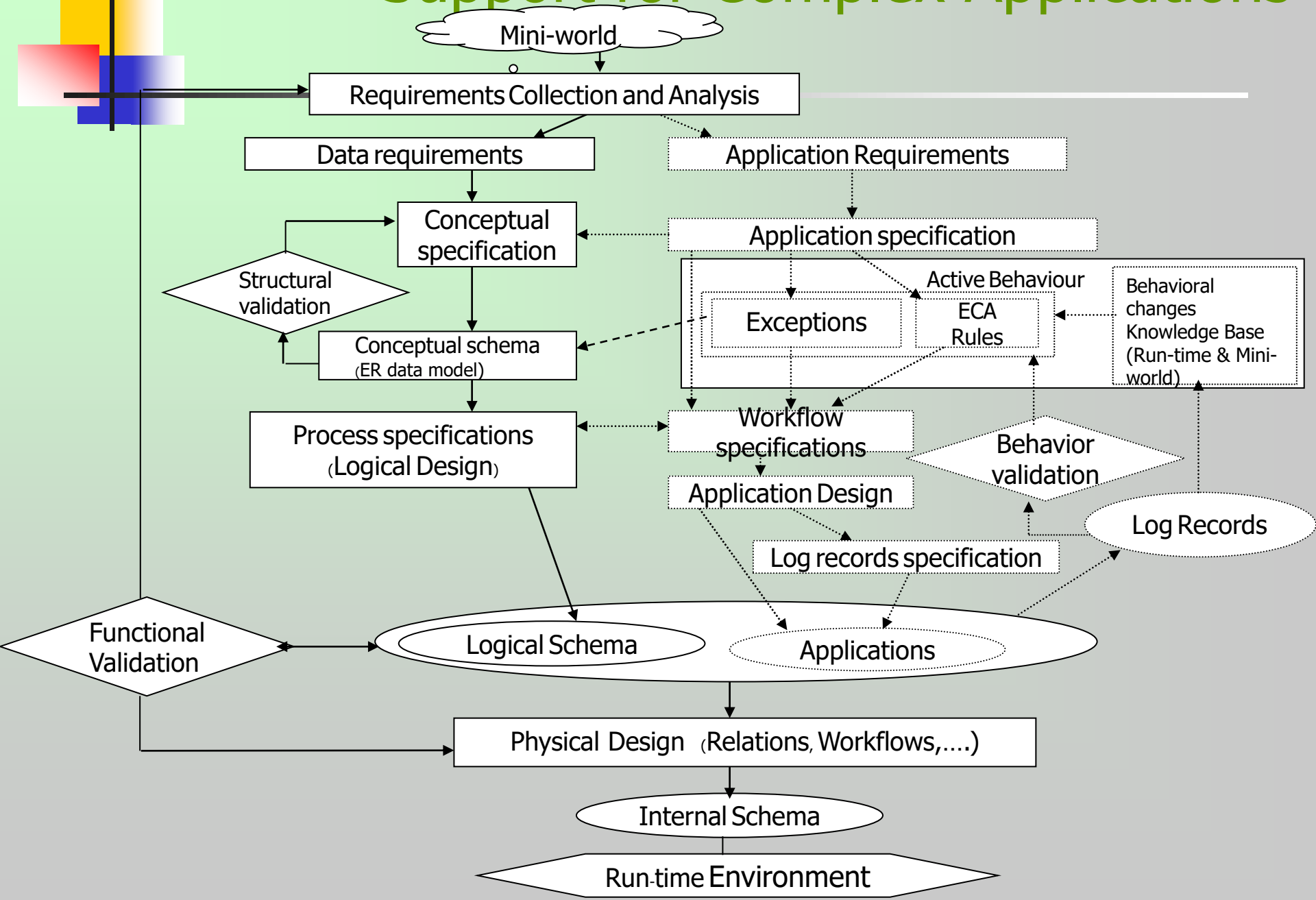


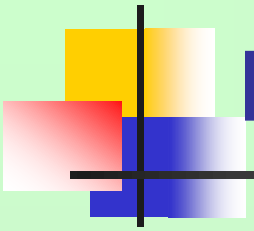

(Elmasri & Navathe (EN), Database Systems)

# Application Support with Workflows



# Support for Complex Applications





# Background

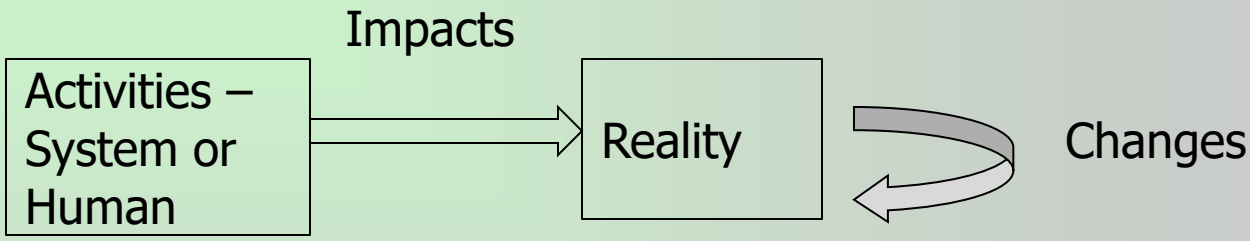
- Information System Design
  - Articulate specifications of the system
- Information System Development
  - Implementation of specifications got during system design on a particular platform
- Gap ?

System design and implementation differentiate between the specification of the system and its implementation.

Any change in design → redesign of implemented system

# Background

## Workflow Activity Execution



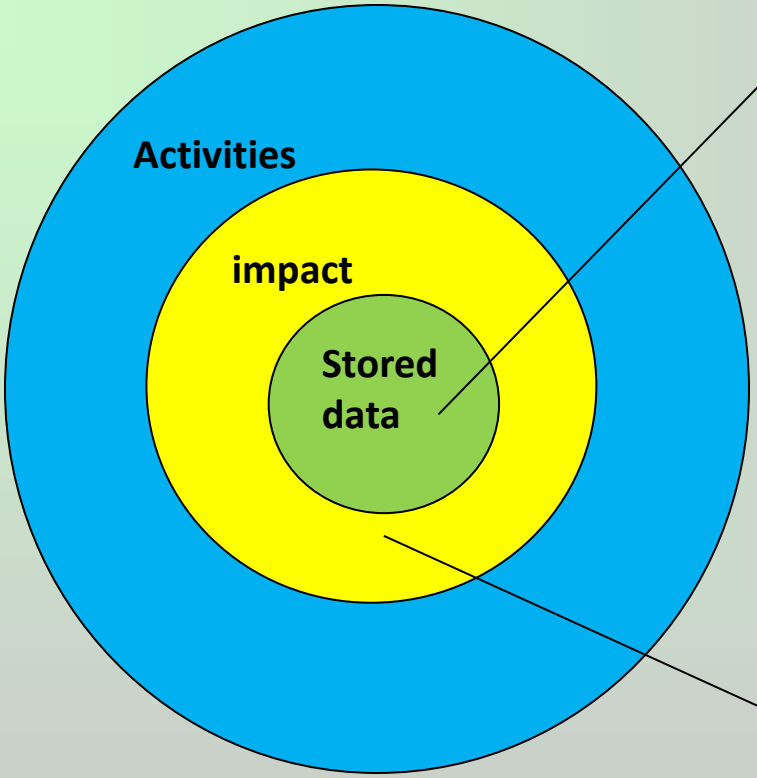

- Changes in reality managed by workflows/processes

How to specify the impact in terms of What, Where, Why, How, When and Who?

Need constructs → to represent the real-world !!!

Why ensuring reflection of changes in reality important ?

# Background

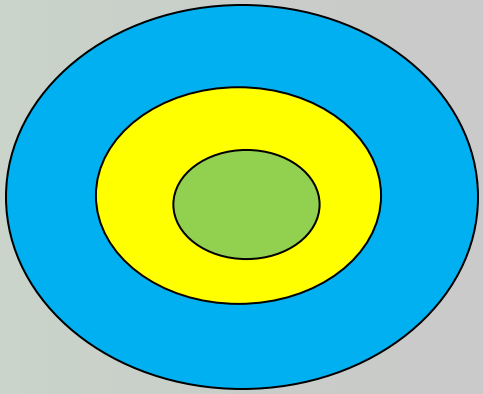



Manifestation of reality

Manifest changes in reality through change in stored data

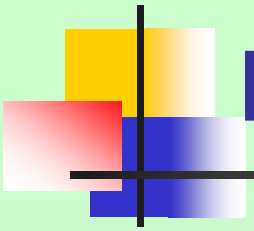
In the database design, the reality is facilitated by storing the data according to the end-users' perception for an application in order to realize the impact.

# Background



- So, database is a model (stored manifestation) of reality in the sense that the database represents a selected set of data that can be managed by the database

Database Creation to store the data is a fundamental step in coming to know the nature and status of that reality



# Background

- Capturing and managing data about ***real world*** is too complex a task.
- *Models* enable reality that is intended to represent real world aspects as closely as possible
- Models also helps in understanding the reality as well as to know its status and manage it.

***The constructs of a model should reflect the real-world reality.***

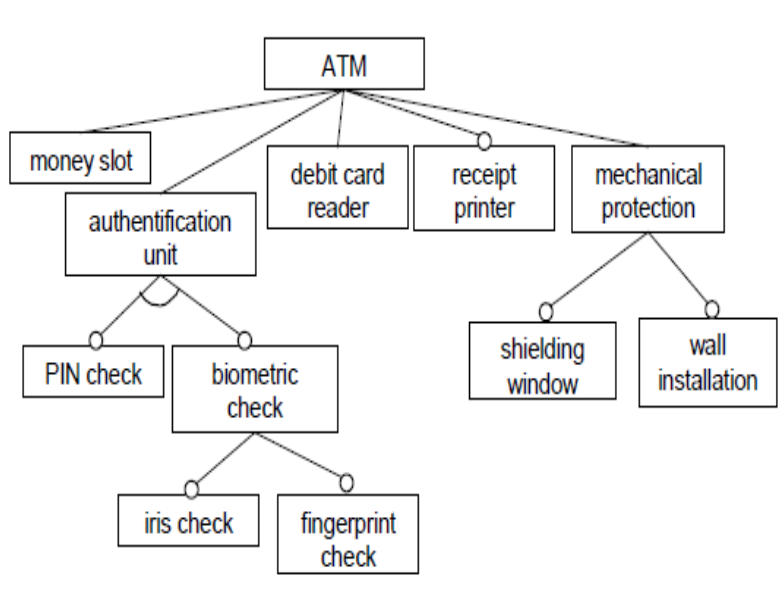
Change of reality can be simple (ex. updating balance) or complex (ex. introduction of new payment instrument).

# Motivation

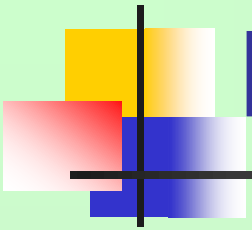
Models are used as manifestation of either processing requirements or data requirements to capture and manage reality.

# Role of Models in Software Engineering

- Models are used for a precise description of systems at the appropriate abstraction level without unnecessary details.
- Fractured reality towards multiple systems and partial information
  - Models for Requirements, design, development, testing and deployment

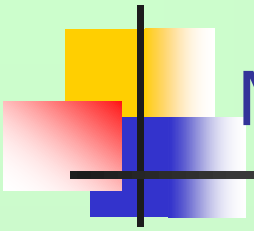



Feature Model Example for ATM



# Motivation

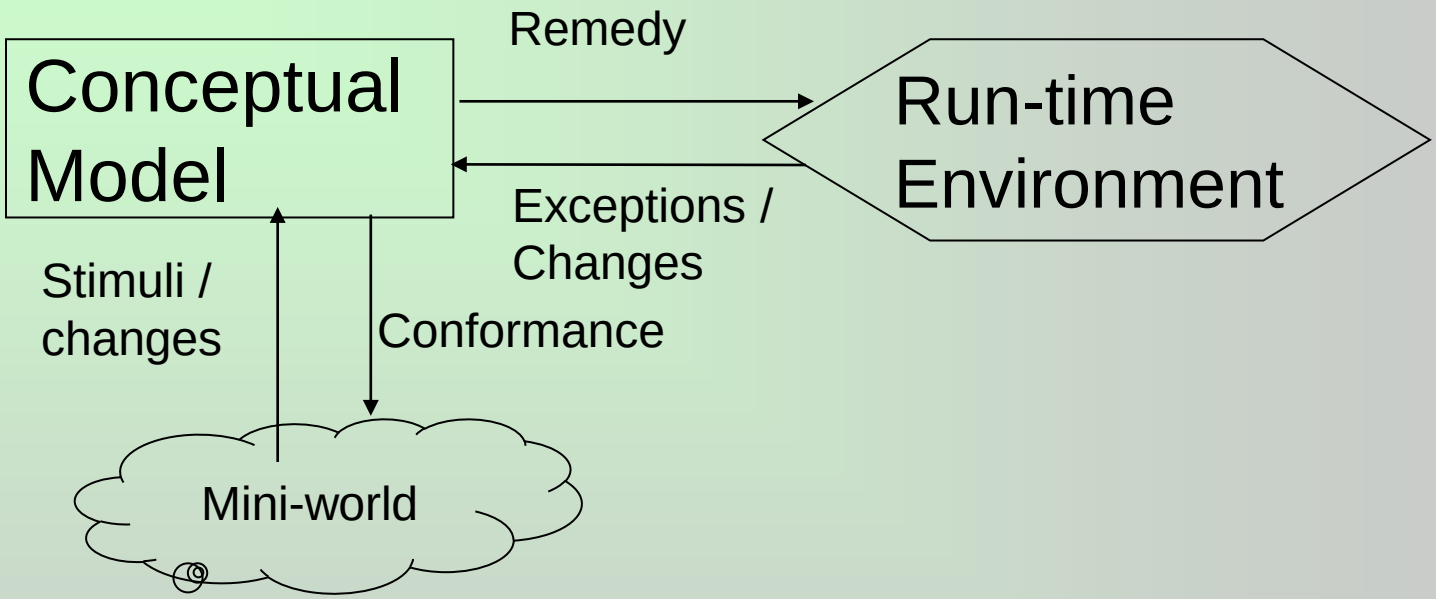
- Complex software development is a big problem
- Complexity of Models
  - Ex.: Relationships among varying subsets of entities
- Changes are inevitable
  - Due to competitive, dynamic and volatile businesses
  - Run-time and Mini-world changes
    - Expected and unexpected
      - Need to handle both
  - Need new constructs to handle new complexities
    - Complexities arise due to changes in facts and relations
    - Ex.: Handling unknown events and exceptions at run-time, role changes, new kind of relationships



# Motivation

- Traditional Software Engineering Principle "Design Once, Use Forever" may not work
- Focus Shift :
  - Re-design Models → Models Self-Design themselves
    - To adopt the changes in reality
    - Lack of clarity on how self-design works – when reality changes – how is it reflected in the model
  - Capture active behavior and respond
    - Mini-world and Run-time changes
- Models can be used to decide structure, behavioral and functional decomposition of information systems dictated by, say, WFMS that ensure consistent execution.

# Conceptual Modeling Framework

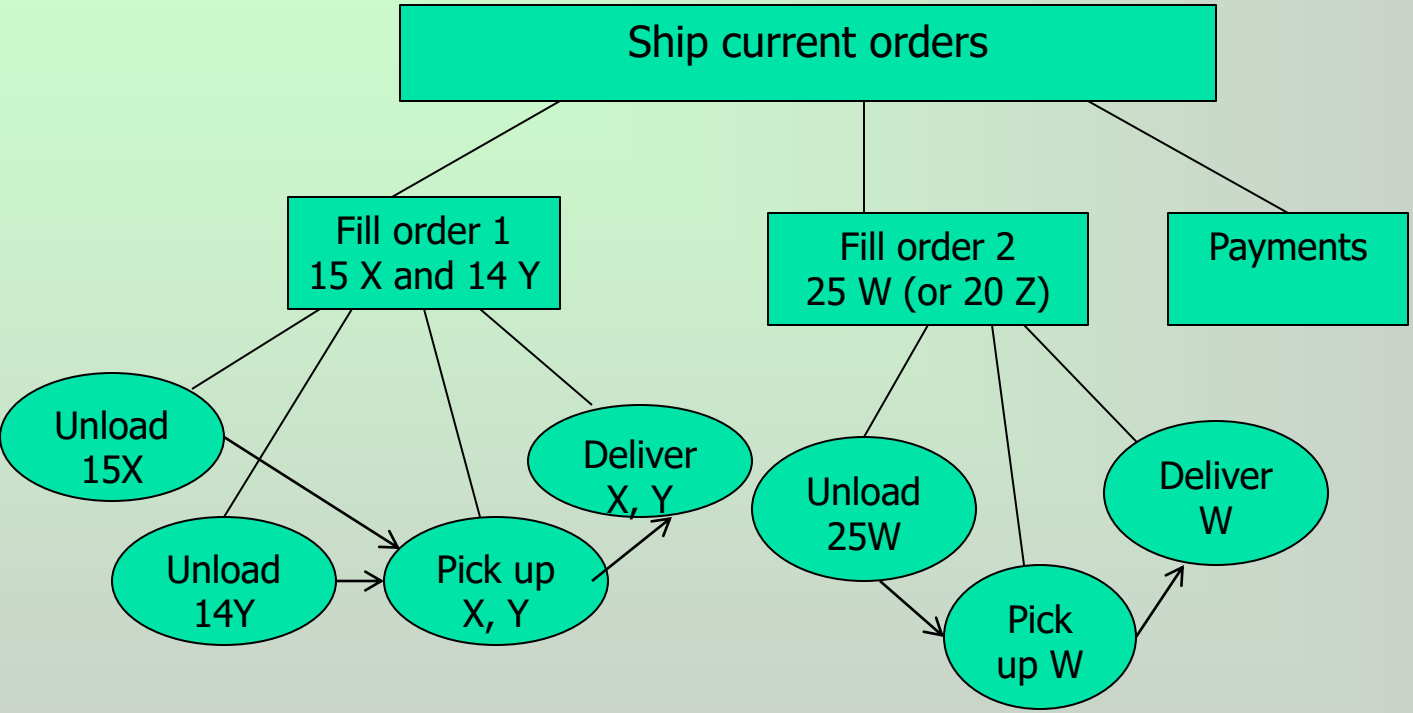



***Modeling complex applications require both human and system driven specification and deployment in order to handle the active behavior of applications.***

# Meta Models (data) and Meta Execution (run time) Models

- Objective:
  - To support integrated management of information systems that manages changes to reality
- Two Major Goals:
  - Develop executable (conceptual) Models
    - Direct mapping from conceptual to implementation level constructs
  - Support evolving (changing reality) requirements
    - Reality changes – how is it supported?
- Simplify (ex. workflow driven) the overall software engineering lifecycle  for certain class of applications

# Shipping Material Example



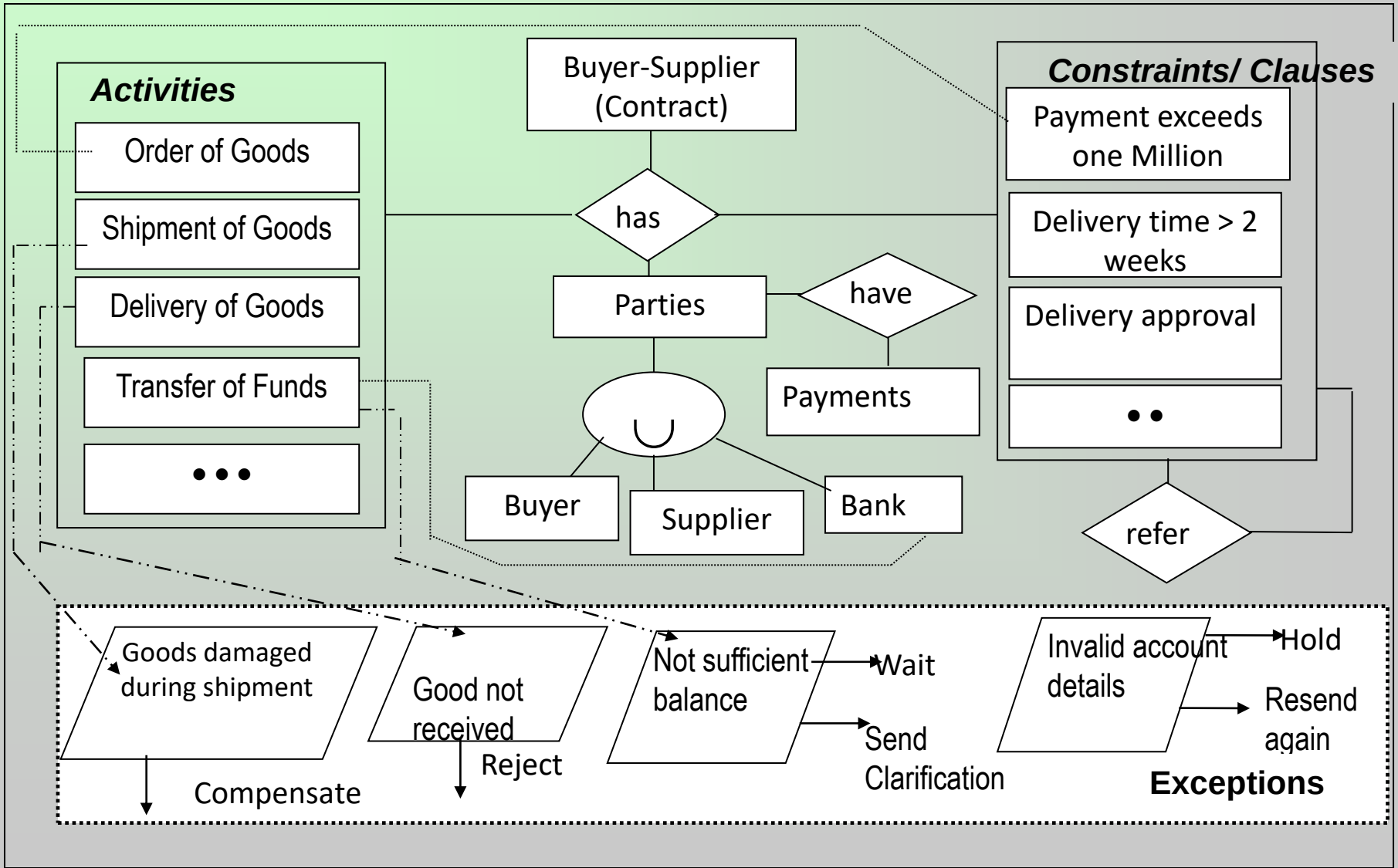

- OR and AND conditions
- Hierarchical relationships
- Precedence relationships
- Constraints (ex. time limit)
- Payment processing

Assume that the items are inspected (manually) before delivery.

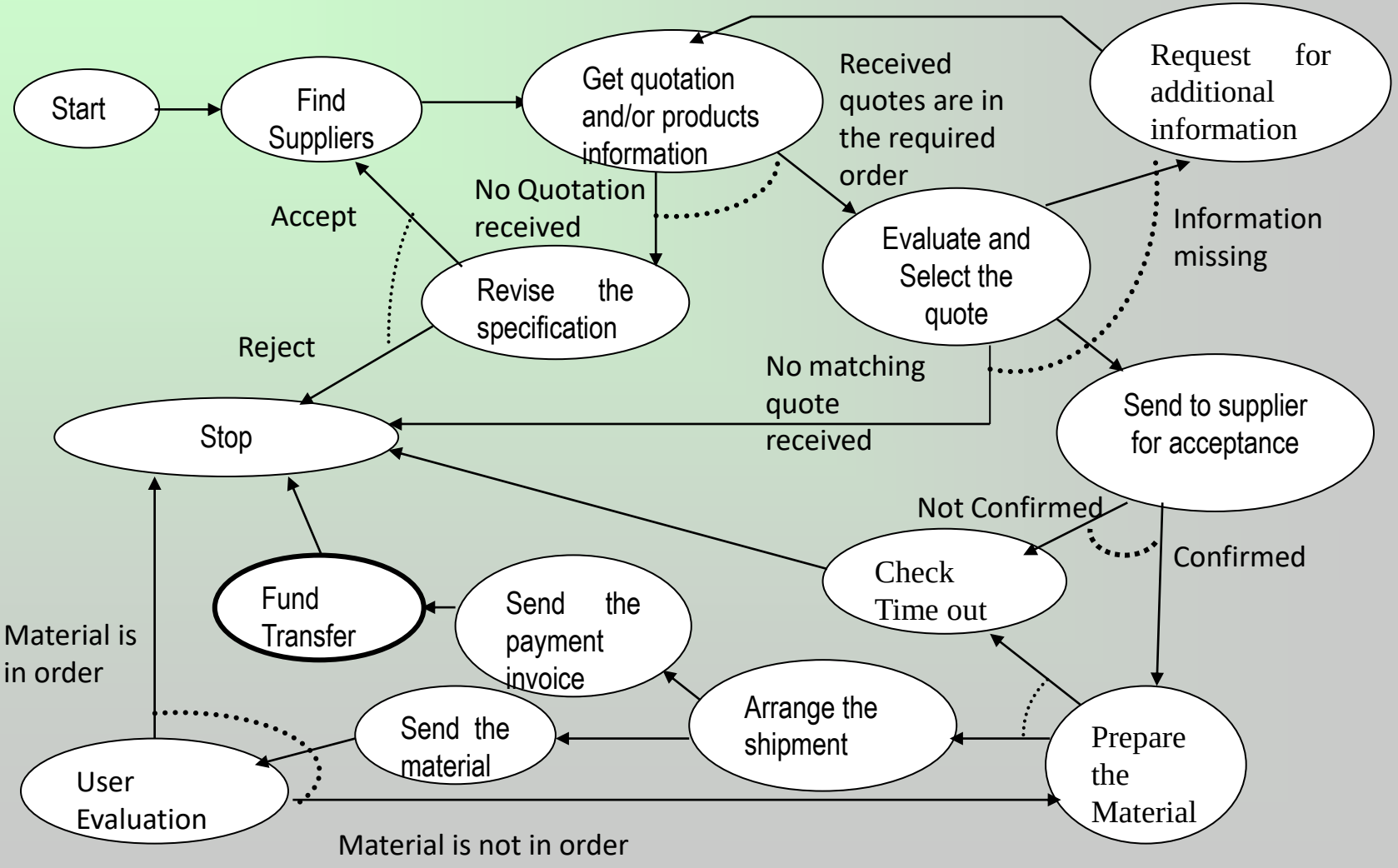
Damaged goods should be replaced/ compensated /imposed penalty.

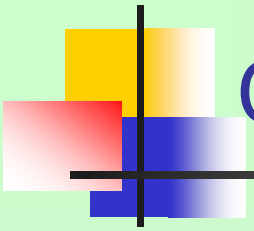
It requires two orders of specified quantities of items X, Y and W (or Z) to ship. The solid edges represent parent-child relationships in the tree. The arrows represent precedence constraints (i.e. a → b means that task a must precede task b).

# Conceptual Modeling Example



# An Example (partial) workflow





# Our work

- Enable on-the-fly changes to how best changes to reality can be accommodated.
  - Basically due to seamless coupling from conceptual to implementation level
- Concentrate on verticals/problems where there can be end to end solution. For example, SAP can use our technology to integrate their solutions.
  - This seamless coupling can be incorporated in large systems for certain class of applications

Focus is towards database oriented conceptual modeling and workflows

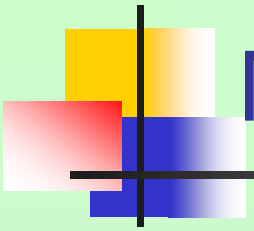
# Focus & Scope

- Our focus is not on the process of building software and techniques to model and build artifacts for such software
- Our focus is on - ideation of Actionable framework (through meta models and meta execution models) which drives the specification and execution of business systems/processes

Supporting change is costly in SE, and in our MEM, change is possible, dynamic, almost real-time and cheap.

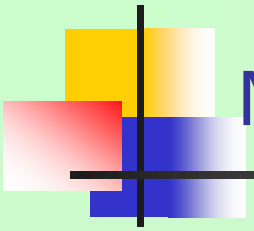
# Introducing Meta Models and Meta Execution Models

- Evolved large number of Meta-related themes



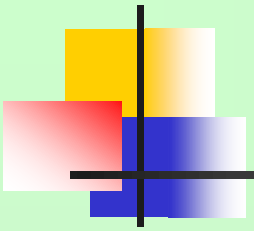
# Meta Data

- Meta Data
  - A well-known and widely used term
  - “Data about the Data”
  - Describe the data
- Meta Model
  - A model about the model
  - Goes beyond basic structural artifacts
    - Facilitates specification of the behavioral semantics
    - Describes Dynamic behavior
    - Can also model (using active database concepts) requisite changes to mappings – and perform them.

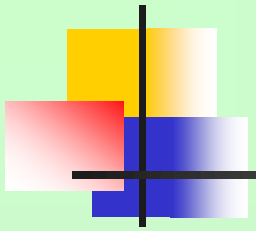


# Meta Execution [Smith84]

- A program execution operates on data; a meta-execution operates on a program execution.
- In our case – there are software engines that drive workflow execution, we model the execution logic of these software engines as a workflow – and support run-time changes to this execution workflow !

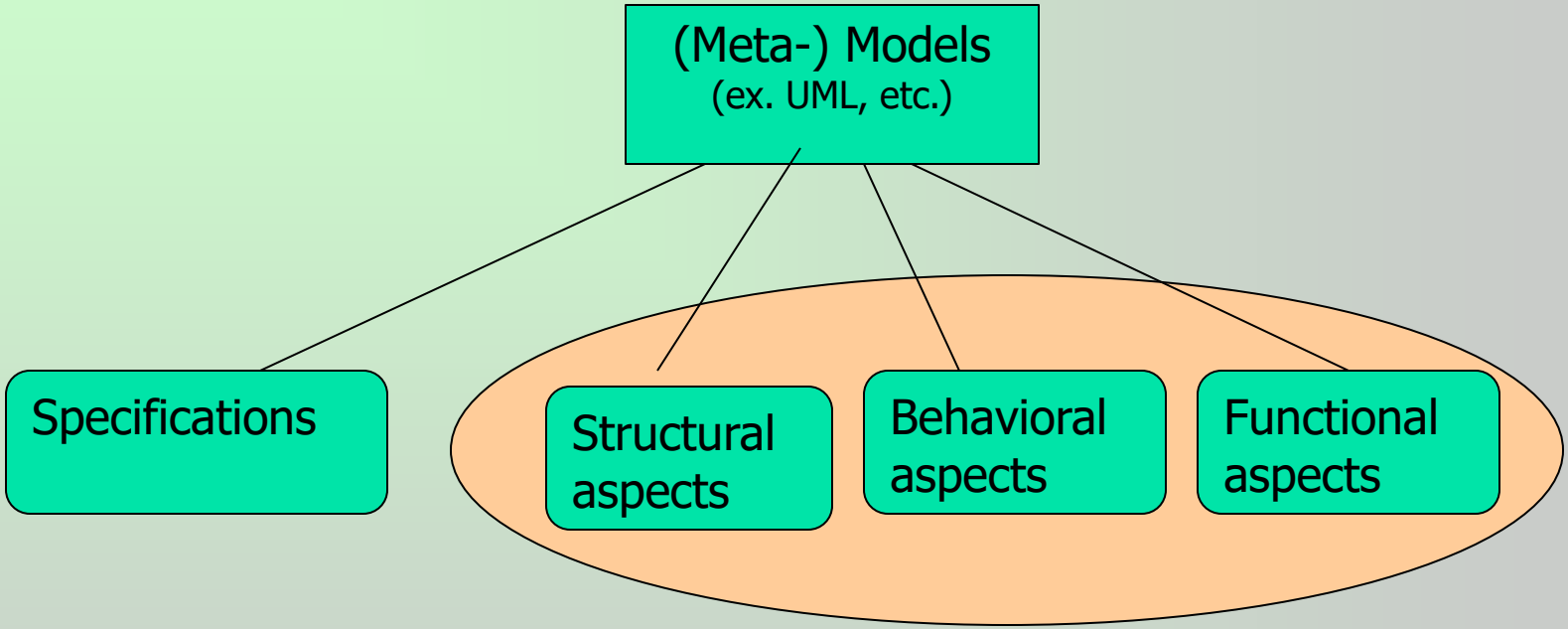


- *Data* and *Processes* are two critical aspects in software engineering lifecycle.
- *Models* enable reality that is intended to represent the real world aspects as closely as possible.
- The data and process requirements for reality can be done by conceptual models (ex. XML, ER, etc.).
- Processing of changes can be supported by workflows.



- *Data* and *Processes* are two critical aspects in software engineering lifecycle.
- *Models* enable reality that is intended to represent the real world aspects as closely as possible.
- The data and process requirements for reality can be done by conceptual models (ex. XML, ER, etc.).
- Processing of changes can be supported by workflows.

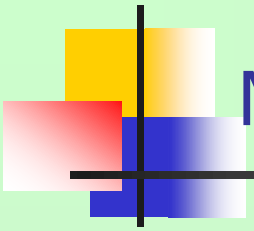
*Some of the changes in reality impact changes to the model, and frequent such changes to model can be supported by meta model. Further, we need meta execution model to facilitate the way the processing to happen when there are changes in reality.*




**For example, UML meta-model supports a particular methodology or process**

A meta issue is whether meta-models follows simplicity or increases the complexity (in understanding model artifacts and design pragmatics) !!!

# Basic Notions



# Models

- Model: Representation of the data/process which can facilitate specification and implementation of application/framework.
- *Models* enable reality that is intended to represent the real world aspects as closely as possible.
- A Model is governed by the constructs it has and semantics of these constructs have capabilities such as **represent**, **activate**, **dictate** and **enforce** for data and processes

- Limitation - Model is limited by the capabilities of the constructs to model the reality.

.

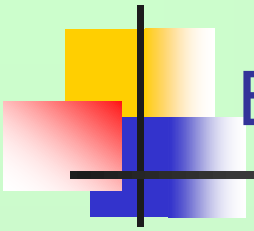
How does this gap between the model (or schema) and reality to be filled

**Need Various Models**

# Meta Models

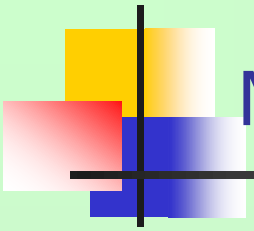
- Meta-model: An explicit model of the constructs needed to build specific models for applications. The developed model must be in accordance with its meta-model.
  - A model or an abstraction which highlights the properties of the model itself.
  - Defines rules and processes which need to be followed to define a model.
- Useful to define (or augment) new constructs, instances, constraints and semantics and for supporting reusability.
  - Applicability to various domains and instantiate data models according to application requirements
- Meta-modeling: The procedure in building meta-models
  - Helps in conceptualizing and instantiating appropriate (customized) data models and provides required facilities to support the functionality required for adapting a data model to changing requirements.

The commonalities, the differences and inadequacies in data models can be captured and pursued as a Meta Model.



# Execution Model (more on Workflows)

- Execution Model : A model to represent set of concepts and/or constructs in the desired order to achieve execution of tasks/activities for running an application.
  - Define procedure (or workflow) of execution engine to execute a set of tasks
  - Mostly fixed (hardcoded)
- Processes get executed by the execution model, which dictate how the models are executed.

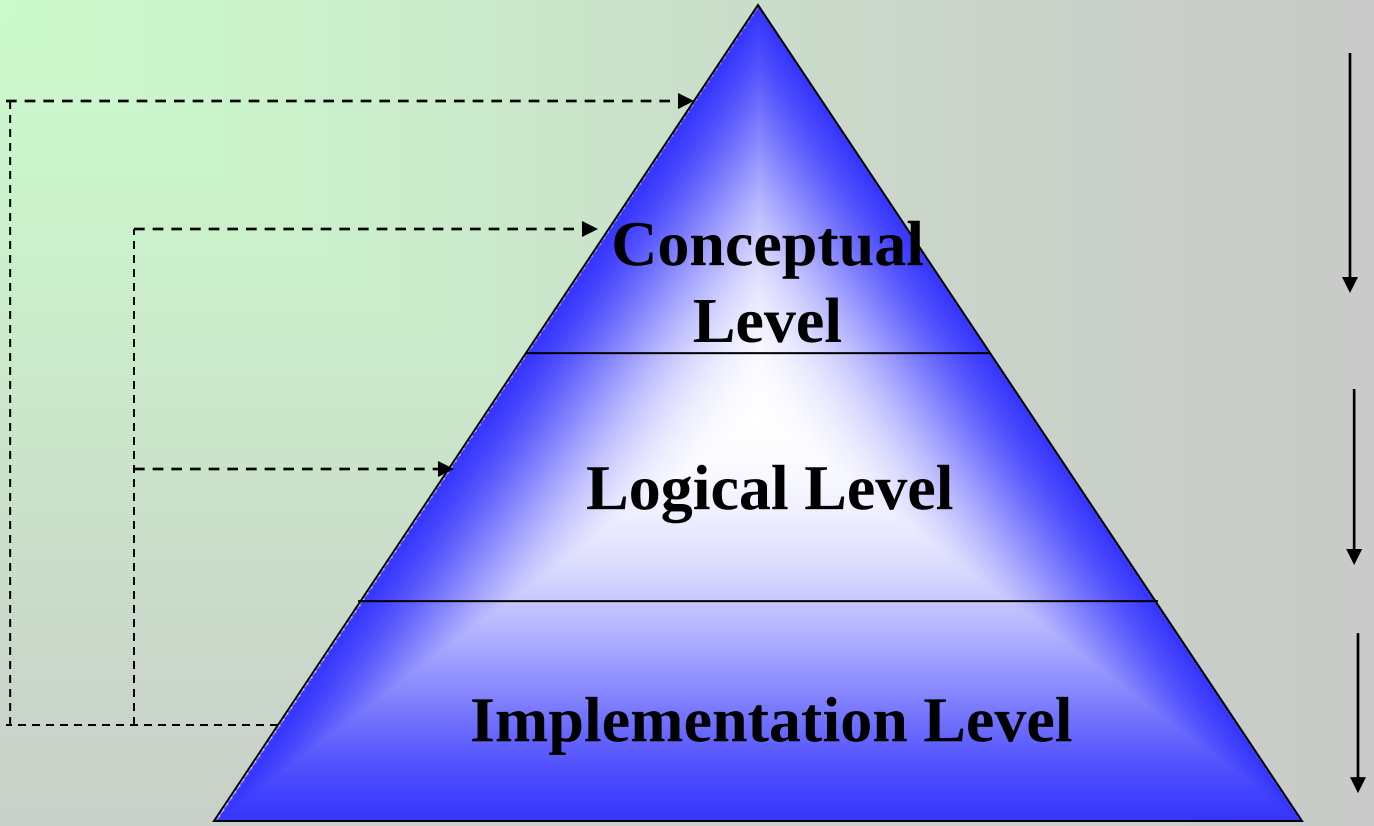


# Meta Execution Model

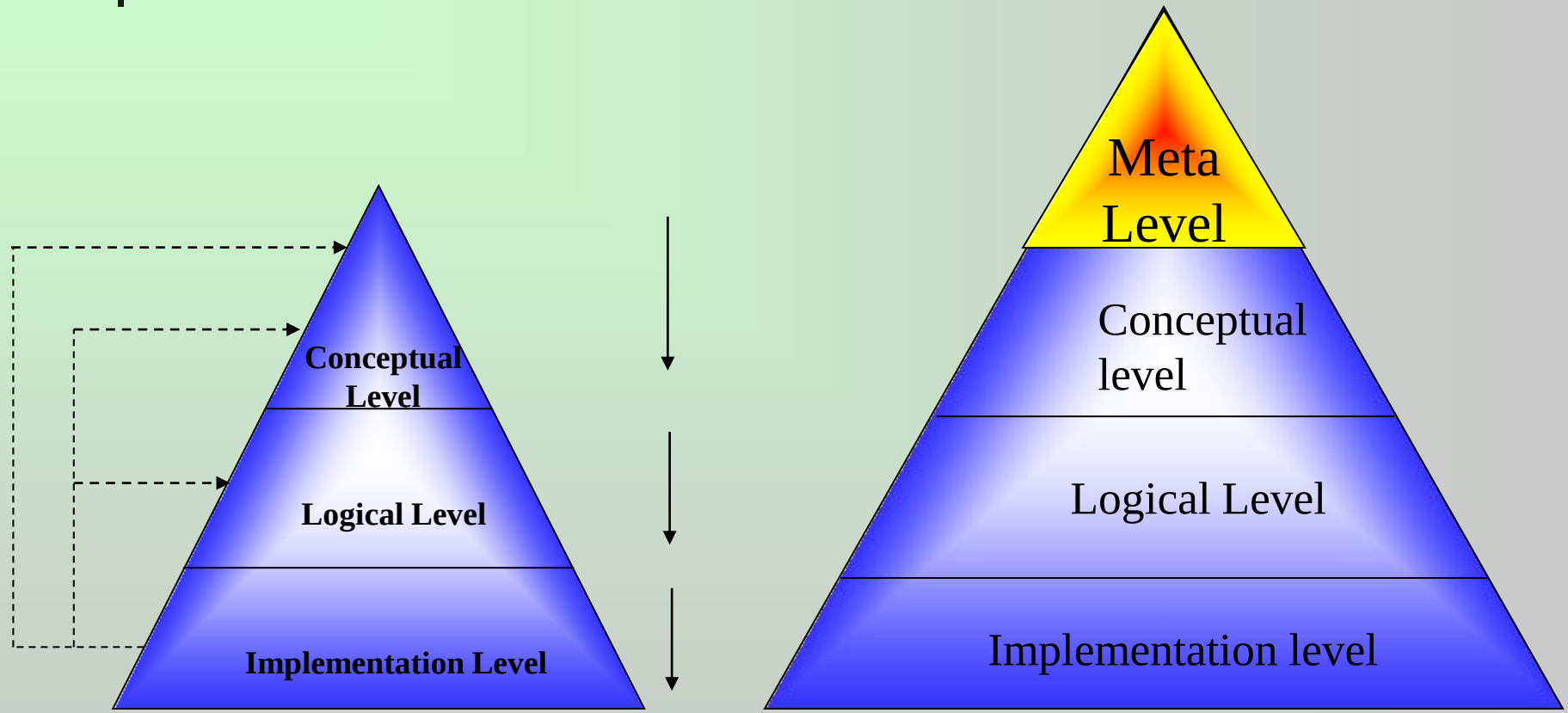
- Meta execution models are specific to conceptualizing and representing the execution logic for executing processes.
  - It is the specification of the execution model as a process using the concepts of execution model.
- Constitutes rules and processes that **define**, **select**, **generate** and **govern** meta-models and meta-model instances (semi-) automatically.
- Drives one or more execution models (or constructing new execution models) for successful execution of an application.
- Example: Workflow Execution
  - There is a workflow engine that executes the workflow
  - So the specification of a workflow engine (that is, the steps taken by a workflow engine) as a workflow gives a meta execution workflow.

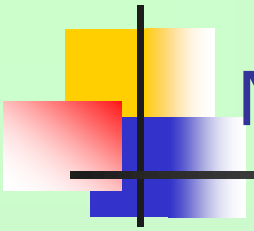
# Meta Models and Meta Modeling (e-contract example)

# Application Modeling



# Complex Application Modeling



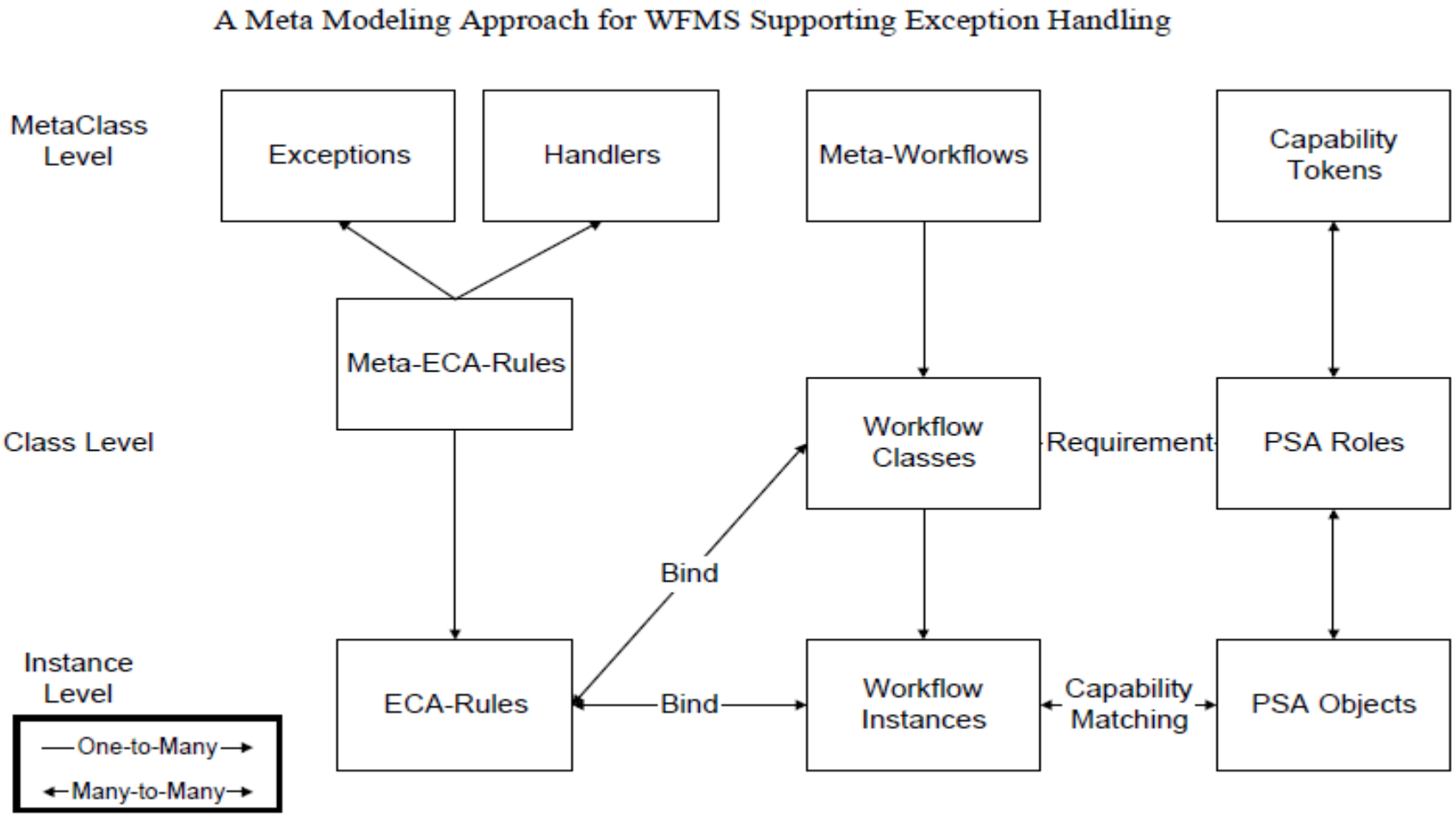


# Need for Meta-Model

- Most of the applications (ex., contracts) have similar structure (like clauses related to payments)
- A pre-cursor to conceptual modeling

[Chiu et al, 1999]

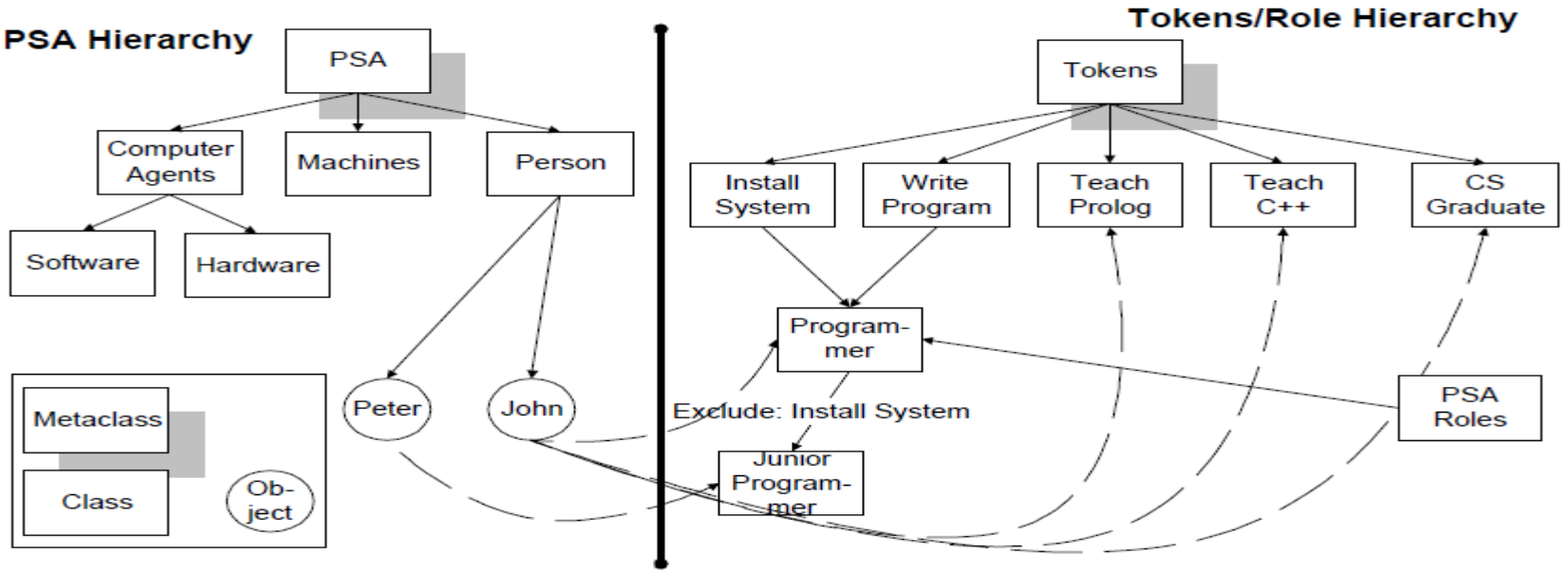
# Meta Model approach for Exception Handling



[Chiu et al, 1999]

# Meta Model Approach for Exception Handling

Meta-class and classes for PSAs




Meta Level declarations for PSA-roles and tokens

```
Role AMS_role
Class_attributes
    Role_Description: string;
    Date_Created: date;
Attributes:
    Date_played: date;
End

Role PSA_Role isa AMS_role
   /* noble role of all PSA */
played_by PSA
attributes
...
end
```

```
role Token isa AMS_role
    /* root class for different Tokens */
played by PSA
 class_attributes
   Exclude_Tokens: set of Token;
   Reverse_derivation: boolean; (default false)
   ...
methods  ...
  /* other methods and attributes to be declared
by the user */
end
```

# E-contract

A contract is a legal agreement involving parties, activities, clauses and payments, which stipulates that the involved parties agree to fulfill specified activities.

An e-contract is a contract specified, modeled and executed by a software system.

Major elements of an e-contract

- **Parties** play different roles in a contract and perform activities of a contract.
- **Activities** : Represent the tasks and e-services that have to be executed during the business process enactment.
- **Clauses** : describes restrictions/ conditions on the execution of activities
- **Exceptions**: Arise due to incomplete or partial fulfillment of clauses.
- **Commitments**: It includes completion of the activities by the parties as per the agreed terms and conditions and successful closure of e-contract enactment
- **Payments**

# Parties

23. LICENSOR shall refer to the President of India acting through any authorised person, who granted Licence under Section 4 of Indian Telegraph Act 1885 and Indian Wireless Telegraphy Act 1933, unless otherwise specified.

22. LICENSEE: A registered Indian Company that has been awarded licence for providing the SERVICE.

38. SUBSCRIBER- Subscriber means any person or legal entity who avails the service from the licensee.

# Activities

1.1 The LICENSEE shall commission the Applicable Systems within 24 months from the effective date of the licence and offer the service on demand to its customers.

1.2 LICENSEE shall be solely responsible for the installation, networking and operation of necessary equipment and systems, treatment of the subscribers' complaints, issue of bills to its subscribers, attending to claims and damages arising out of his operation. The LICENSEE shall make its own arrangements for all infrastructures involved in providing the SERVICE.

# Clauses

1.8 The billing disputes or differences, between the LICENSEE and its subscribers will be settled amongst themselves.

1.9. MTTR (Mean Time To Restore):

1.9.1. 90% of faults resulting due to subscribers complaints should be rectified within 24 hours and 99% within 3 days.

1.9.2 The Licensee will keep a record of number of faults and rectification reports in respect of each service area and produce the same to the Authority as and when required.

# Exceptions

(b) if the LICENSEE fails to perform any other obligation(s) under the Licence including remittance of timely payments of Licence fee due to the LICENSOR and the LICENSEE does not rectify the failure within a notice period of 30 days or during such further period, as the LICENSOR may authorise in writing in this regard.

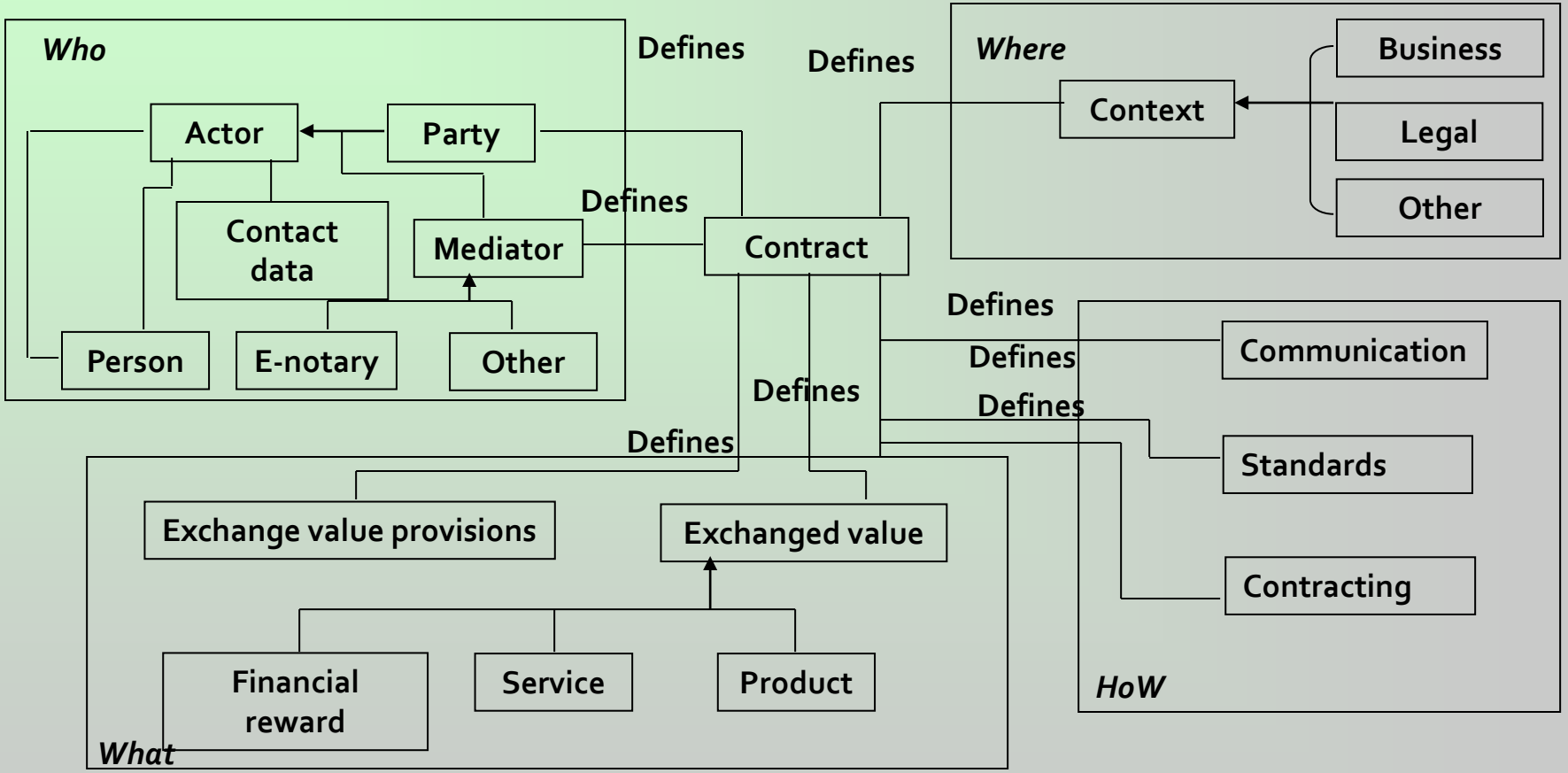
10.3 TERMINATION FOR CONVENIENCE:

If the LICENSEE desires to surrender the licence, it shall give an advance notice of 30 days to the Licensor to this effect. If the service is in operation, the licensee shall also intimate its subscribers of consequential withdrawal of service by serving a 15 days notice to them. The financial liability of the licensee company for termination of the licence for convenience shall be as below:-

(a) After start of service:- If during the notice period, acceptable level of service is not delivered to the customer, the licensee shall forfeit all claims on the Performance Bank Guarantee which shall be encashed and the amount shall be adjusted towards damages.

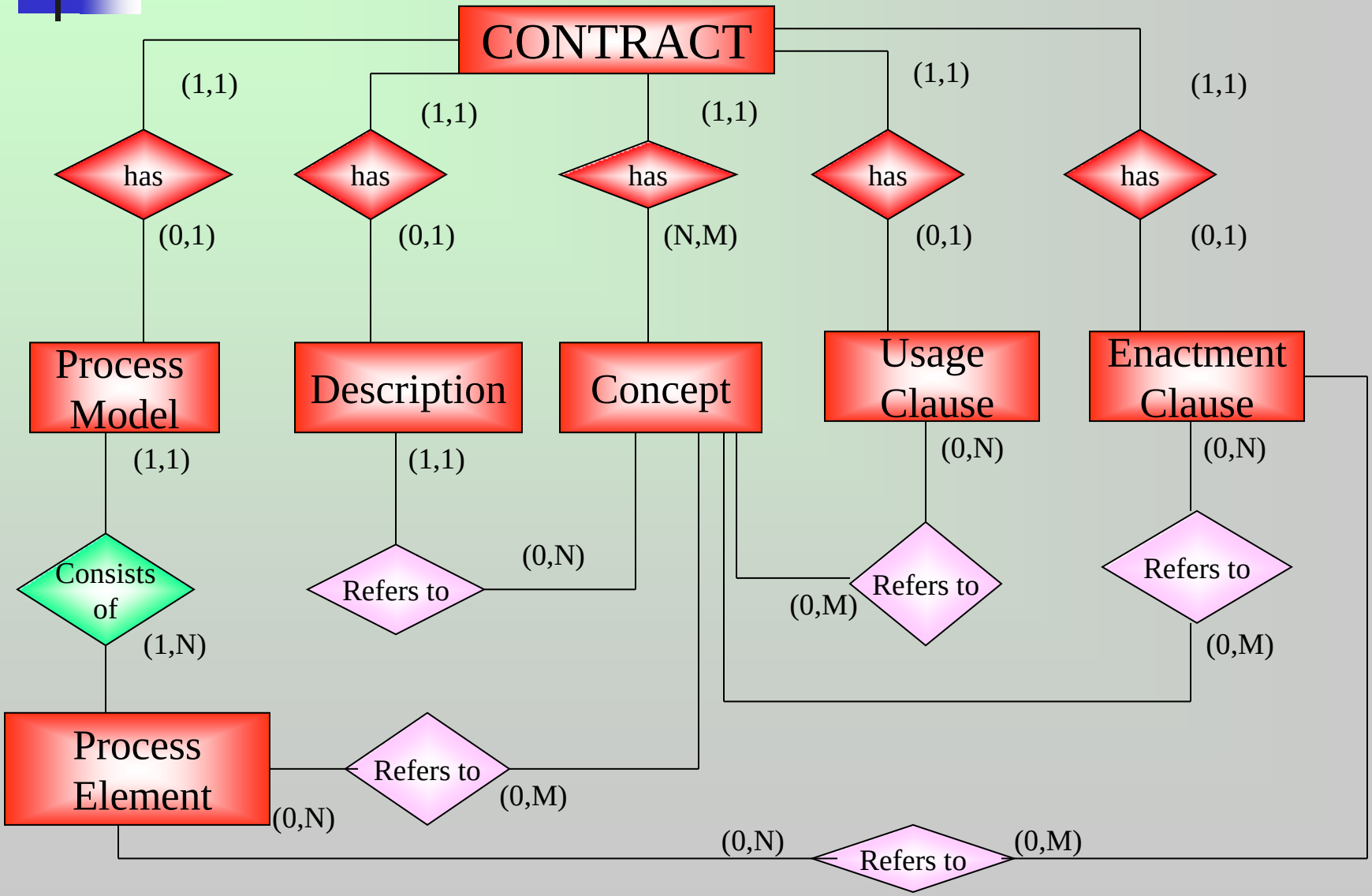
# Modeling E-contracts

# 4W E-contract Model



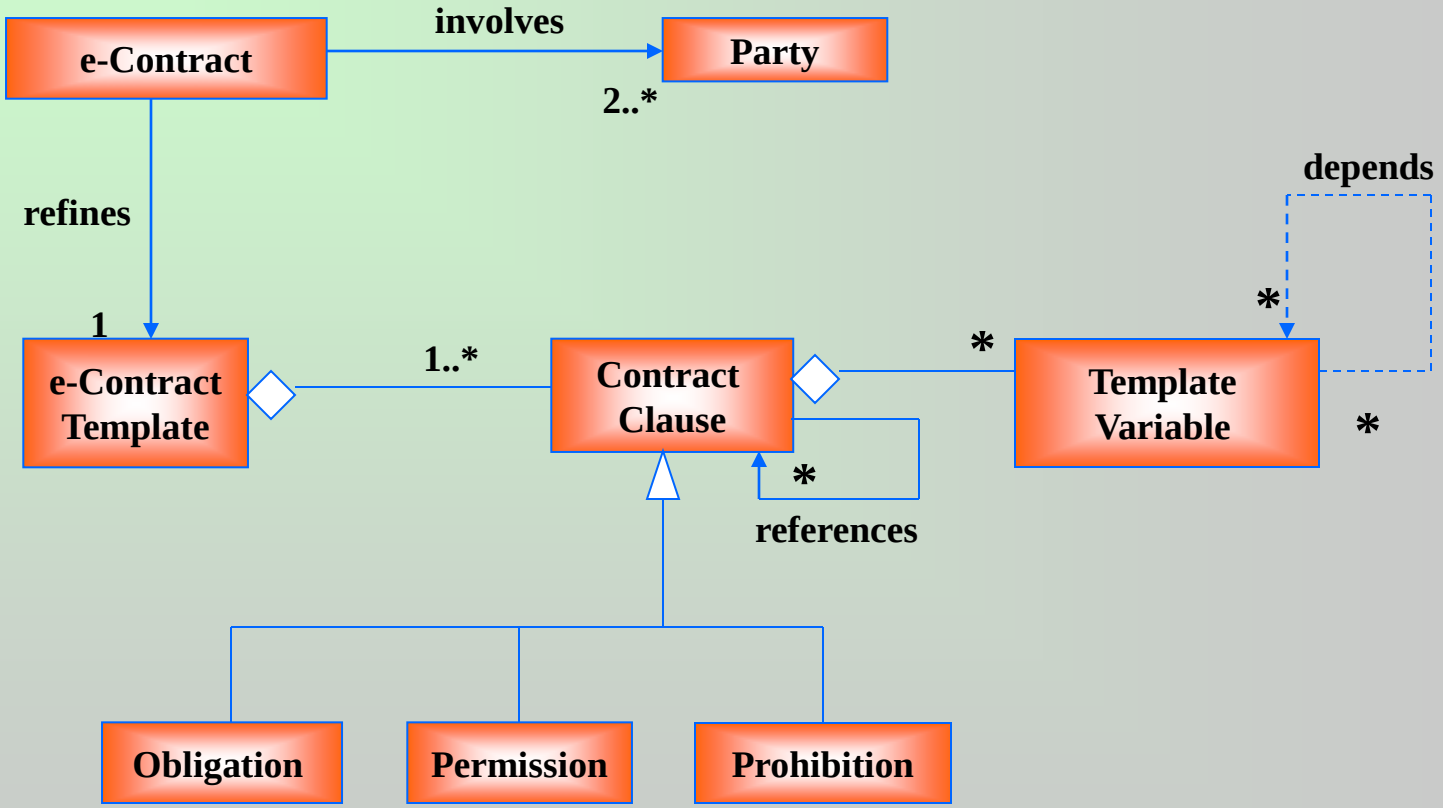
# CrossFlow e-contract meta-model

[Grefen et al]



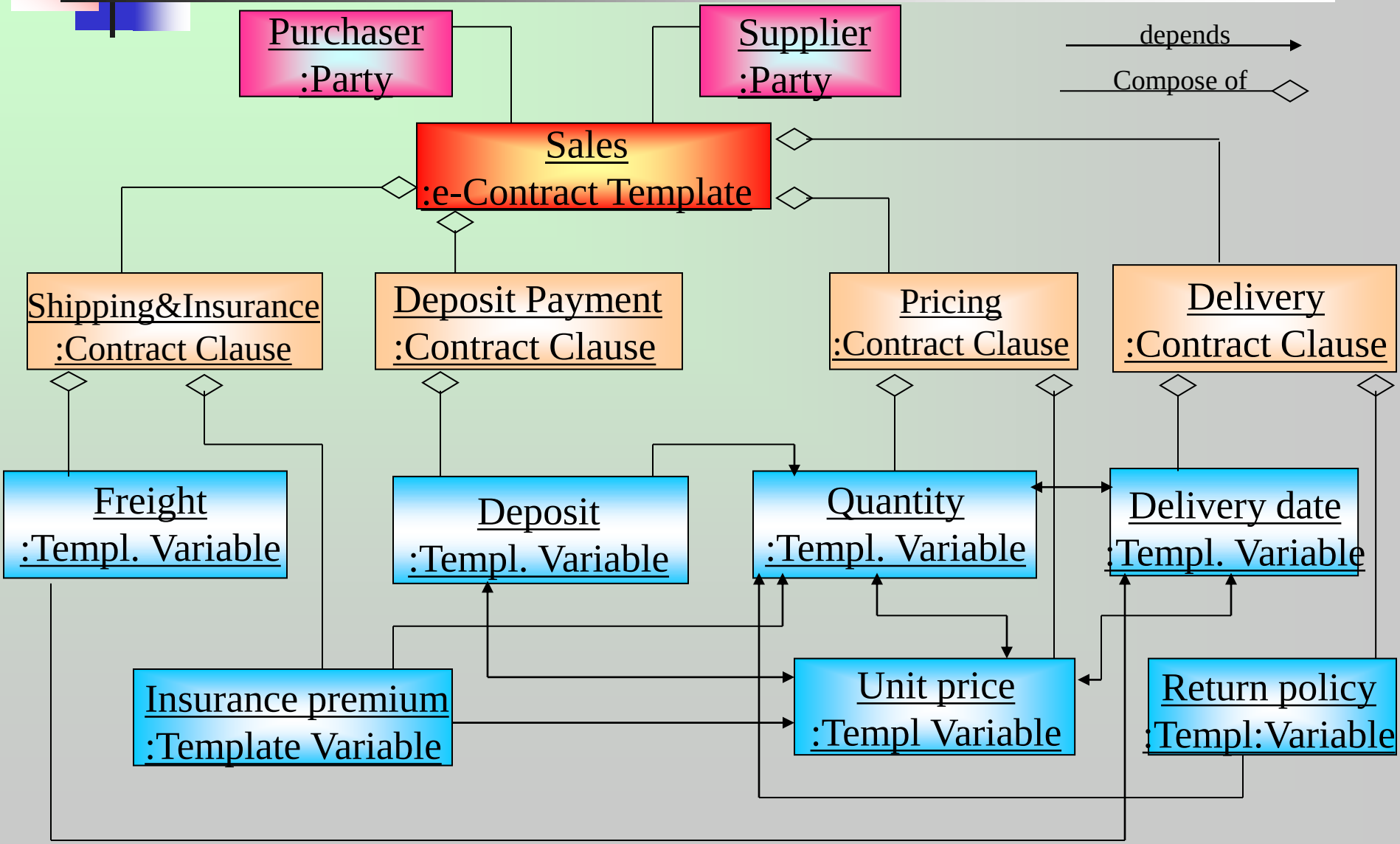
# Meta –Model for e-Contract template

[Chiu et al, 2005]

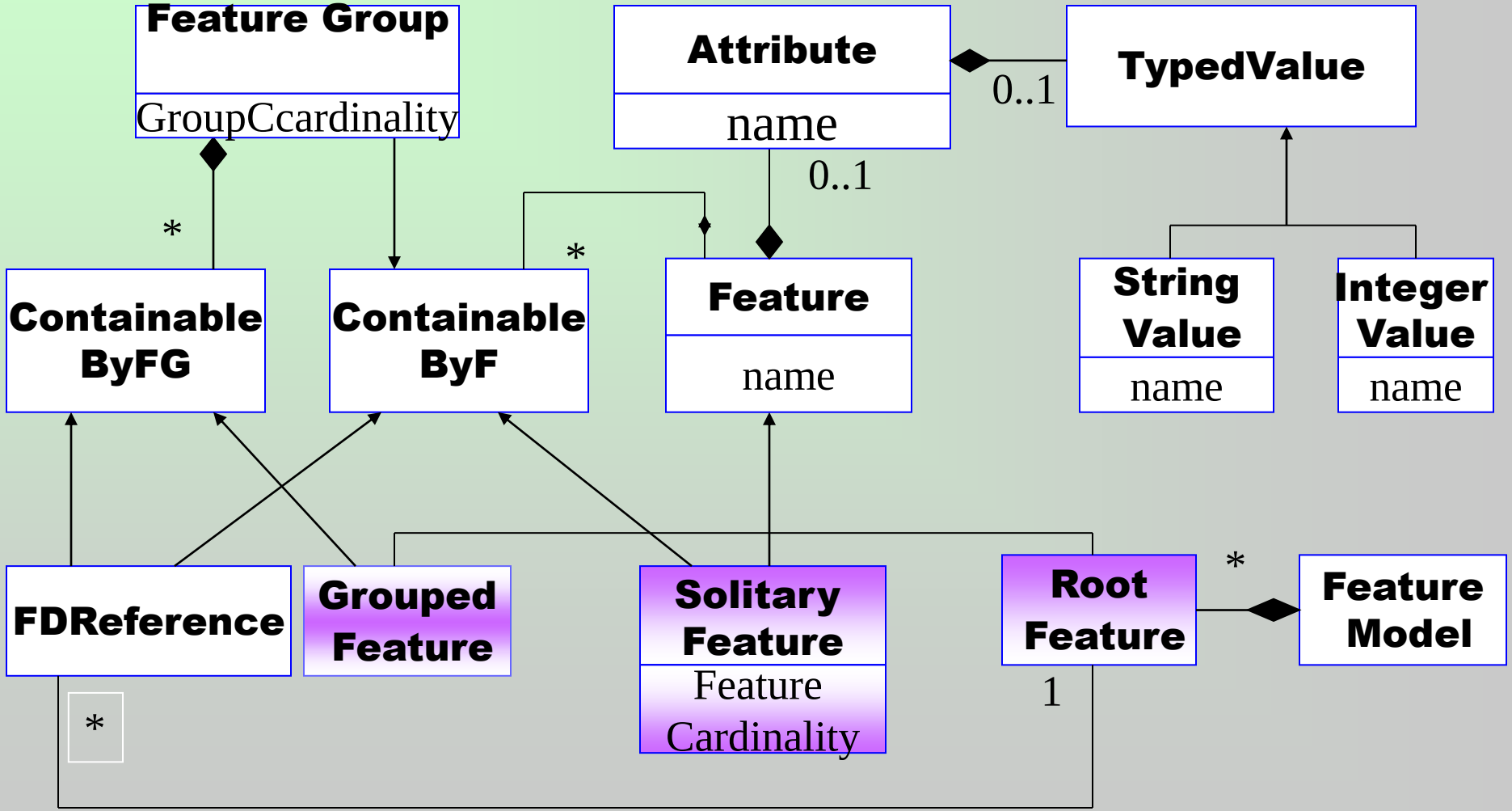


Chiu et al, 2005

# A sales e-Contract template as an instance of the meta-model



# Feature Meta-model [Fantinato et al, 2006]



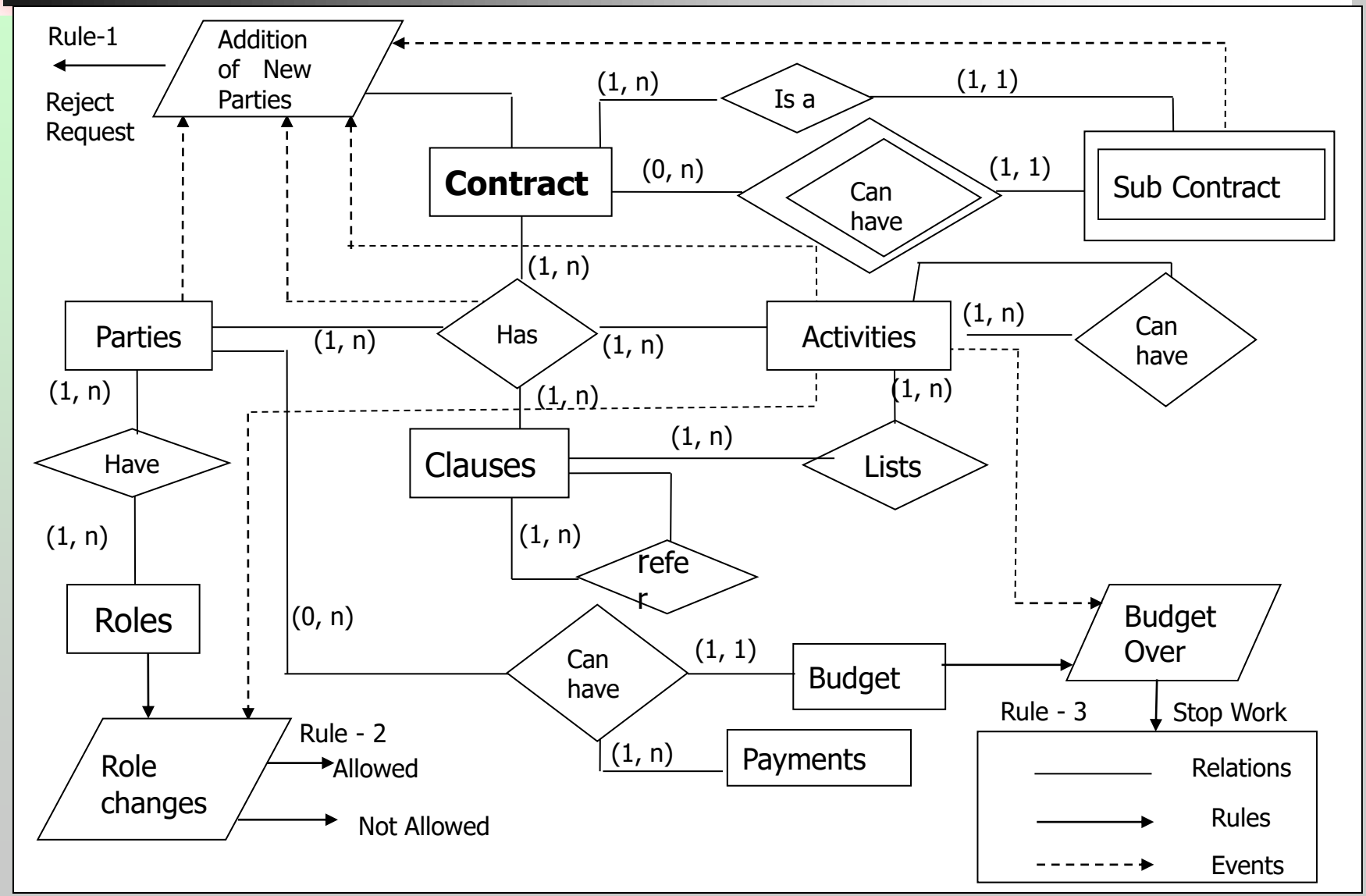
# Conceptual Modeling of E-Contracts

- Complexity in modeling E-contracts
  - Voluminous documents
  - Complex inter-relationships among activities, clauses, etc.
  - Determining which particular exception clause should be enforced when a violation is detected.
  - Ambiguity and fuzziness of natural language statements
  - Actual parties' activities may not be known during signing of the contract
  - Domain specific terminology and regulatory compliance
  - Autonomous nature of individual organizations/parties
  - Involvement of multiple sub-contracts at the time of enactment as well as during enactment

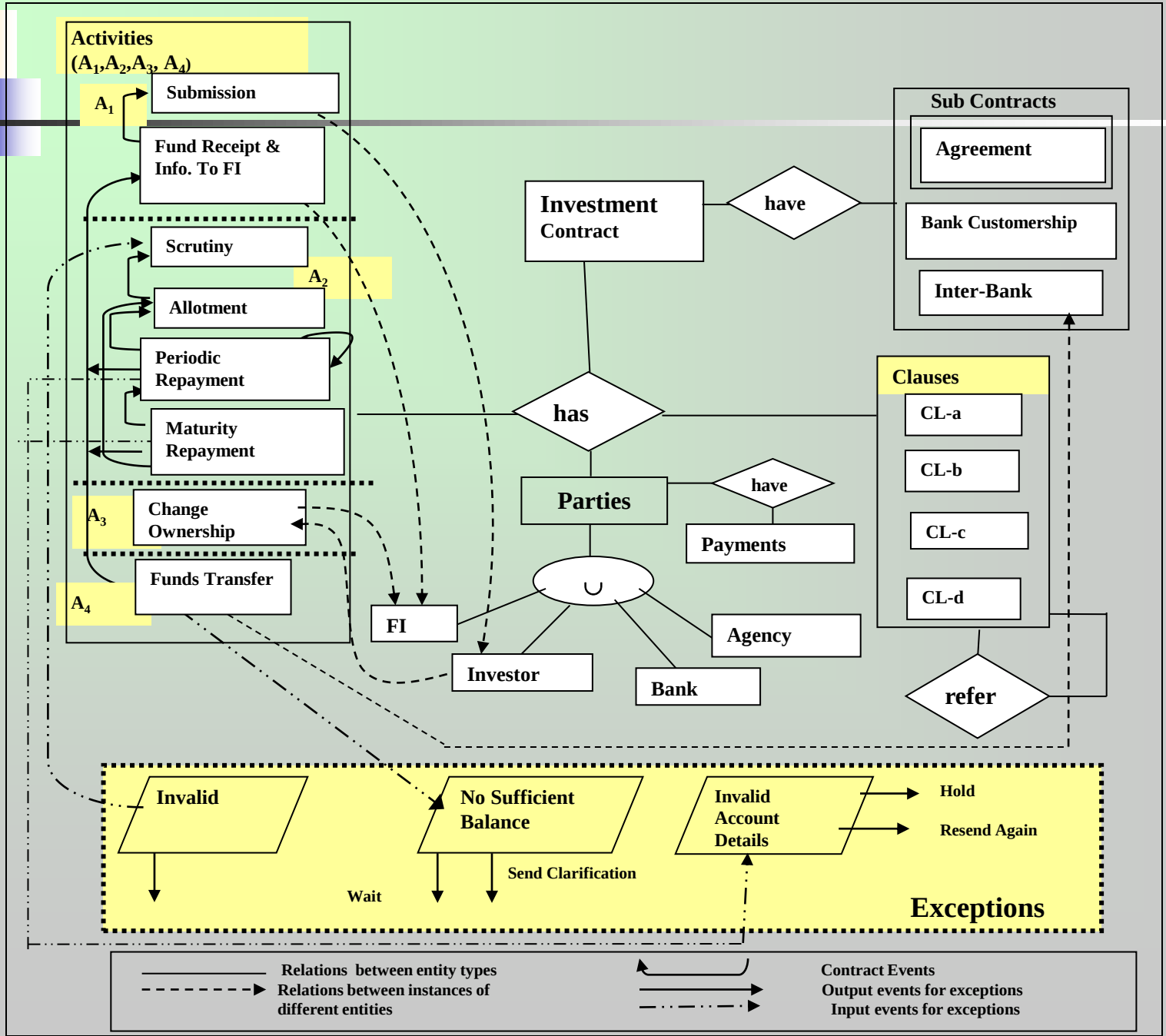
# Conceptual Modeling of E-Contracts

- Need for Meta-Model
  - Most of the contracts have similar structure (like clauses related to payments)
  - Guided approach to conceptual modeling
  - Templates can be designed for specific domains
  - Provides generality and flexibility
  - Allows reusability and extensibility

# An ER$^{EC}$ Meta Model for E-Contract



# ER[EC] Model Instance: Investment Contract



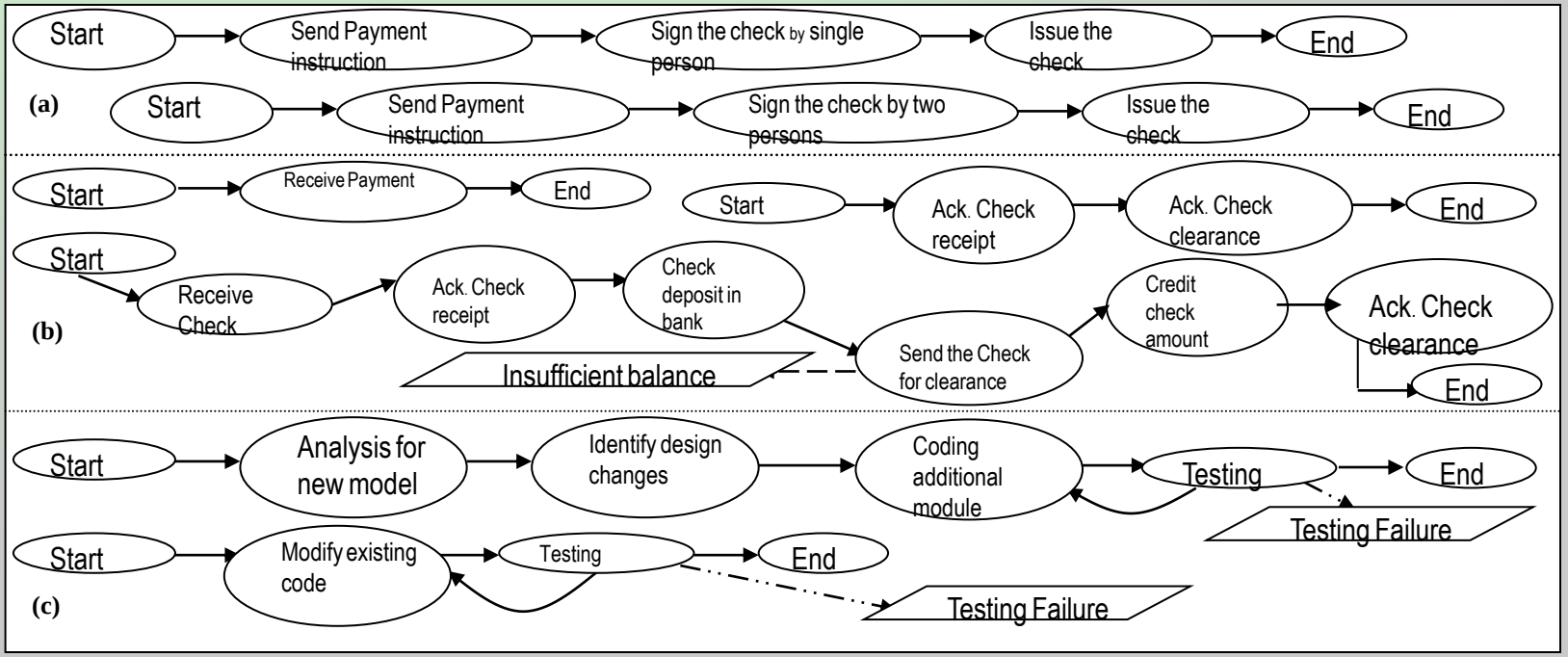
- e-contracts cover a vast amount of business related processing in the world  and our aim is to streamline it and support it using meta models and meta execution models.
  - e-contracts can be streamlined by using workflows.

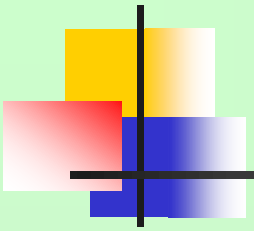
# Mapping E-contracts to Workflows

ER$^{EC}$ model $\rightarrow$ Workflows

1. All parties are mapped to agent types/roles.
2. Activities to workflows and activities in a workflow.
3. Contracts to events that occur.
4. Clauses to conditions that need to be satisfied.
5. Exception handling to additional activities in a workflow.
6. Payments and contracts to documents and additional input/output events.

## Parametric workflows, Workflow Views, Dynamic Workflows





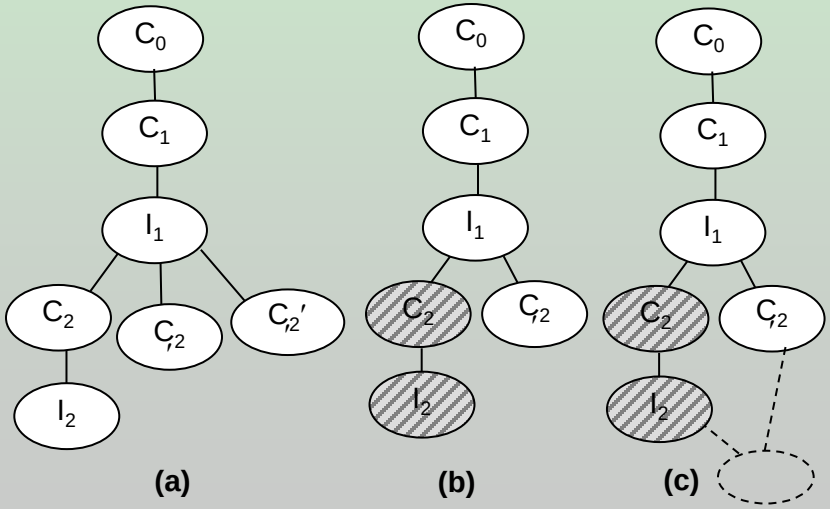
# $ER^{EC}$ Model

- Forms a template
  - $ER^{EC}$ model defines a template for a specific contract
  - A template is an instance of a meta-model with constraints for a specific purpose.
  - Models data and process/functional requirements of an application
- Captures Low level relationships
- Enable modeling events and exceptions
- Additional flexibility by customization
- Mix of entity types and entities
- Standardized model
- Deployable workflows for e-contracts
  - Mapping from modeling constructs to workflow entities
  - Requirement of on-the-fly generation of workflows
    - parametric workflows, workflow views and dynamic workflows

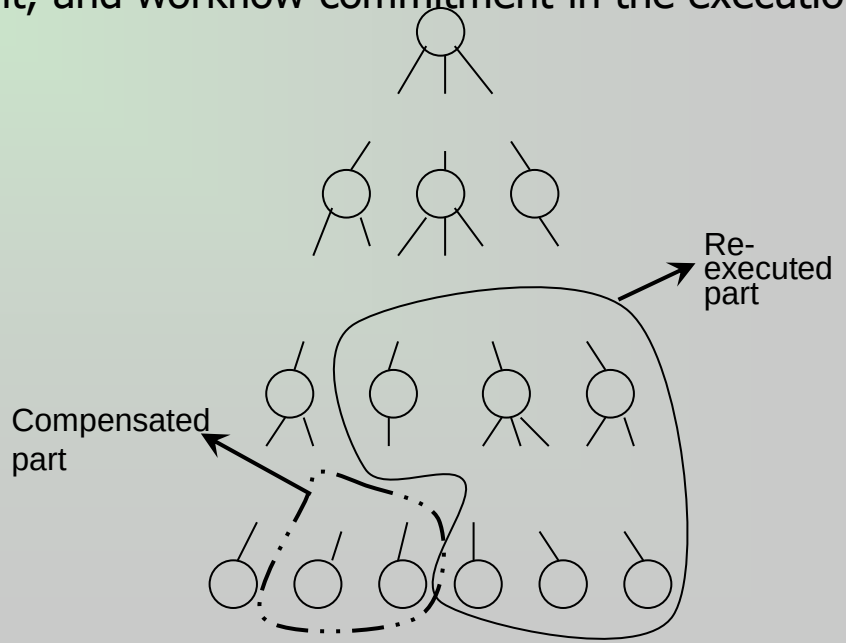
**Enable Visual Representation to Conceptualize e-contracts**

# Activity Commitments

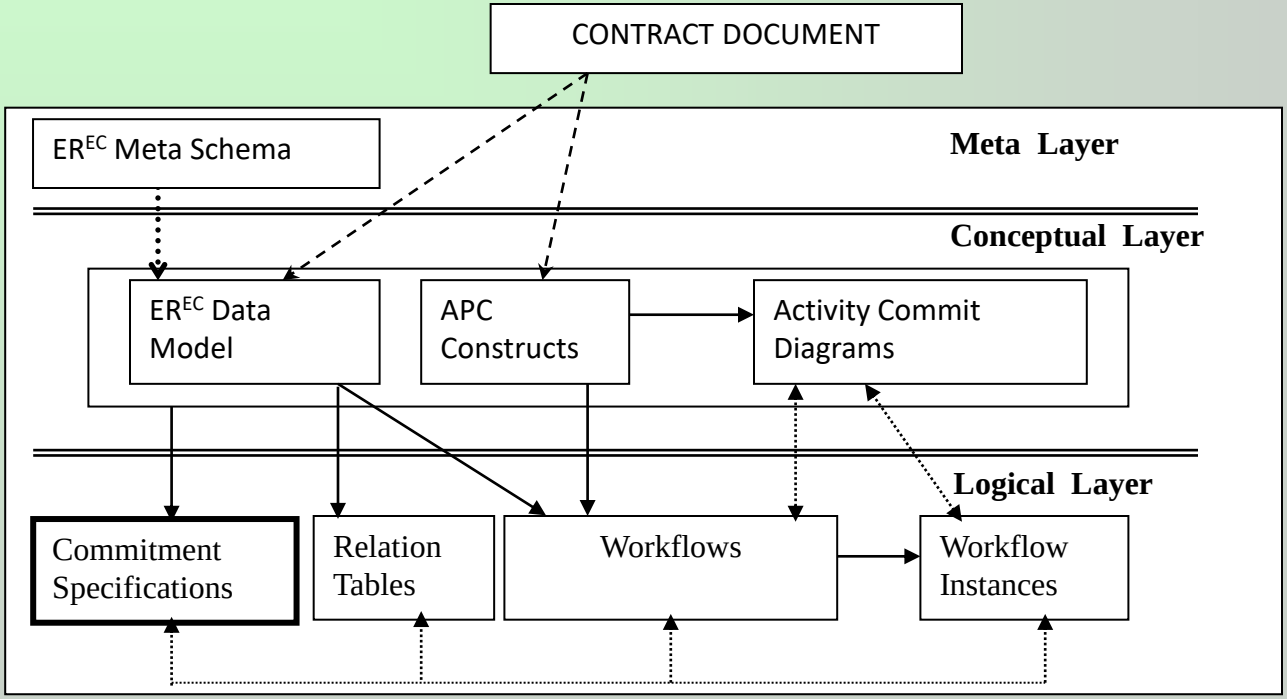
- Meta Model enable commitment specifications for e-contracts
  - activity-commitment semantics at conceptual level are specified based on the contract document.
  - However, during the specification of these semantics, the actual execution level commitments are not known a priori.
  - At logical level, the commitment specifications facilitate specifying the semantics for transactional support, activity commitment, and workflow commitment in the execution of an e-contract.




(a) A composition, (b) An execution of the composition, (c) A closed c-tree for the execution-tree



# $ER^{EC}$ Business Process Model for specification and execution of e-contract enactment.



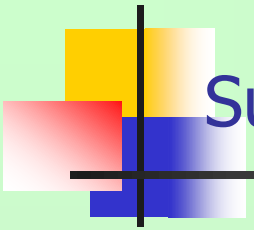

*Legend*:

Input

Process Flow

Monitoring & updation

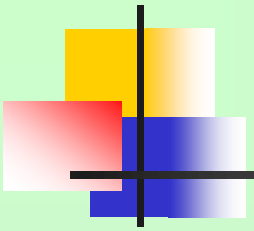
Instantiation



# Summary

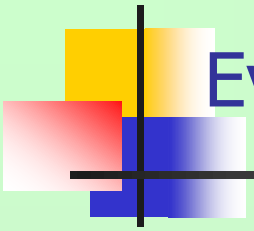
- Provides guided approach to conceptual modeling
- Templates can be designed for specific domains
  - Instances of a meta-model for a specific application domain (with certain constraints)
  - Guide the modeling and Execution processes
- Provides generality and flexibility
  - Support functionality for adapting a data model to new and changing requirements
- Allows reusability and extensibility
  - Addition of new constructs
  - Merging of two or more models/ model instances
- Ability to define meta-events and meta-ECA rules
- Sustainability of conceptual models for longer durations

# Modeling Evolving Applications



# Evolution Needs

- Evolution of Business Environments
- Changing Market Requirements
- Involvement of multiple organizations
- Competition
- Changes in Government Policies and Laws
- Advancements in Technologies

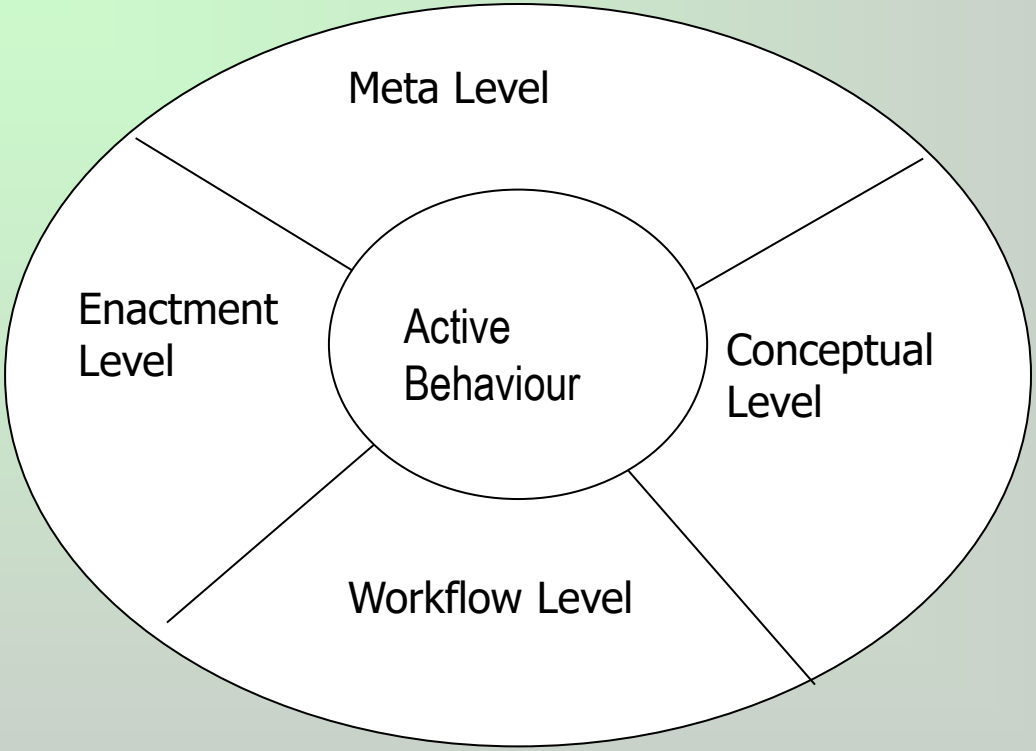


# Evolving Applications

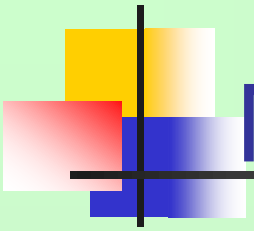
- Two Kinds of Changes
  - Run-Time changes
  - Mini-World changes
- Exceptions
  - Expected exceptions
  - Unexpected exceptions

need of active behavior to synchronize the changes in business logic and business processes across different levels of conceptual/logical models.

# Evolving Applications



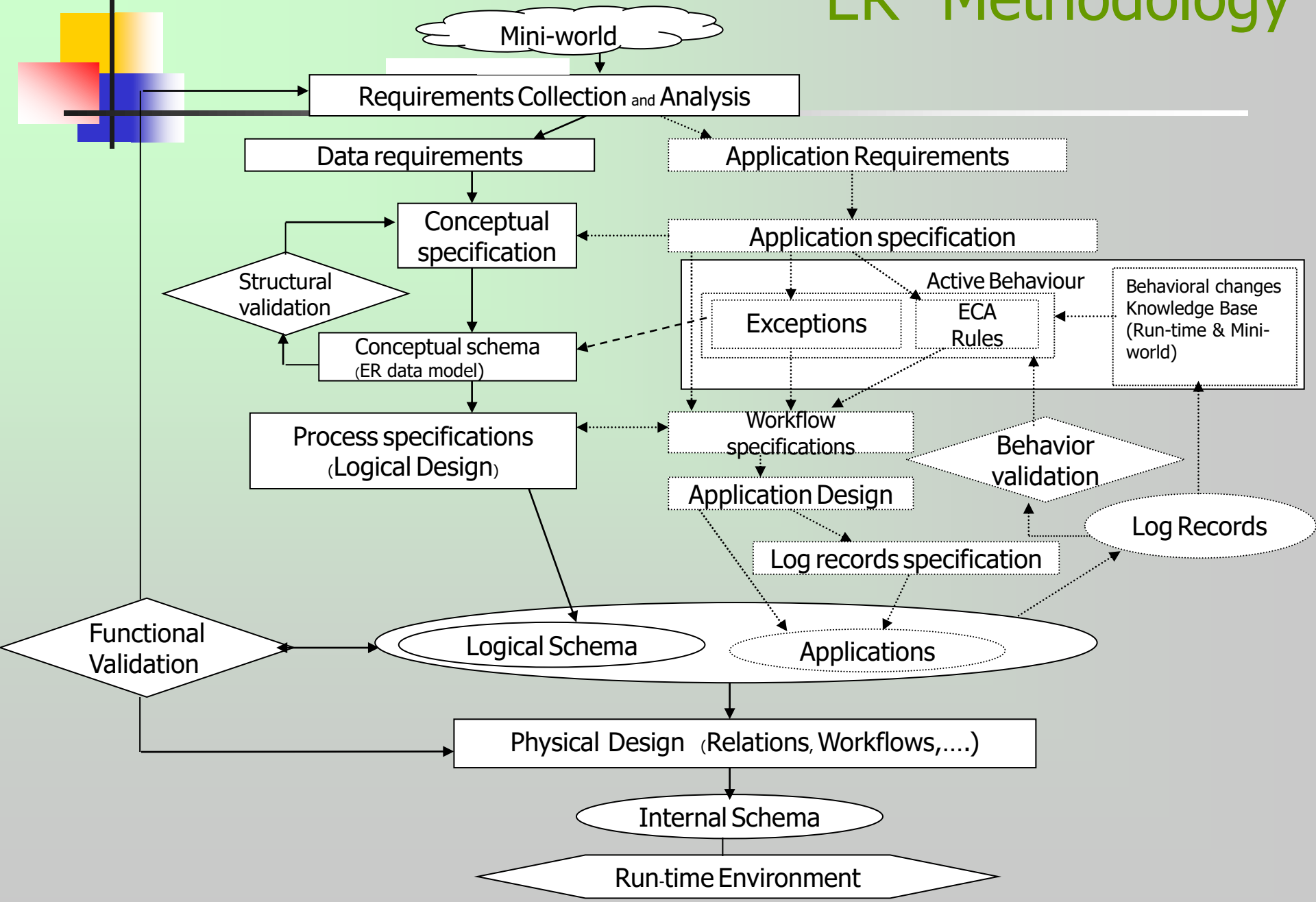

**Modeling active behaviour at various levels**



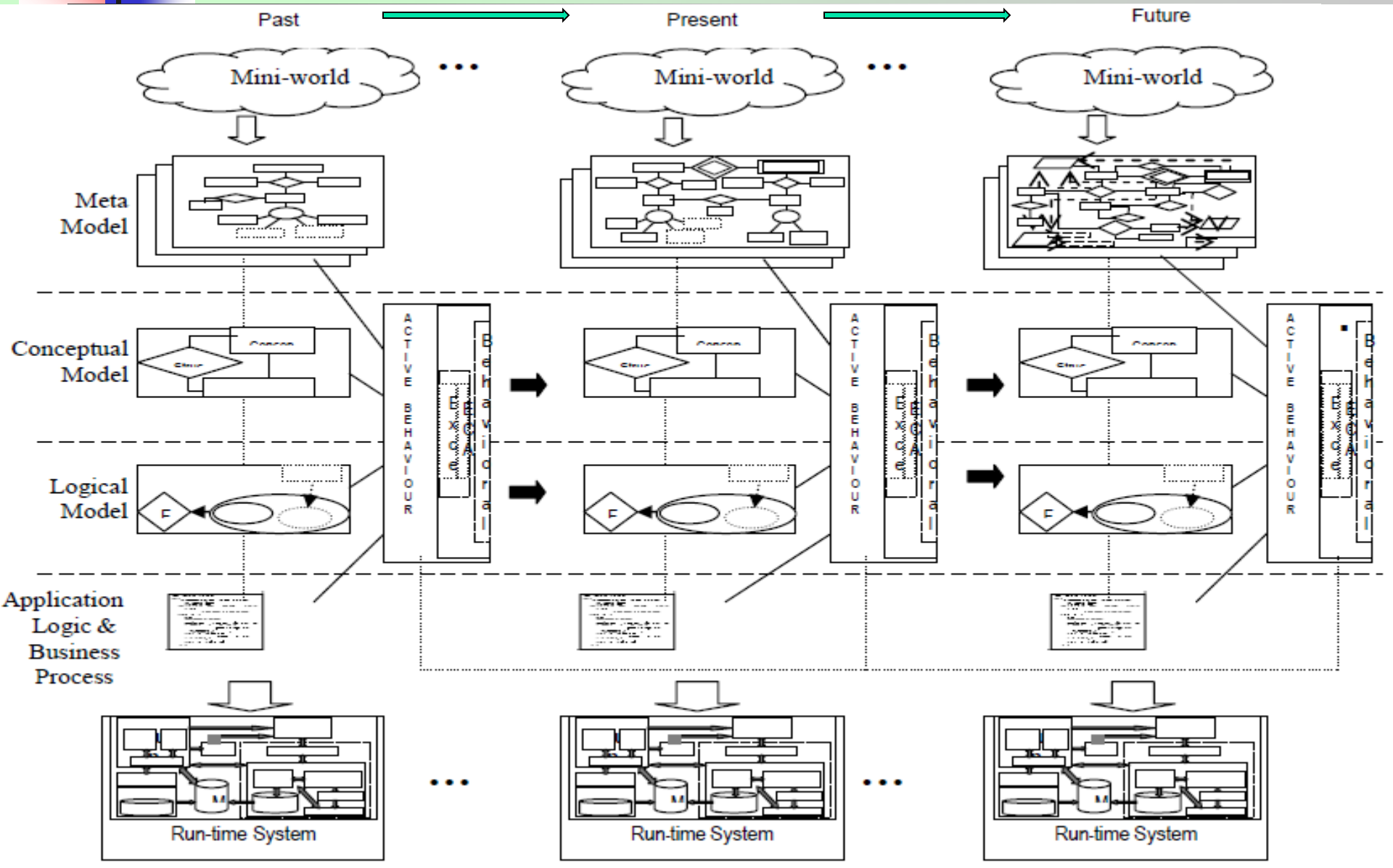
# Modeling Evolving Applications

- How to re-design the conceptual models (for instance, ER model)? How to synchronize the changes in mini-world and/or run-time environment to other levels?
- This calls for an iterative active methodology that constantly monitors run-time environment and changes in real-world specifications to keep the deployed applications/processes current.

# ER* Methodology



# Two way perspective of active conceptual models



# ER* Methodology for Evolving Applications

A two-way perspective of actively evolving conceptual models:

i) across the time domain (present, past and future)

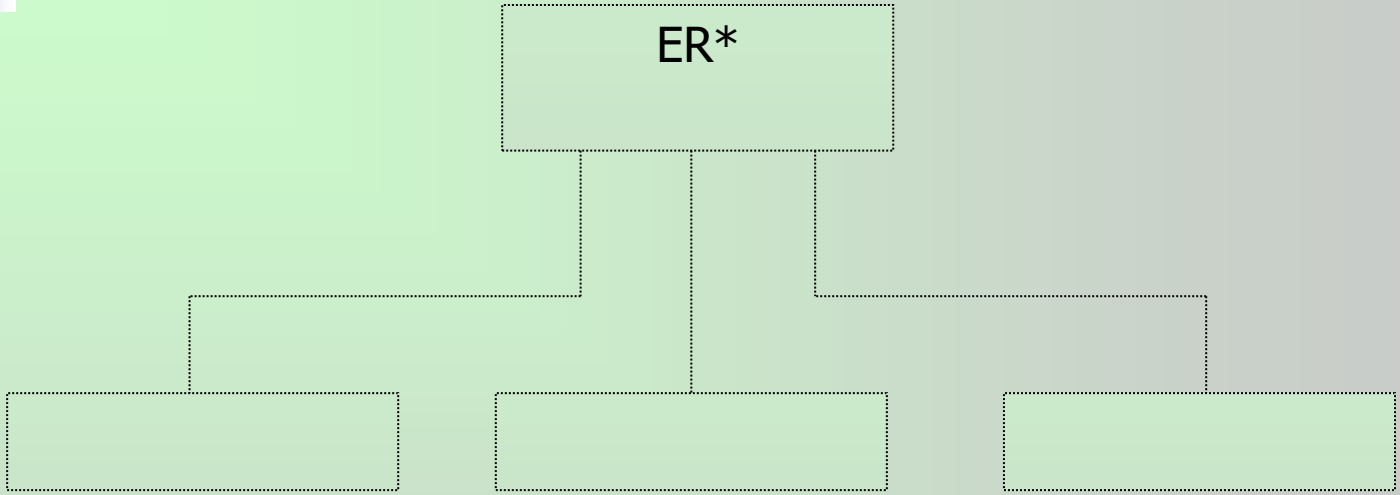
ii) at various levels (meta, conceptual, logical and application level).

Approaches for evolution from present to future

- Template selection
- Operator assisted evolution of ER models
- Complete re-design of ER models (from scratch)

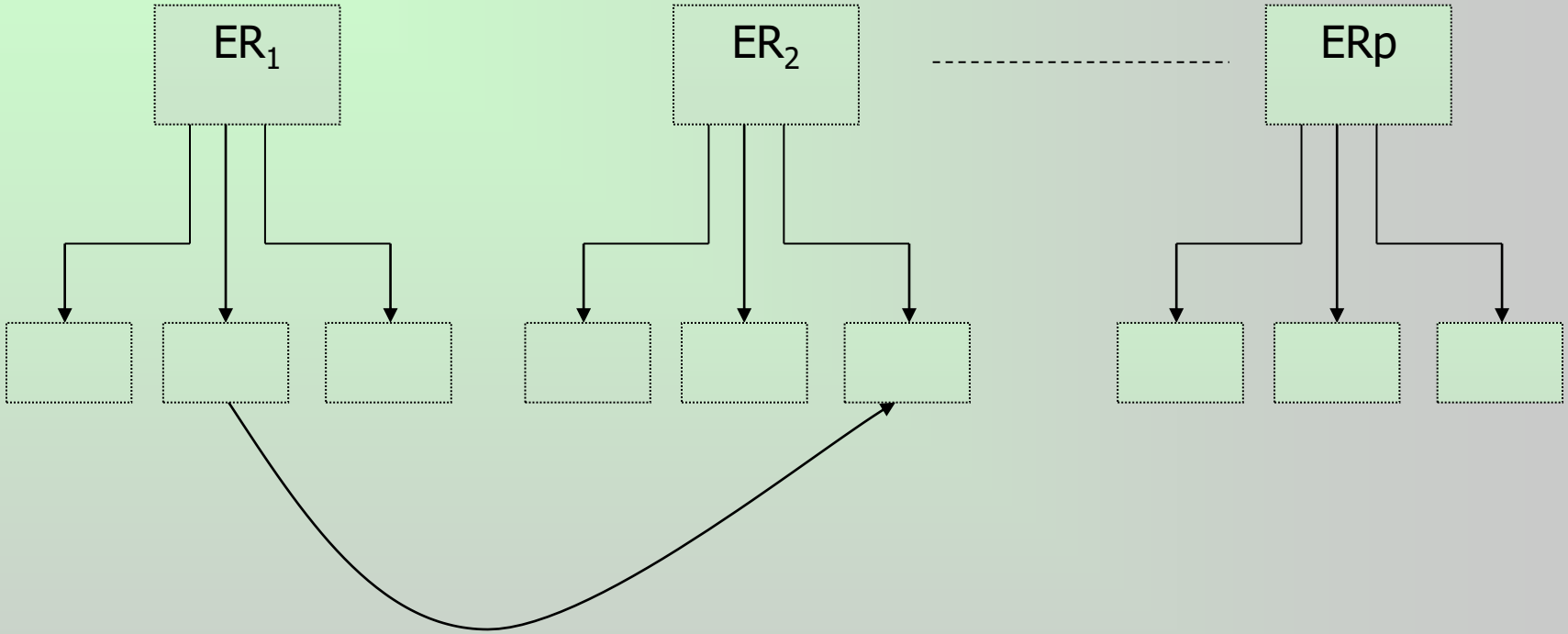
The template selection mechanism manifests itself as a ER* methodology problem.

# Approach 1: ER* Model Instantiation



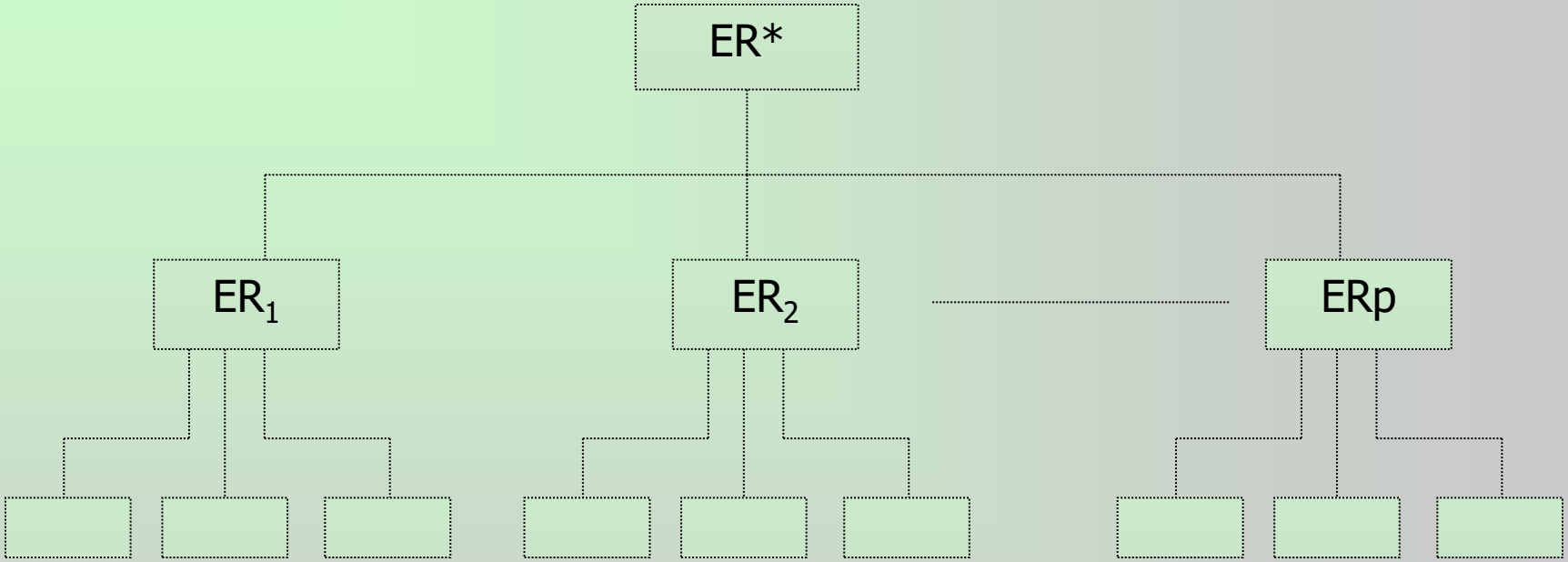

**An appropriate ER model is instantiated from ER* model and necessary modifications can be made on it depending on the revised scenario**

# Approach 2: Template instantiation from multiple ER models




**An application requires one or more additional template elements**

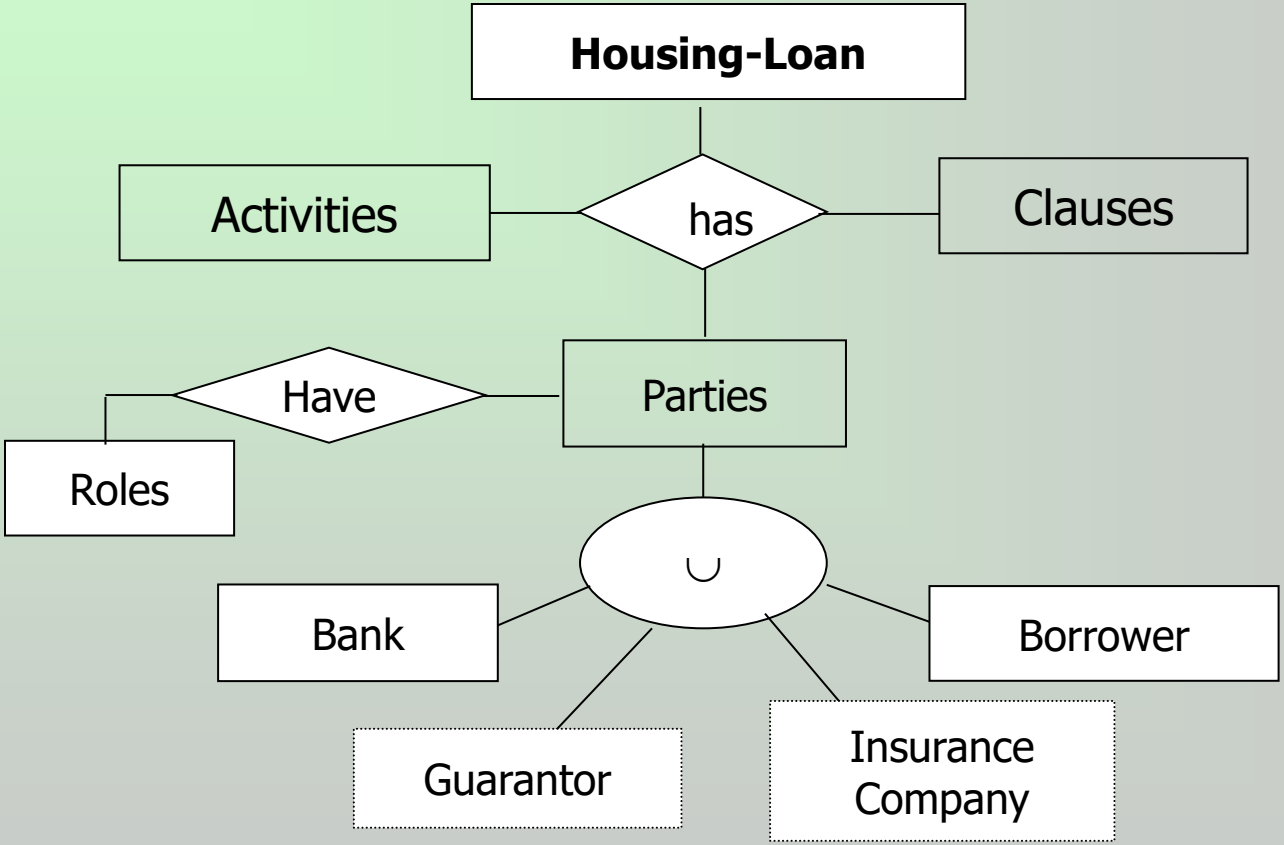
# Approach 3: Build new template



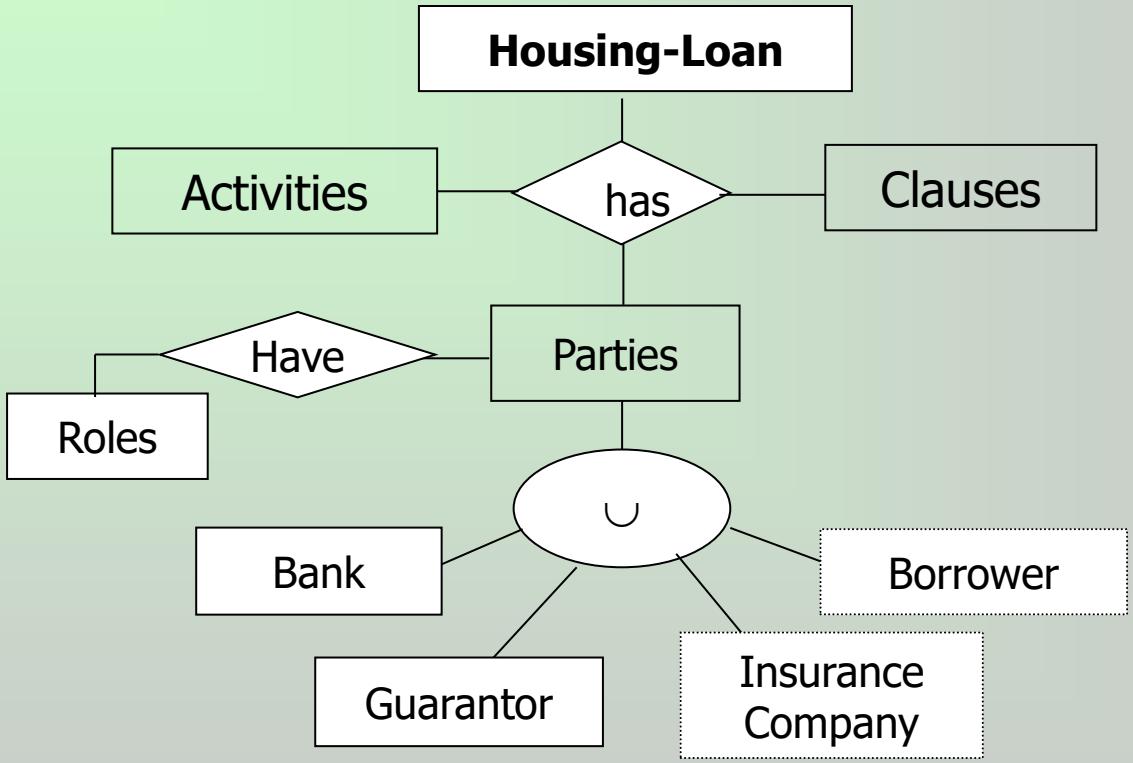

**The change could evolve the template itself**

# Example :
# Housing-Loan contract

# Standard template of Housing-Loan contract

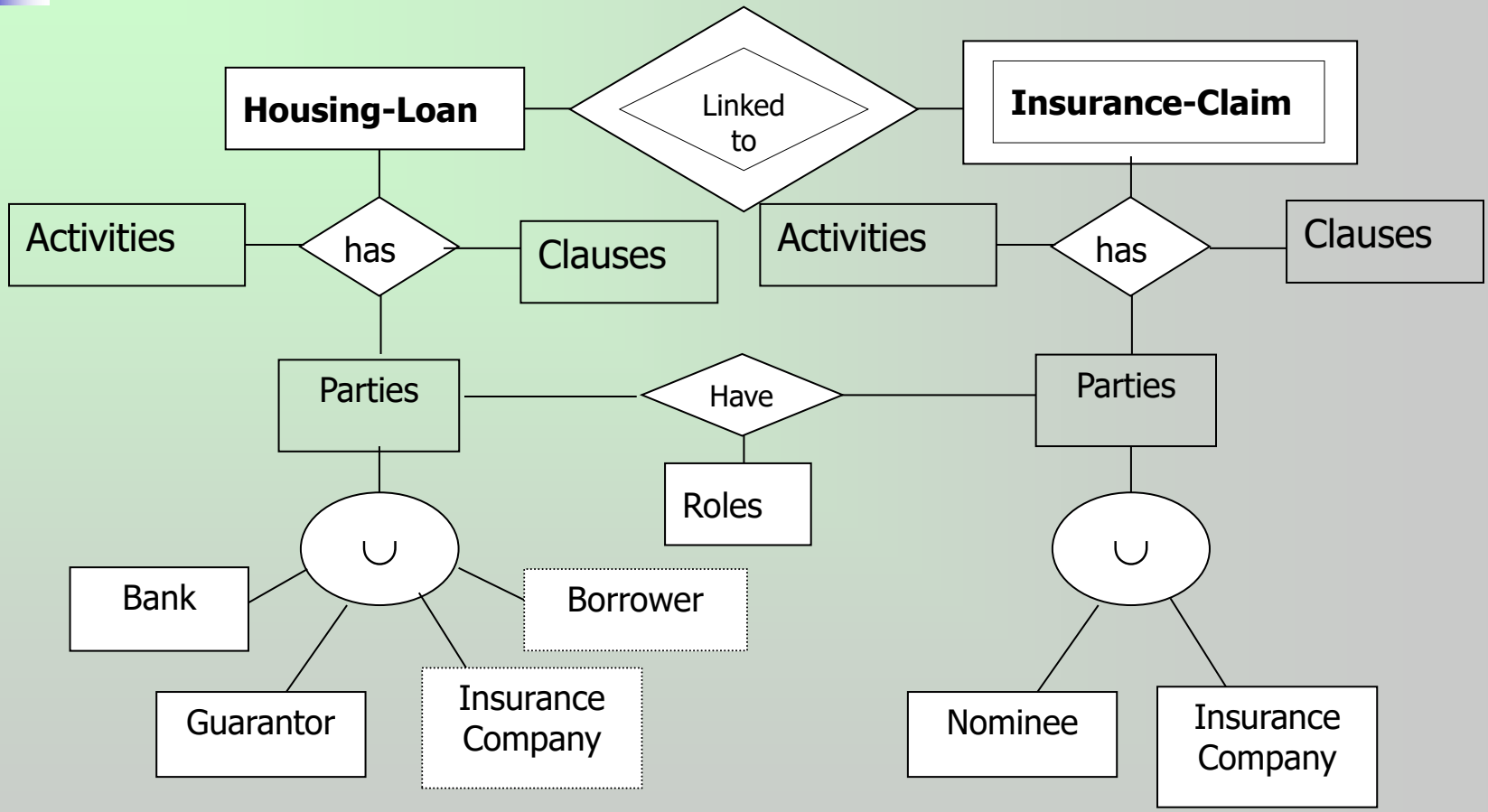


# Case 1: (Run-time change) - Borrower defaults



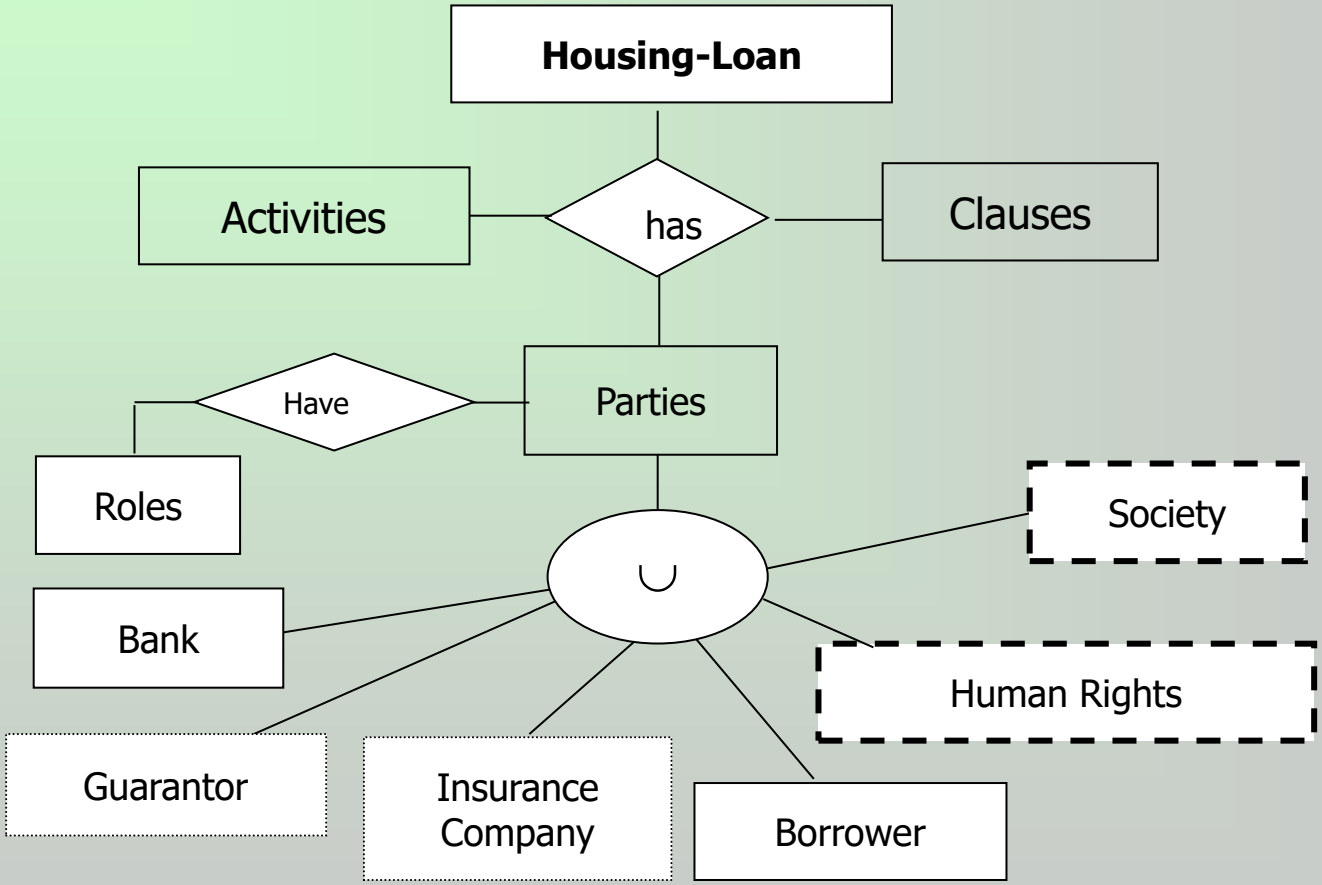

**Template with change of roles**

# Case 2: (Run-time change): Borrower's death/disablement

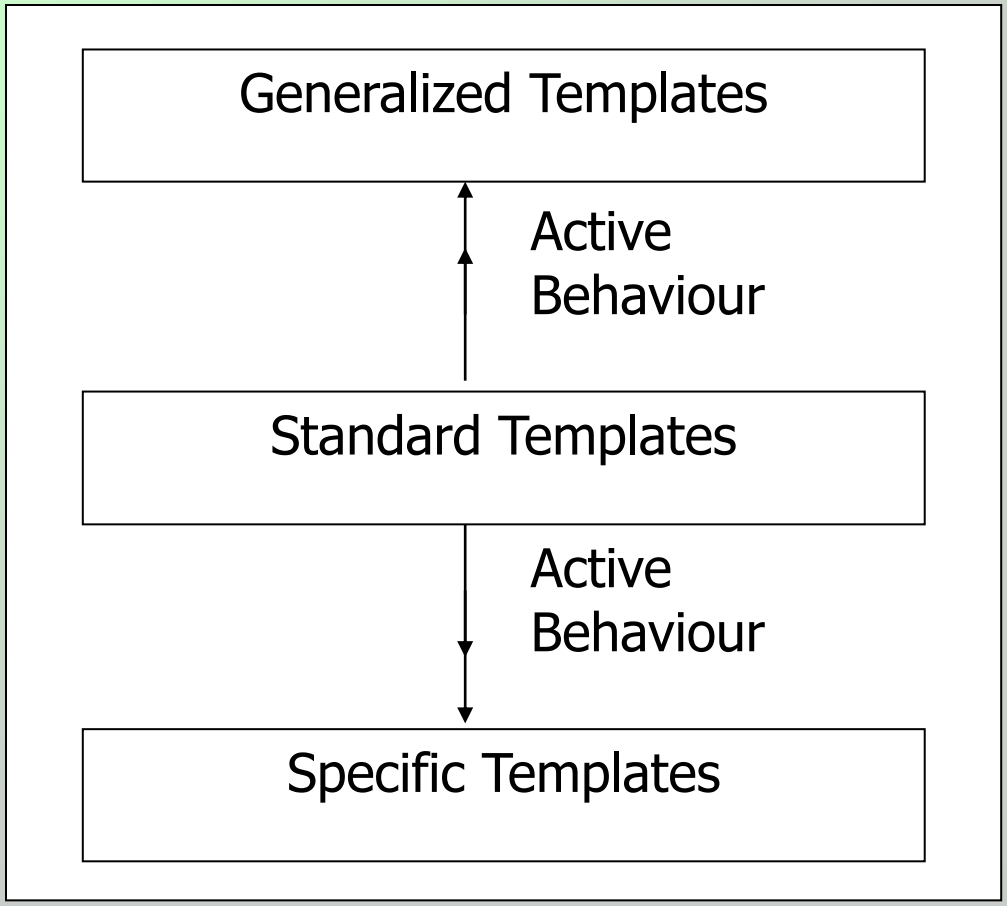



**Template with addition of subcontract**

# Case 3: (Mini-world change) - road expansion



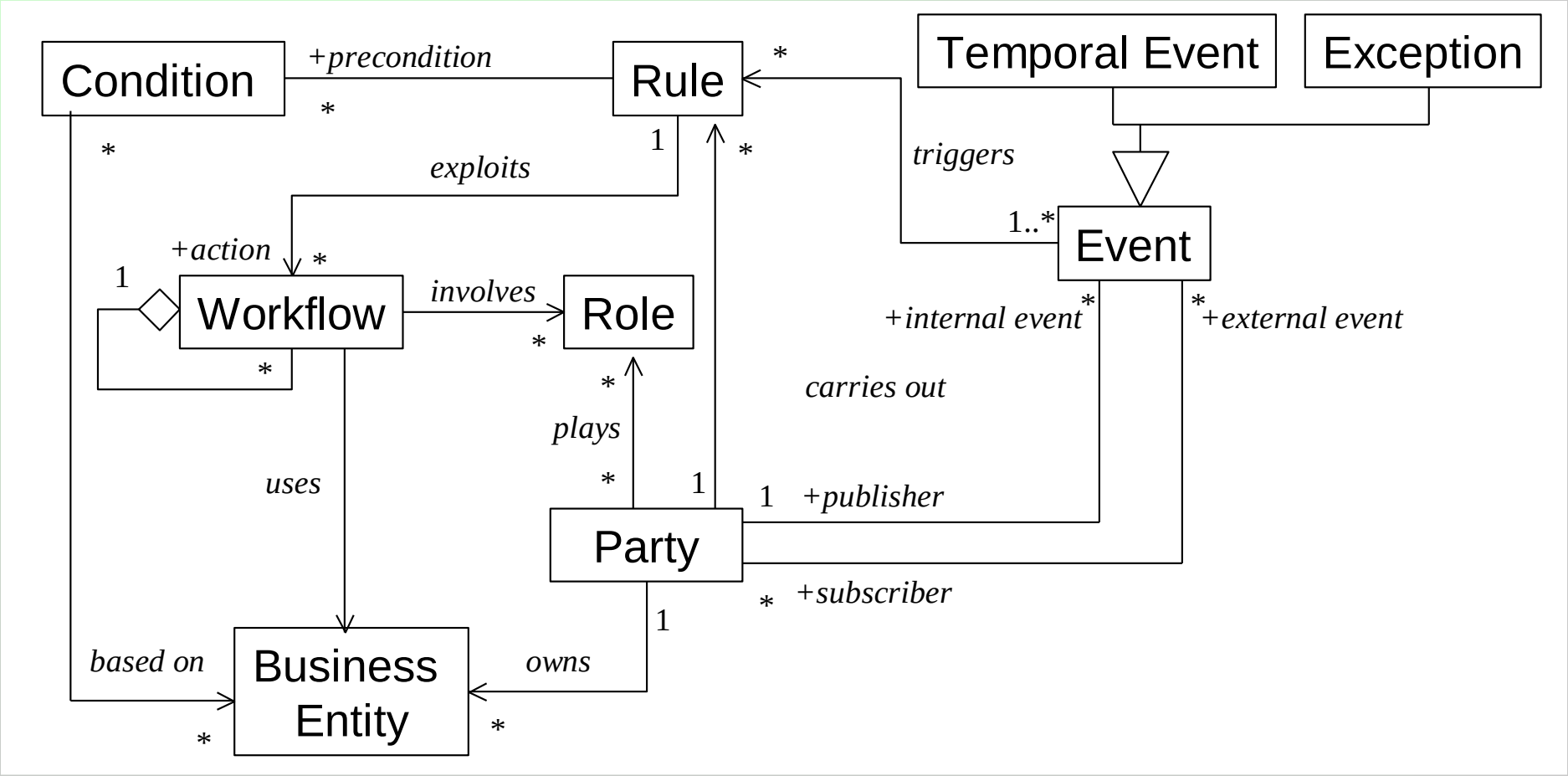

**Template with additional concepts**


Generalized Templates
Active
Behaviour
Standard Templates
Active
Behaviour
Specific Templates

# Taxonomy of operations

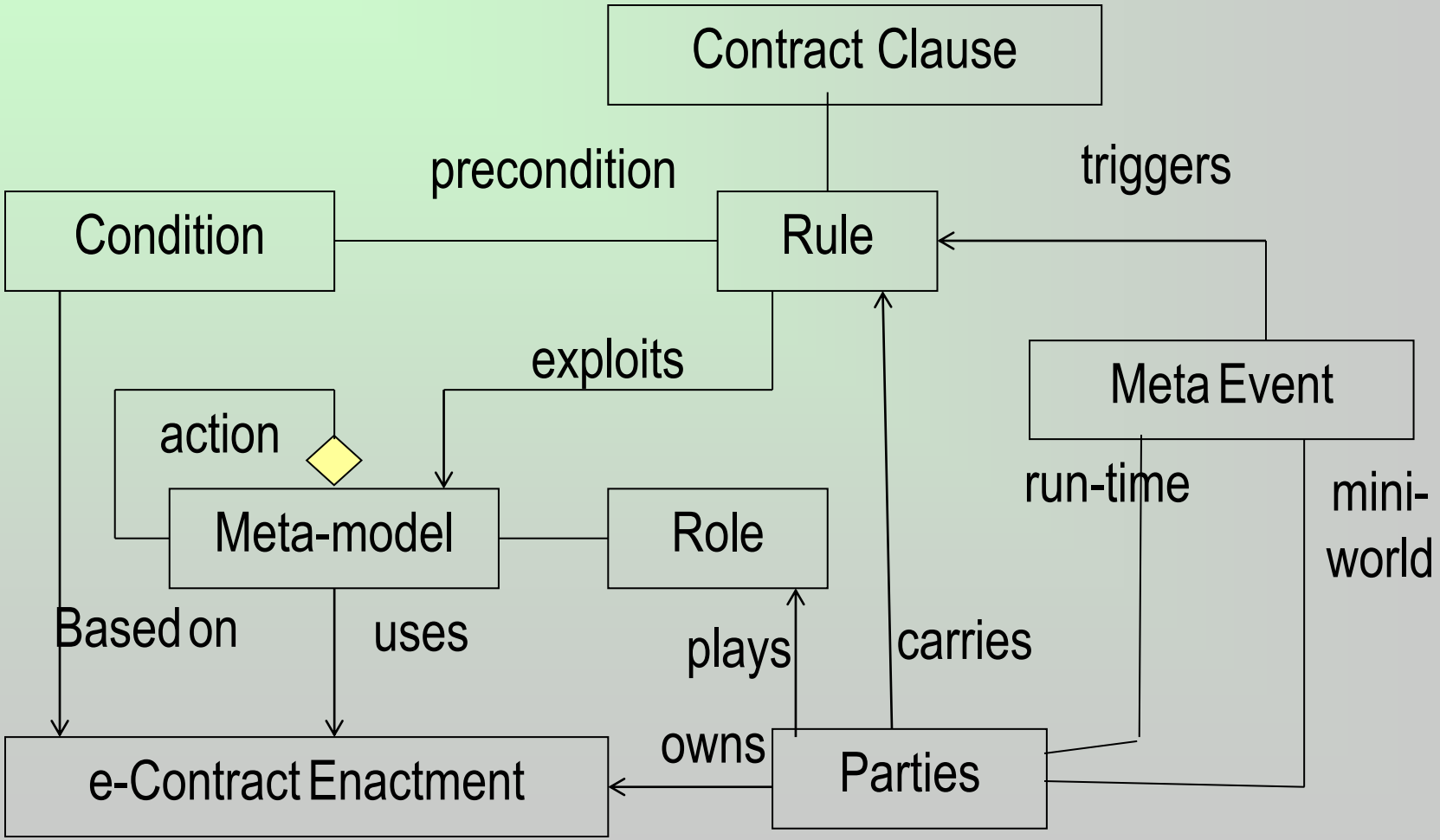
- Operations on Meta-model:
  - ***Adapt:*** *The model is allowed to adapt based on the new requirements.*
  - ***Migrate:*** *The change affects the current model instance and hence a new model has to be instantiated.*
  - ***Merge:*** *A new model is instantiated and merged with the current model.*
  - ***Build:*** *The change cannot be handled with current model and also a new model cannot be instantiated.*
- Instances of models before and after change execute:
  - allow any one running instance at any point of time
  - Allow multiple running instances during some period
    - **Abort:** *The change needs to have a new model instance immediately after the change occurs.*
    - ***Additive:*** *The change needs to have a new model while continuing the current model.*

# Meta-Model for ECA rule: *On* event *if* condition *then* action

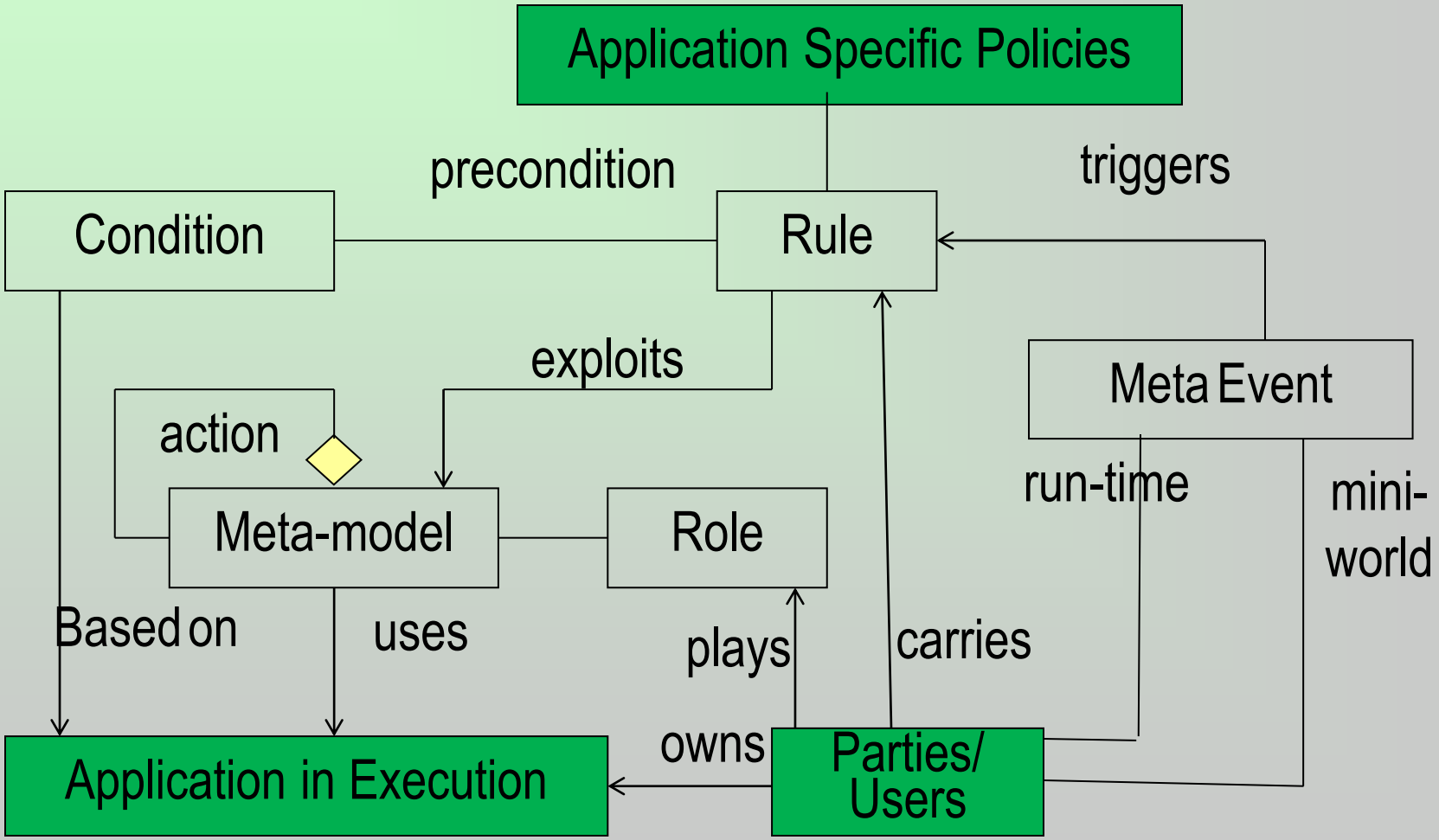



Chiu et al, 2003

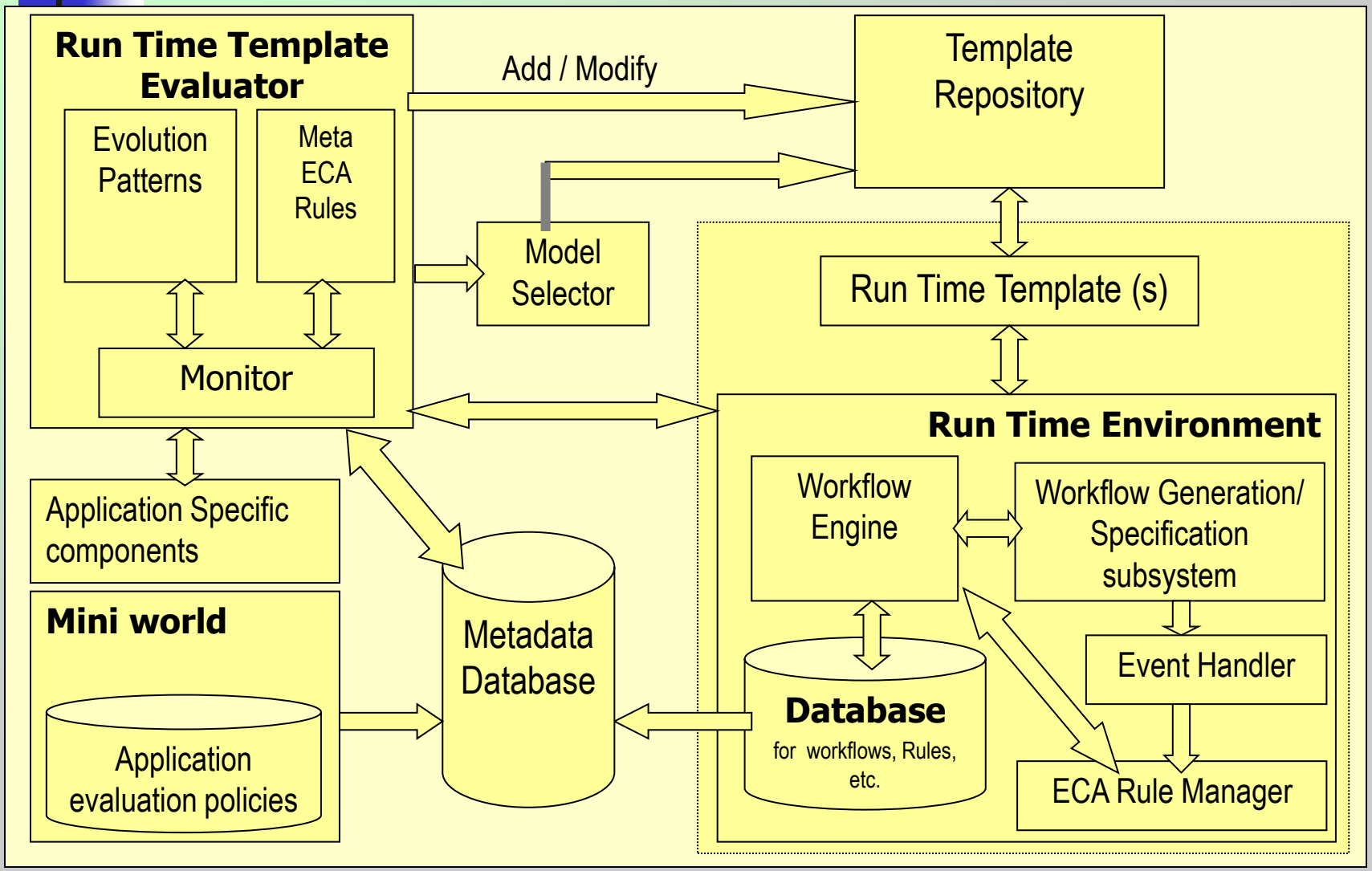
# Meta-ECA Rule Driven E-contract Evolution

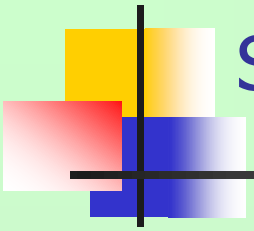


# Meta-ECA Rules Driven Evolving Applications



# ER*EC architecture for evolving contracts





# Summary

- Extended $ER^{EC}$ model and methodology to actively reflect the changes across various levels of data models in an e-contract.
- Two-way active behaviour and mechanisms tracks the progression of e-contract execution.
- Taxonomy of operations for template selection procedure for modeling evolution
- Meta-events and meta-ECA rules to adapt the models.
- Architecture to support evolving applications by facilitating run-time environment as well as capturing evolution patterns for run-time template evaluation/selection.

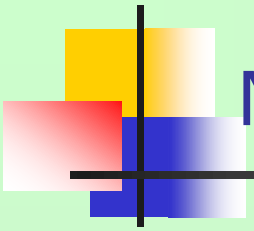
Our methodology helps in visualizing evolution procedure and develop specific procedures, methodologies and tools to actively support e-contract evolution.

# Meta Execution Models

# Need of Meta Execution Workflow

- Current WfMSs follow a (hard-coded) fixed execution control flow (with multiple control flows) to execute workflow instances.
- The workflow definitions can be changed, but the ways in which a workflow instance is executed is fixed. The challenges are
  - Dynamically change the way a WfMS engine functions without modifying the engine code, and
  - Allow changes in workflow specification procedure
  - Execution of user workflows with minimal human intervention and provide support for exception handling.

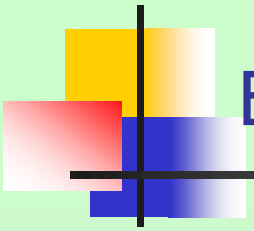
The challenge here is providing adaptability to WFMS, flexibility to workflows in execution, and generating new workflow specifications and execution scenarios during workflow enactment.



# Need of Meta Execution Workflow

- Requires *dynamically changing the way a WfMS engine executes workflows without modifying the workflow engine code.*
- A basic step towards providing flexibility is to generate abstraction of workflows and generate new workflows on-the-fly based on the current execution state of a workflow instance.

WfMS supports workflows executing workflows, where a WfMS engine procedure is specified and implemented as a Meta Execution Workflow.



# Example

Consider the following workflow execution logic:

(i) Get a task - execute - continue till no more tasks

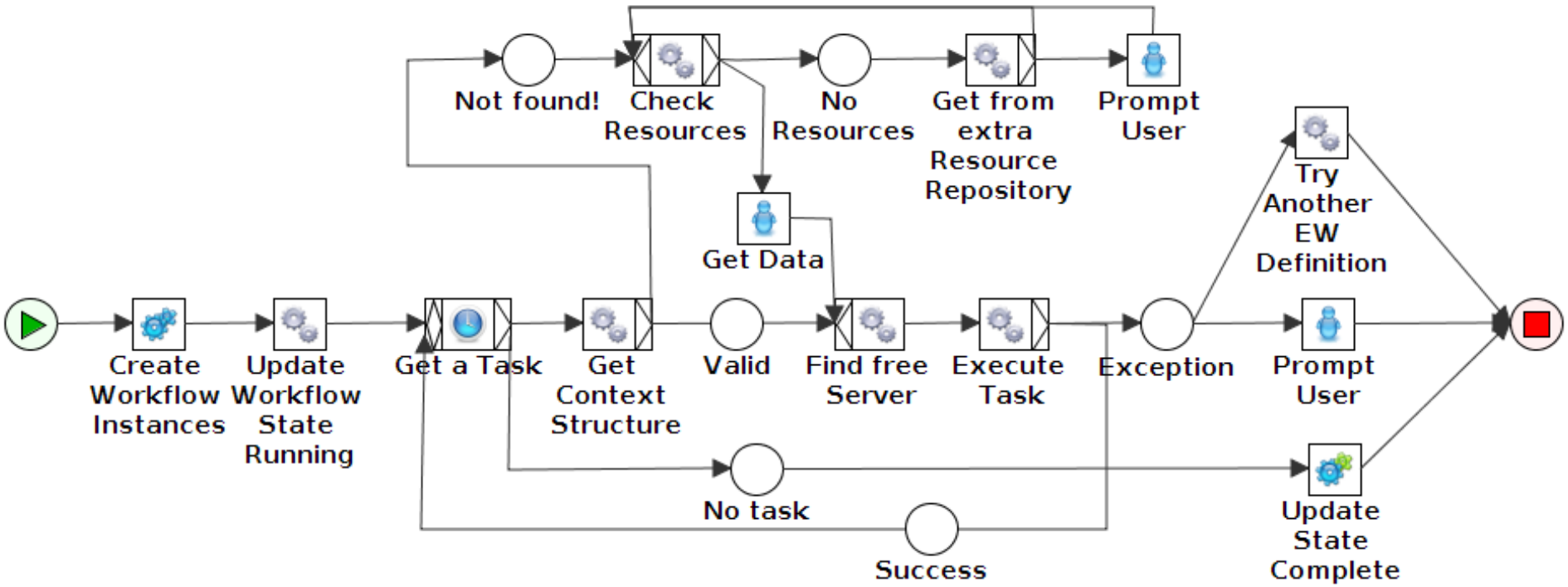
(ii) Get a task - check resources - execute - continue till no more tasks

(iii) Get a task - check resources - execute - call exception manager if required – handle exceptions - continue till no more tasks

Here, (i) simple workflow that do not require resources, (ii) requiring resources, and (iii) further needing exception handling.

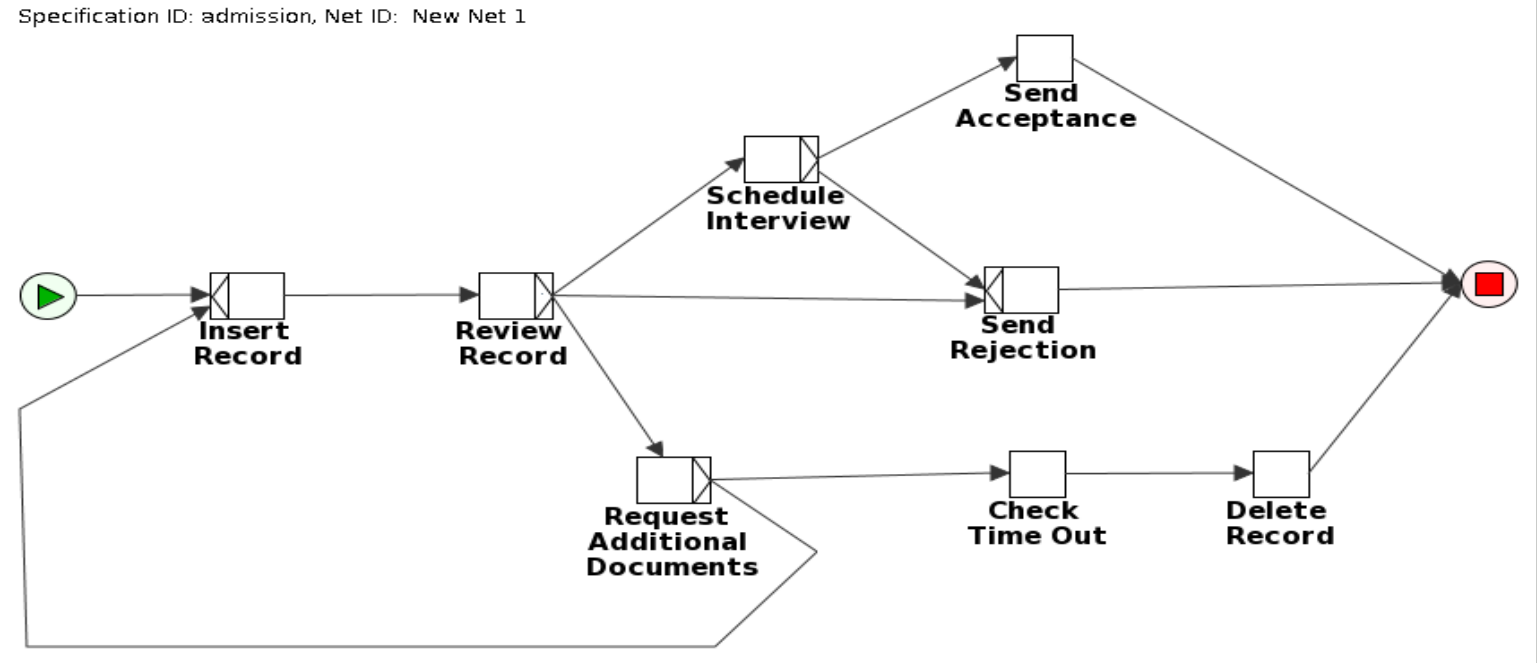
need to shuffle or add new execution steps, depending on how we want the workflow execution engine to execute workflows

# Meta Execution Workflow

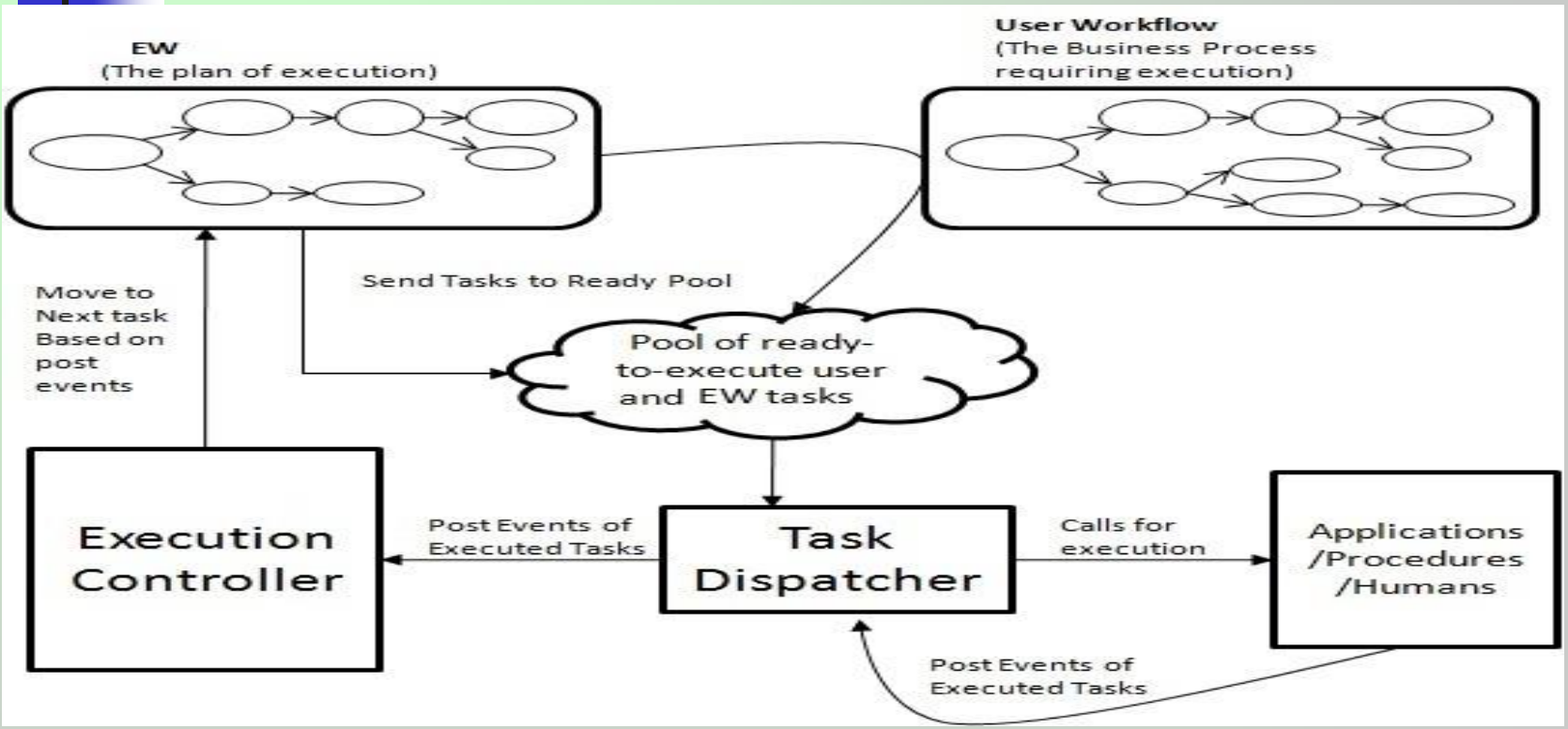



Systems based on meta-execution driven WfMS allows execution procedure to be modified, according to business and technological needs that have been built.

# Workflow for PG Admission in an Institution



# MEW driven WFMS components




From a pool of tasks ready to be executed, TD selects a task and assigns it to appropriate agents, applications or humans for execution.

# Workflow for PG Admission in an Institution

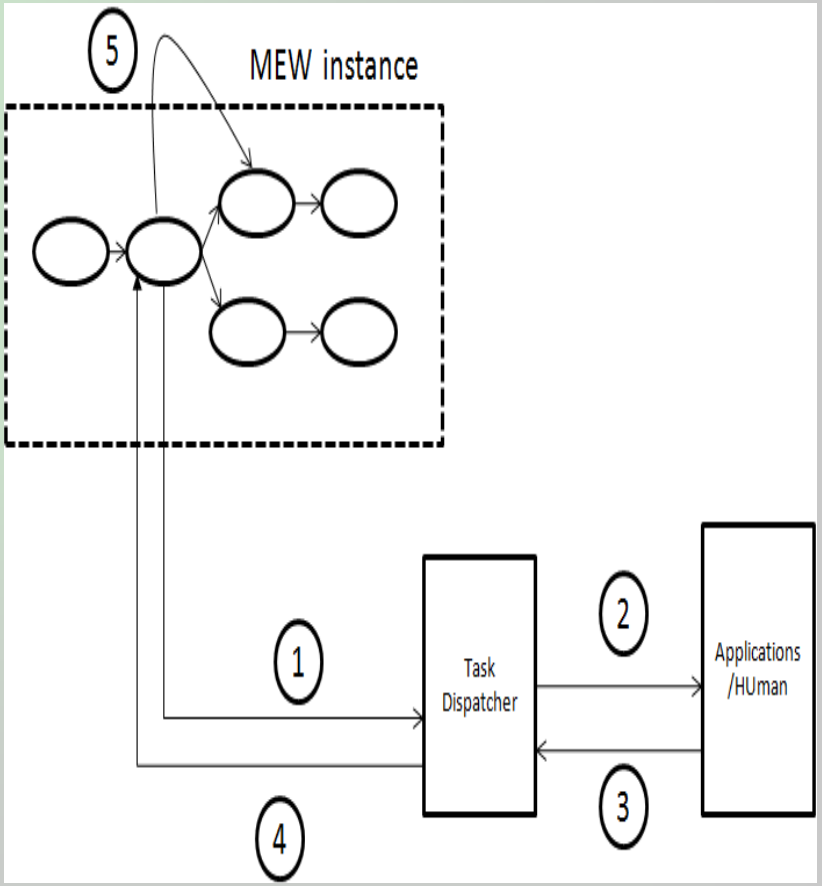
- Consider a single workflow instance
  - User Workflow Instance ID :13
  - Coupled EW Instance ID: 10
  - Coupled EW Instance Current Task: Check Resources
  - User Workflow Instance Current Task: Schedule Interview

the control is with the EW instance

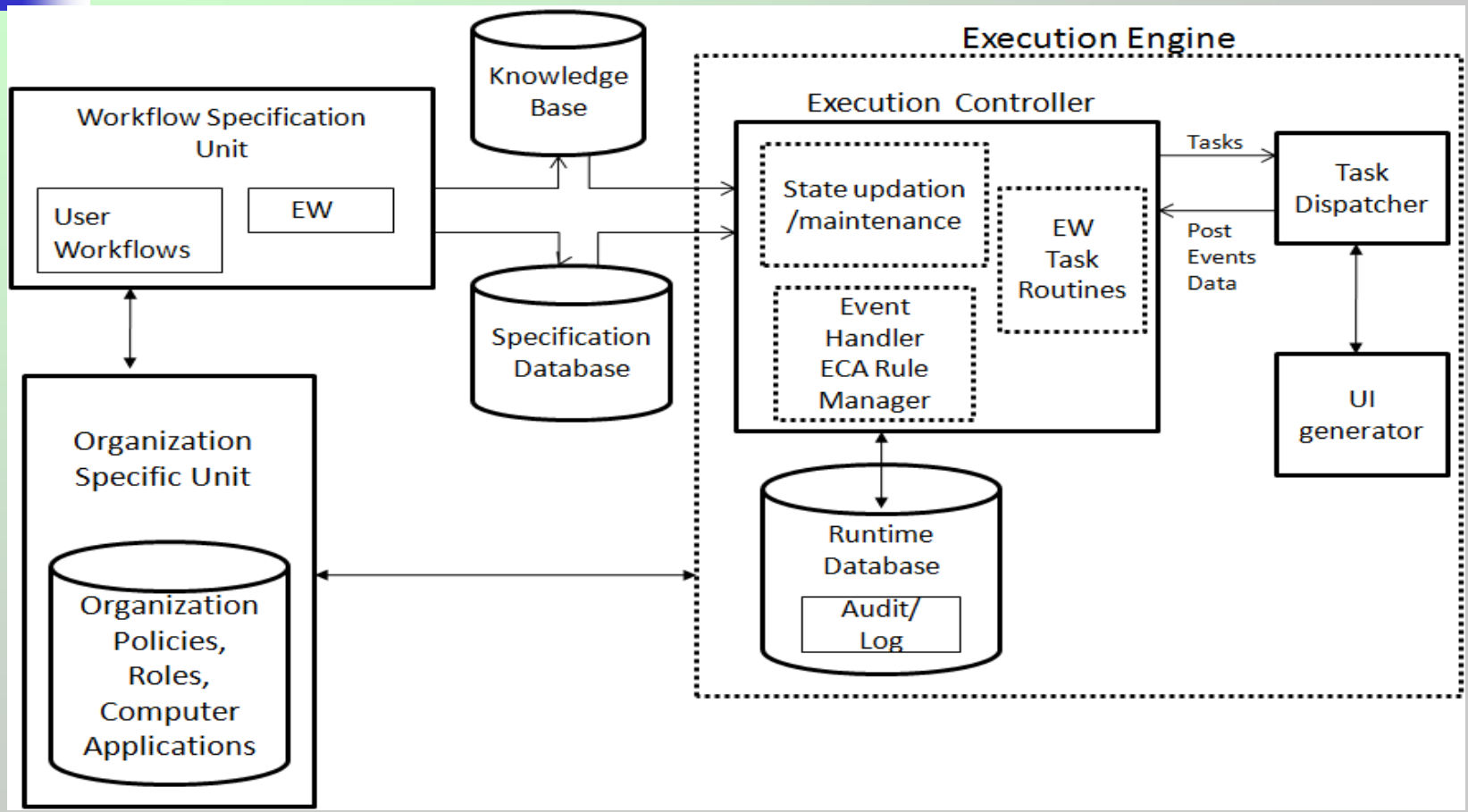
# Steps involved

The steps involved for shifting of control and data

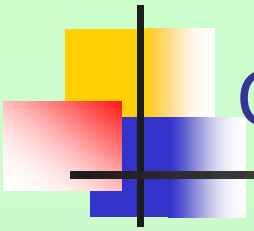
1. The MEW instance 10 passes the control to the TD, for the execution of its task “Check Resources" for “Schedule Interview".
2. TD sends the task for execution to an application/human.
3. The task is executed; events are captured by the TD and the control if returned to the dispatcher.
4. The TD returns a post event data tuple to the EW instance ID 10. With this step, the control and data tuple is returned back to the MEW instance task – “Check Resources" (where it started from).
5. Based on the post event (suppose “success" in this case), the current MEW task passes on the control and the post event data to the new MEW task – “Execute".



# Workflow Specification and MEW Based Execution - Architecture

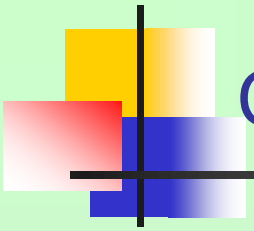



Supports (i) on-the-fly specification and execution of workflow and (ii) exception handling



# Context-Aware Execution Workflow (CEW)

How to model context for conceptual model, meta model and meta execution models.



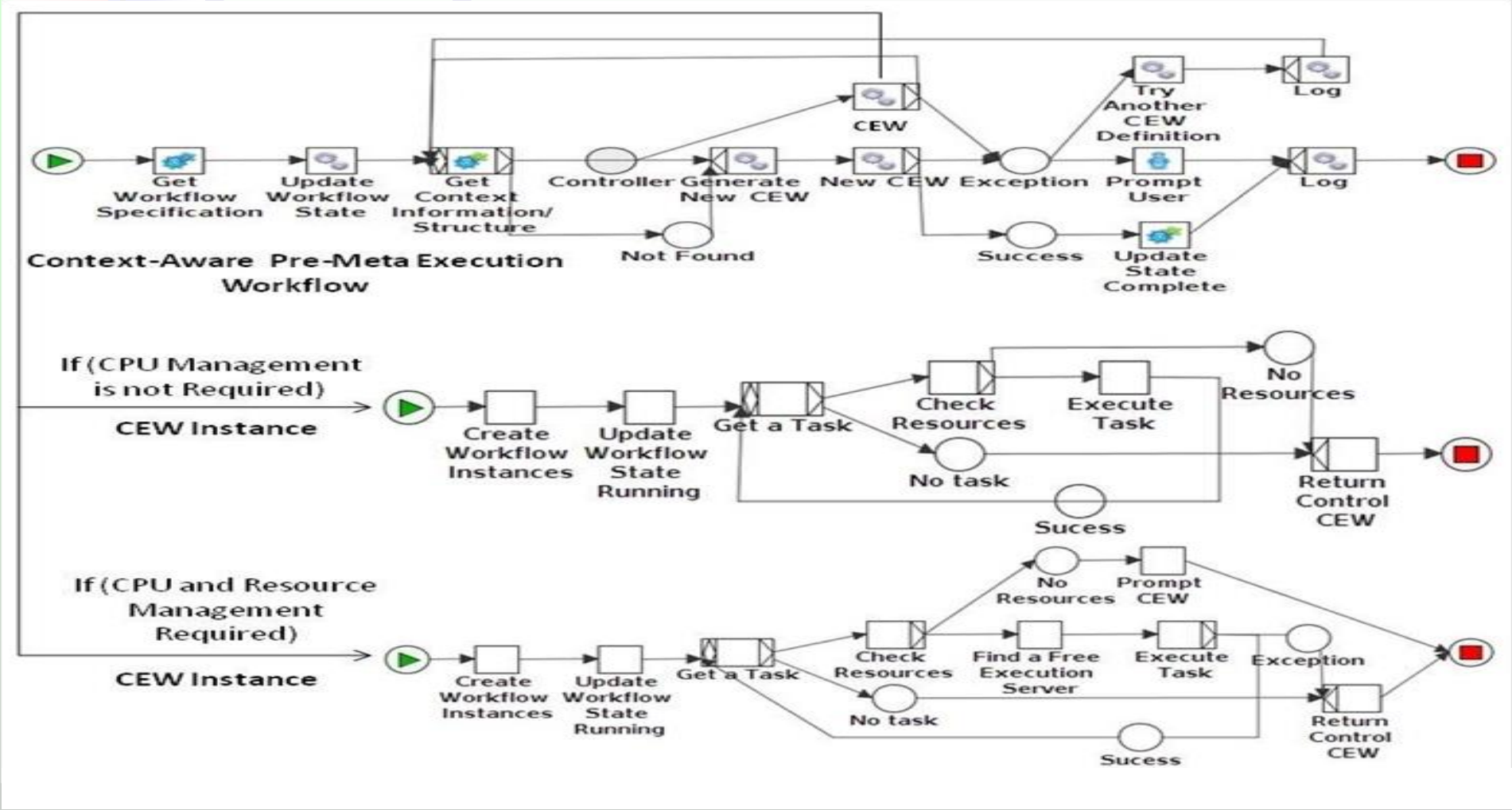
# Context Vs Exception

- Exceptions are handled by Exception handlers (in terms of workflows)
- Context is beyond exception and they require different kind of handlers.

# Context-aware (Meta) Execution Workflow

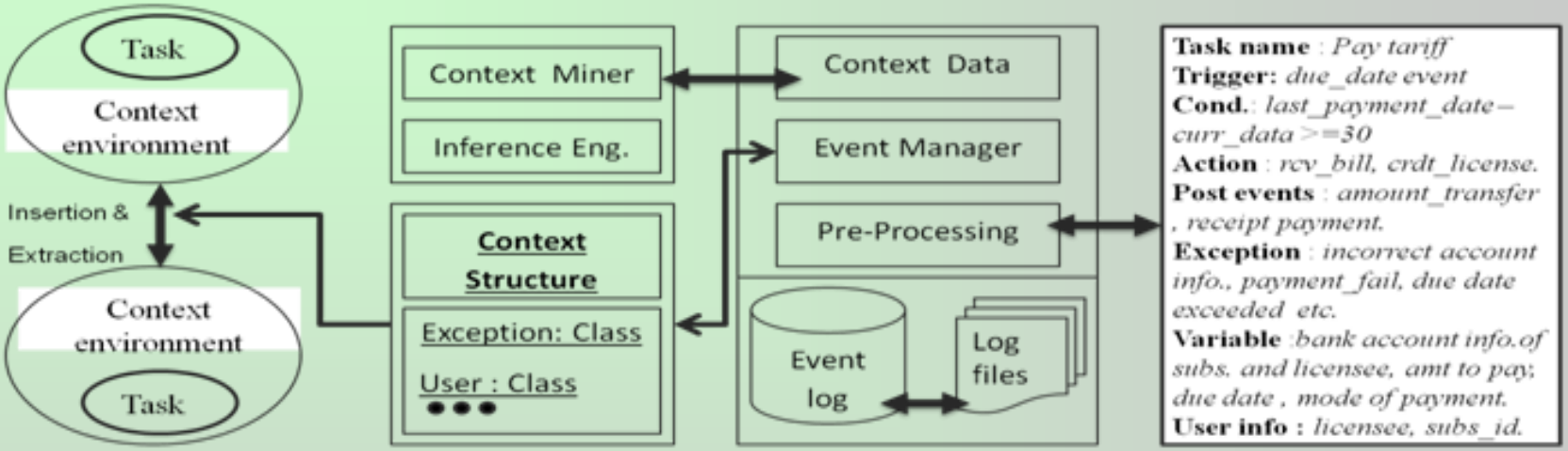
- Allows flexible execution control flow and procedure to specify and execute workflows.
- CEW procedure is implemented as the workflow engine, and can be modified or enhanced based on the current context.
- Developed a re-usable context-aware workflow execution for generating context-aware execution workflow (CEW) instances.

# Context-aware (pre-)Meta Execution Workflow generating CEWs




re-configure and manipulates its objects based on the context information.

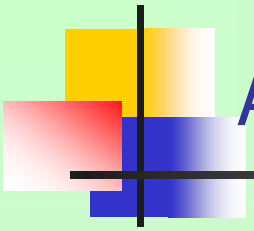
# Task handling - Payment of tariff




Example: When the due date for payment of tariff falls below a deadline, the payment has to be done by the client to the service provider to renew the service.

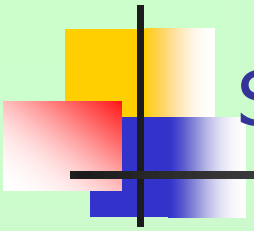
Need context information

*Based on clientInformation (last renewal date, due date, type); bill details; provider's bank information for crediting payment, etc.*



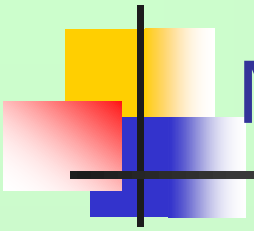
# Activity Execution

- Execution of an activity is done by using specified transition path, transition condition between the tasks and other parameters related to a task.
- Resolve the various dependencies between tasks such as data-dependency from users, temporal event dependency and dependency of a task on external events.
- Benefits:
  - Less redundancy and consumption of time, because instances of an activity usually consists of same definition and prone to similar exceptions.
  - Pre-conditions can be evaluated apriori and execution engine can proceed without allowing it to wait for the evaluation of that condition.
  - Time allocation can be done better depending upon the past execution time of different instances of same task.



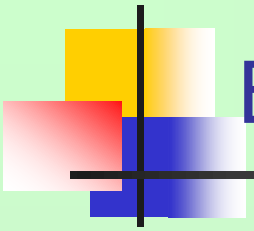
# Summary

- Meta Execution Workflow driven workflow execution extends support for
  - dynamic and flexible workflow executions.
  - exception handling
  - context-awareness to applications
  - evolving application requirements



# MDA - Ecosystem

- Model-Driven approaches to the system development
  - A shift from programming to modeling activities
  - Generation of software components from models.
  - Reduce human interaction
- Areas
  - requirements engineering
  - information system Design
  - Databases design
  - Information System Development
  - …….



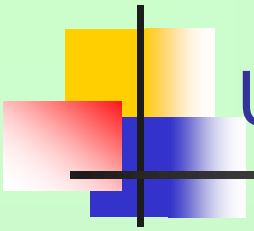
# Background

Models in Software Engineering - Terminologies

- Model Driven Software Engineering
- Model Driven Software Development
- Domain Specific Modeling
- Model Driven Architecture
- Software Process Models
- Business Process Models

These technologies deals with coping with the complexity of software development by raising the abstraction level and introducing more automation in the process.

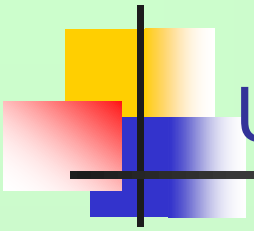
Their focus is mainly to improve software quality, increased traceability between artifacts, early defect detection, reducing manual and error-prone work and reduce development cost



# UML meta-model

- UML meta model defines a language for specifying UML models
  - Use Case Diagram Meta Model
  - Class Diagram Meta Model
  - Capable of adding new members (UML profiles) to the family
- Lack of effective Model Transformations and their traceability.

"The UML standard has evolved but, with this evolution, the syntax has become even more complex and the necessary supporting mechanisms and tools for dealing with this added complexity are not yet available. Even something as conceptually simple as exporting a UML diagram from one tool to another has not been accomplished yet with ease." – Mohagheghi et al, 2008

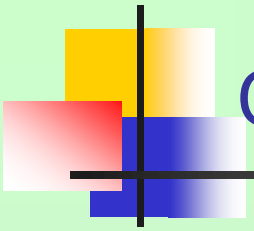


# UML Vs our approach

- Can our approach complement/overlap with UML modeling ?
- Can we solve some of the issues with UML ?
- Can UML helps in enhancing our approach ?

Our exposure to UML modeling is limited.

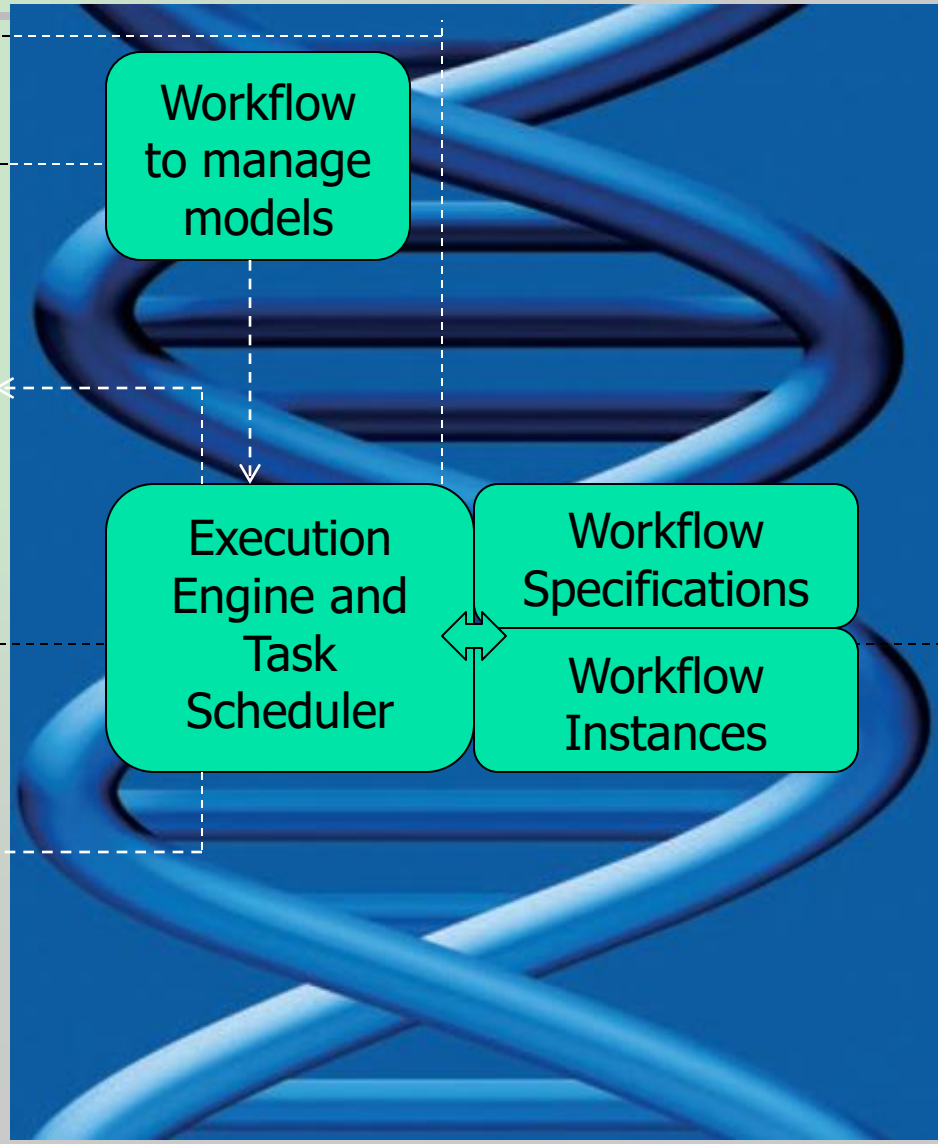
Open for Ideas and Discussion !!!!



# Open questions

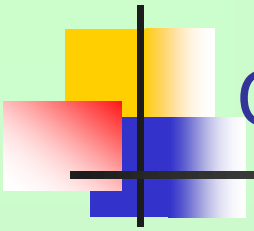
- What are the problems we solved that will help SE
- What are the problems in SE that we cannot solve
- What is the scope and reach of our solution from and towards SE perspective

# The big picture



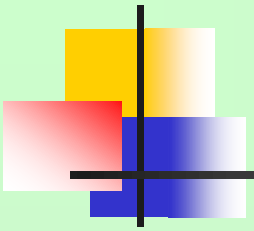

**DNA of Information System Processing**

# Open Issues



# Open Issues

- The level of abstraction needed to model the reality as close as possible.
- Zero down (atleast reduce) the semantic mismatch between model design and reality for a class of applications
- Actionable meta models and meta execution models
- Tracking the evolution
- Scalability of models
- Run-time models
- Attributing Risk involved and cost associated



# Conclusion

Meta Models and Meta Execution Model provides a powerful constructs to seamlessly model data and processes.

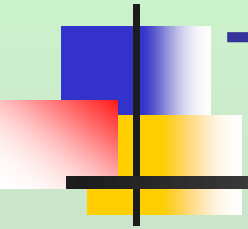
- Manage the data and processing capabilities of changing reality in a seamless manner.
- Helps tie up from conceptual layer to actual physical layer by using appropriate specific constructs, their implicit semantics and constraints to cater to dynamic and evolving reality.

# Bibliography - 1


- Boudewijn F. van Dongen and Wil M. P. van der Aalst, A meta model for Process Mining Data, In EMOI-INTEROP, 2005.
- Brottier,E., Fleurey, F., Steel, J., Baudry, B. and Traon, Y. L., Metamodel-based test Generation for Model Transformations : an Algorithm and a Tool, In proceedings of the IEEE ISSRE'2006 conference, 2006.
- BPMN and OMG. Business Process Model and Notation (BPMN), Version 2.0, January 2011, http://www.omg.org/spec/BPMN/2.0
- Dashofy, E., Asuncion, H., Hendrickson, S., Suryanarayana, G., Georgas, J., and Taylor, R.N., ArchStudio 4: An Architecture-Based Meta-Modeling Environment. In ICSE COMPANION '07: Companion to the proceedings of the 29th International Conference on Software Engineering, 2007.
- Fantinato, M., de Toledo, M. B. F., and de Souza Gimenes, I. M., WS-contract establishment with qos: an approach based on feature modeling. Int. J. Cooperative Inf. Syst., 17(3):373–407, 2008.
- Fey, D., Fajta, R. and Boros, A., Feature Modeling: A Meta-Model to Enhance Usability and Usefulness, Proc. of the Second International Conference on Software Product Lines (SPLC2), Springer-Verlag, LNCS 2379, pp. 198-216, 2002.
- Krishna, P. R. and Kamalakar, K. , "A Methodology for Evolving E-contracts Using Templates", IEEE Transactions on Services Computing, Vol. 6, No. 4, pp. 497-510, 2013.

# Bibliography - 2


- Jain, H., Krishna, P. R. and Karlapalem, K., Context-Aware Enactment of E-Governance Contracts, IJCAI 2013 Workshop on Activity Context Aware System Architectures, Beijing, China, August 2013.
- Laguna, M.A. and Marques, J.M., Feature Diagrams and their Transformations: An Extensible Meta-model, 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009 ( SEAA '09), pp. 97 – 104, 2009.
- Lodhi A, Koppen V and Saake G, An Extension of BPMN meta-model for evaluation of Business Processes, Scientific Journal of Riga Technical University, vol. 46, pp.27-34, 2011.
- Muller, P-A., Fleurey, F., Drey, Z., Pollet, D., Fondement, F. and Studer, P., On Executable Meta-Languages applied to Model Transformations, Model Transformation In Practice (MTIP) workshop, 2005.
- Pabitra, M., Krishna, P. R. and Karlapalem, K., E-contract Enactment using Meta Execution Workflow", 21st International Conference on Cooperative Information Systems (CoopIS 2013), 11-13 Sept 2013, Graz, Austria, 2013.
- Rohlik, O., Pasetti, A., Ekstein, K. and Chevalley, P., A Meta-Modelling Approach to Feature Modelling, http://pnp-software.com/XFeature/pdf/XFeatureToolConcept.pdf, June 2005.
- Sharma, S., Karlapalem, K. and P. R. Krishna, A Case for a Workflow Driven Workflow Execution Engine, 22nd Workshop on Information Technologies and Systems (WITS 2012), Orlando, Florida December 15-16, 2012.



# Thank you

Happy Meta-World !!

kamal@iiit.ac.in

Radhakrishna_p@infosys.com