

# Advanced Technical Aspects of Requirements Development

Toronto SPIN workshop

Sergey Diev

[www.diev.com](http://www.diev.com)

January 2009

© Sergey Diev

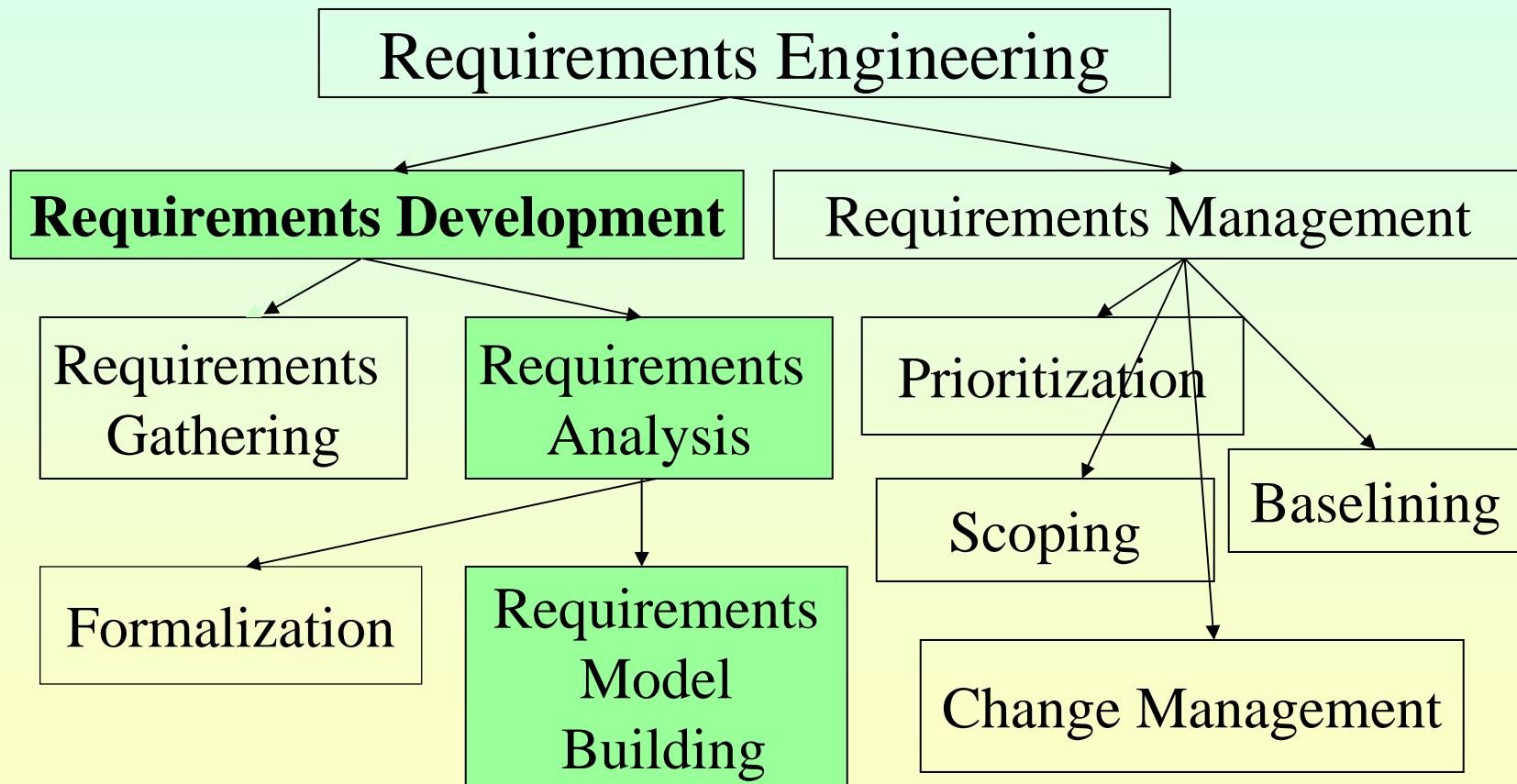
# Workshop agenda

- Presentation “Technical aspects of requirements development” (60 min)
- Break (10 min)
- Case study (105 min)
- Wrap up (5 min)

# Contents

- Requirements development
- Requirements complexity
- Aspect: Requirements structure
- Aspect: Concepts
- Aspect: Model vs. representation
- Aspect: Use case visualization
- Aspect: Multiple representations

# What this workshop is about



# Sources of requirements complexity

- The problem
  - “The complete idiot's guide in trouble-free heart surgery”
- The product
  - “Nuclear power station in five simple cartoons”
- The project
  - “Manned moon mission in three simple steps”
- The life cycle model
  - “Agile skyscraper construction for dummies”

# When are requirements complex?

- There are many business concepts (to implement, to deal with)
  - Drives the need in a conceptual model
- There are many new concepts
  - E.g., in e-mail application - 'import'
  - May lead to creating a subset of the concept model
- There are many requirements
  - Drives the need in separate views and sets
- There are many levels of requirements
- There are many kinds of links between requirements
- There are many dependencies between requirements
- Same requirements are used in many places
- Etc.

# Ways to deal with complexity

Simplicity is prerequisite for reliability.  
Edsger Dijkstra

Vision without action is a daydream.  
Action without vision is a nightmare.  
Japanese proverb

- **M**odeling
- **A**rchitecture
  - Separation of requirements in views and sets
  - Specialized views
  - Layering and hiding
  - Links
- **R**epresentation of requirements

# Questions about aspects

- How important is this aspect?
  - Are there other examples demonstrating importance of the aspect?
- How can this aspect be addressed?
  - Without a tool
  - With a tool
- What are the advantages of specific solutions?

# Sample problem

*Deliver an application that allows a registered client to order items using a catalogue.*

*The application shall also allow administrators to see clients activity and blocked orders*

# Aspect: Structure

- Requirements gathering, analysis and implementation depend on how requirements are structured (and represented)

- Read this:

*“suppose all requirements are defined in plain text suppose there are no spaces no punctuation marks no statement structure and no paragraphs”*

# Structure cont'd

*And then read this:*

*“Suppose all requirements are defined in plain text. Suppose there are no spaces, no punctuation marks, no statement structure and no paragraphs.”*

Is this structure the only one that we need to take care of?

*Requirements structure is one of most important aspects of requirements development*

# Linear text is not enough

- It is tempting to work with *textual* requirements as just with another text; however, in most cases
  - requirements are more than just a linear text
  - they have more than just a two-level structure
  - their structure is more complex than just a tree
- Various aspects of a system under design – causal, temporal, navigational, etc. – can be presented better using other structures than linear text
- That improves requirements consistency, completeness, readability, changeability, etc.

# Spreadsheet structure is not that good too

- In many projects behind of what is called “requirements” lies a complex structure of objectives, assumptions, statements, conditions, constraints, etc.
- Ignoring this structure leads to obscure requirements, incompleteness, **requirements bugs**, lower productivity

# Requirements structure examples

1. A 100% new system in a known business domain
  - The high-level application architecture view would probably be not required to be present among requirements views
2. A 100% new system in a new business domain
  - The concept view would probably be the center point
3. A technical upgrade
  - Only the use cases view would probably be required (to be used in test plans)

# Architecture

*For structurally complex requirements it is necessary to define **requirements architecture***

*– in accordance with the properties of the problem*

*Architecture: “The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.”*

ANSI/IEEE 1471-2000

# Requirements architecture

- Requirements architecture is about:
  - Making requirements easier to capture, analyze, modify, find and understand
  - Seeing requirements in appropriate groupings and contexts and at appropriate levels of details
  - Avoiding requirements duplication
  - Making requirements easier to reuse
  - Making requirements more consistent and complete
  - Etc.

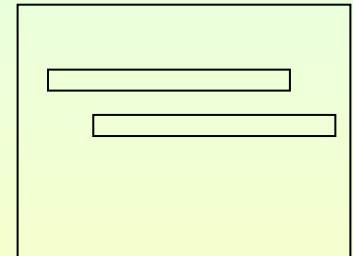
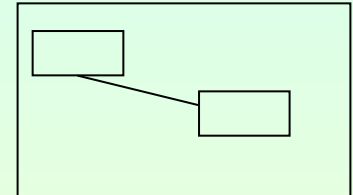
# Requirements architecture decisions

- Which requirements sets will be at the core of the requirements model
- Which views to use
- How many levels of requirements to use
- How links will be used
- How to link requirements to each other
- How to represent views
- How to document the requirements model
- Etc.

*These decisions depend also on the tool*

# Links modeling

- Links are modeled in many ways:
  - By creating an explicit connection between modeling elements
    - Most popular way
    - Requires a tool to manage properly
  - By placing requirements together
    - E.g. a detailed requirement is placed under high-level requirement
  - By proper naming
    - E.g. requirements related to order trading could have names “ORTR\_1”, “ORTR\_2”, etc.
  - By placing one requirement on the content-diagram of another
  - Etc.



# Structuring decisions for the sample problem

We may decide that:

- We need a conceptual model
  - Because the domain is new
- Functionality will be defined using use cases
- Arbitrary number of requirements levels will be allowed
- Dependencies between requirements will be shown with the help of simple links
- Etc.

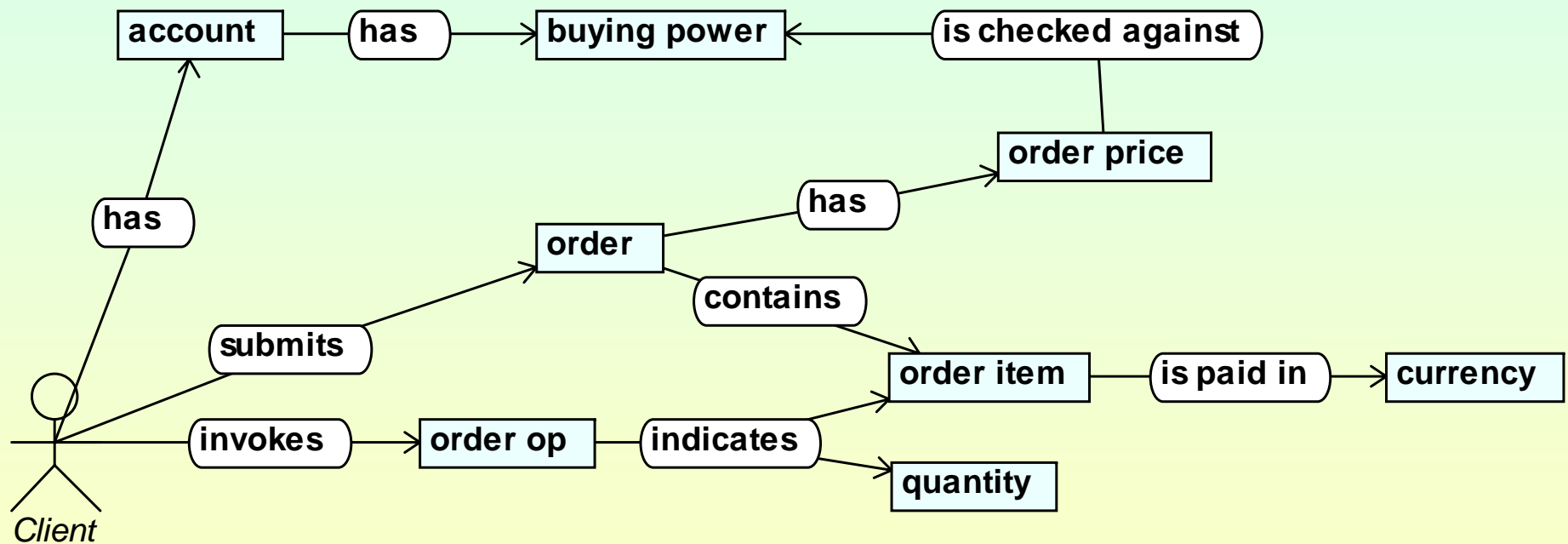
# Aspect: Conceptual model

- Contains concepts
- Shows how concepts are linked to each other
- Could be a multi-level, multi-view structure

# Conceptual model: Crystallization

- An activity resulting in requirements related to a particular concept placed around this concept
  - This can be represented on the diagram with the concept or on the content-diagram of the concept or link
- Helps to make requirements complete and consistent

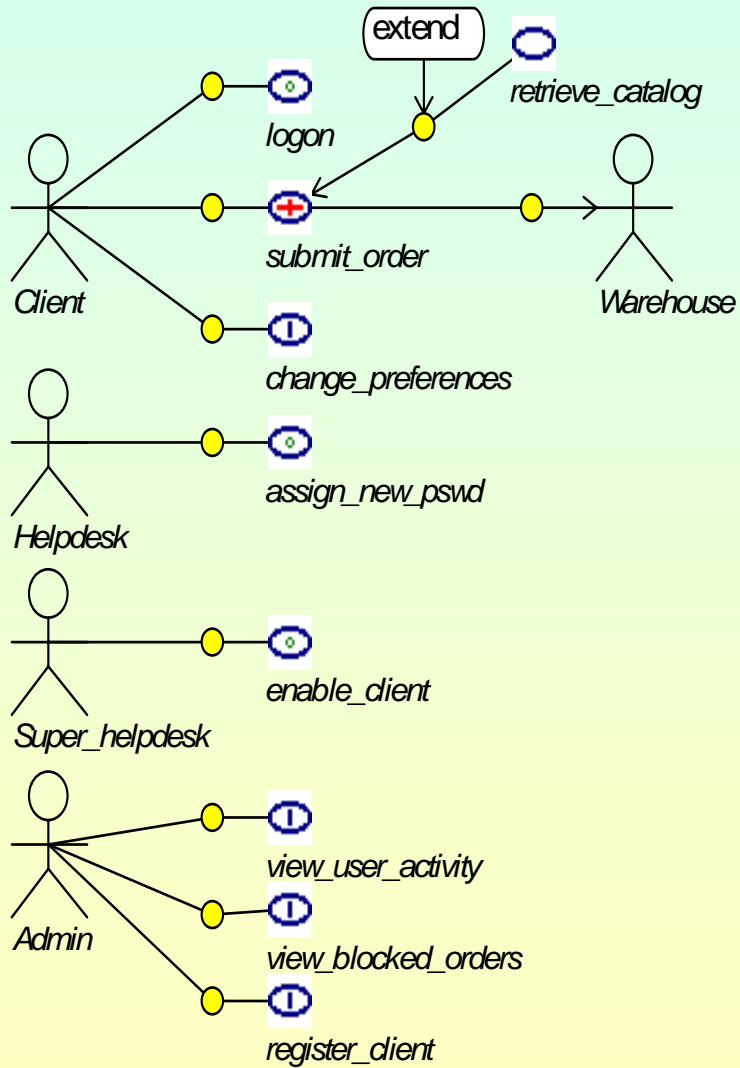
# Conceptual model



# Selecting views: Use cases view

- Defines functionality in a coherent way at the level of:
  - a) use case survey
  - b) use case descriptions

# Use cases view



# Selecting views: Screens view

- Defines logical screens
  - Is derived from use cases
  - Is useful for presenting the system under construction to the business client
  - Is required for developing user interface

# Screens view

## **Client screens**



*signon*



*home*



*order\_creation*



*order\_status*



*order\_review*



*preferences*

## **Helpdesk screens**



*Client\_settings*

## **Super\_helpdesk screens**



*enabling*

## **Admin screens**



*run\_activity\_rept*



*user\_activity\_rept*



*raun\_blocked\_orderr\_rept*

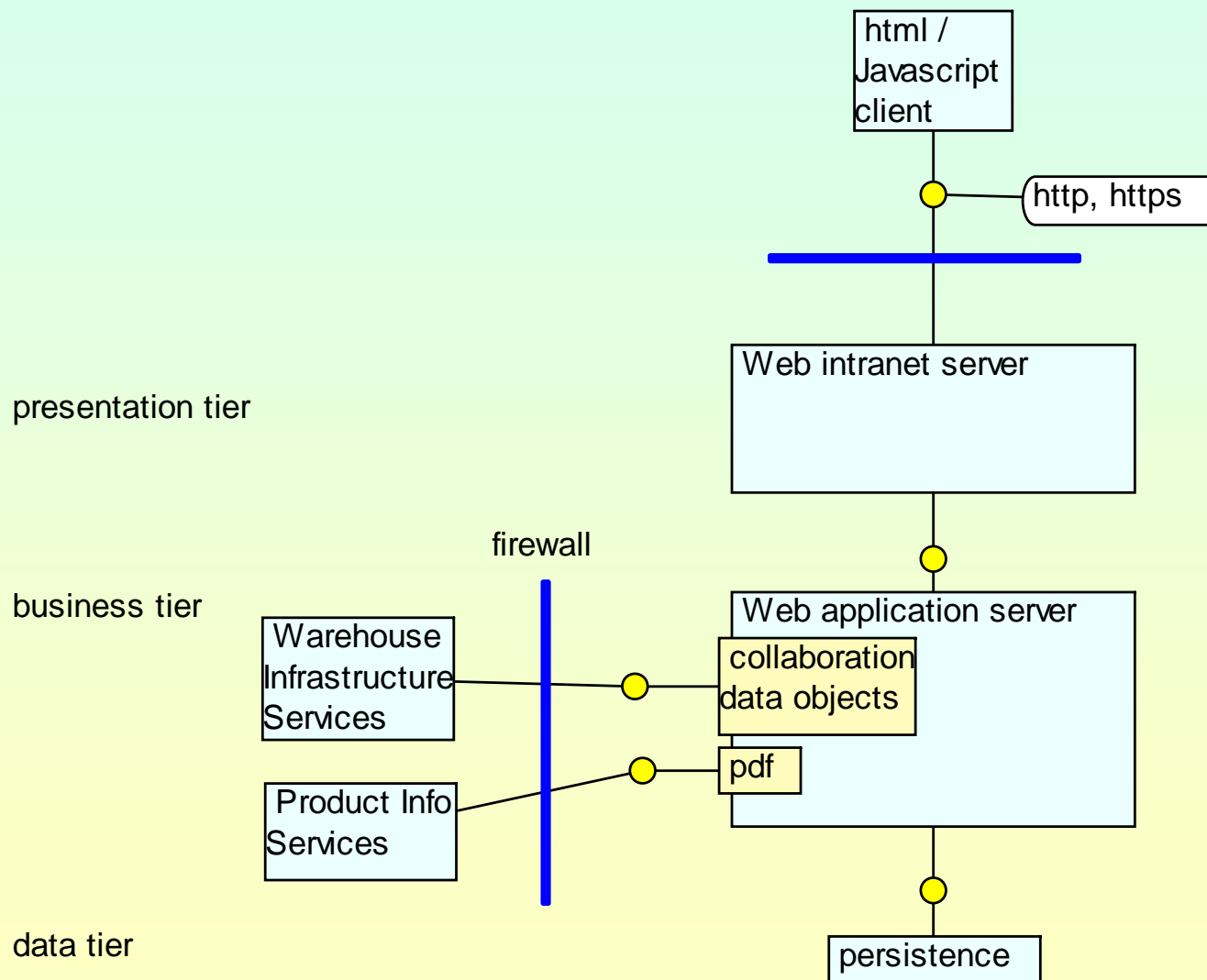


*blocked\_orders\_rept*

# We may need: Navigation view

- Shows allowed transitions between screens
- Useful when there are many screens
- Is a good way to represent conditional navigation in presence of non-trivial business rules
- May contain not all screens (unlike screens view)

# We may need: Architecture view



# We may also need

- Reports view
- Estimation view
- Problem-specific views / sets
  - trading set
  - quotes set
  - scheduled events set
- Etc.

# More details for views?

- We have to answer these questions:
  - Do we need more structure within each of these views at the current stage of the project?
  - How will these views be linked to each other?
  - Which kinds of links do we need?

# Static views

- Features view
- Concepts view
- Objects view
- Business rules view
- Data view
- Screens view
- Functions view
- Application architecture view
- Reports view
- Etc.

# Dynamic views

- Business use cases view
- Use cases view
- Operational scenarios view
- Workflow view
- Data flow view
- Screen navigation view
- Events view
- States view
- Etc.

# There are other views

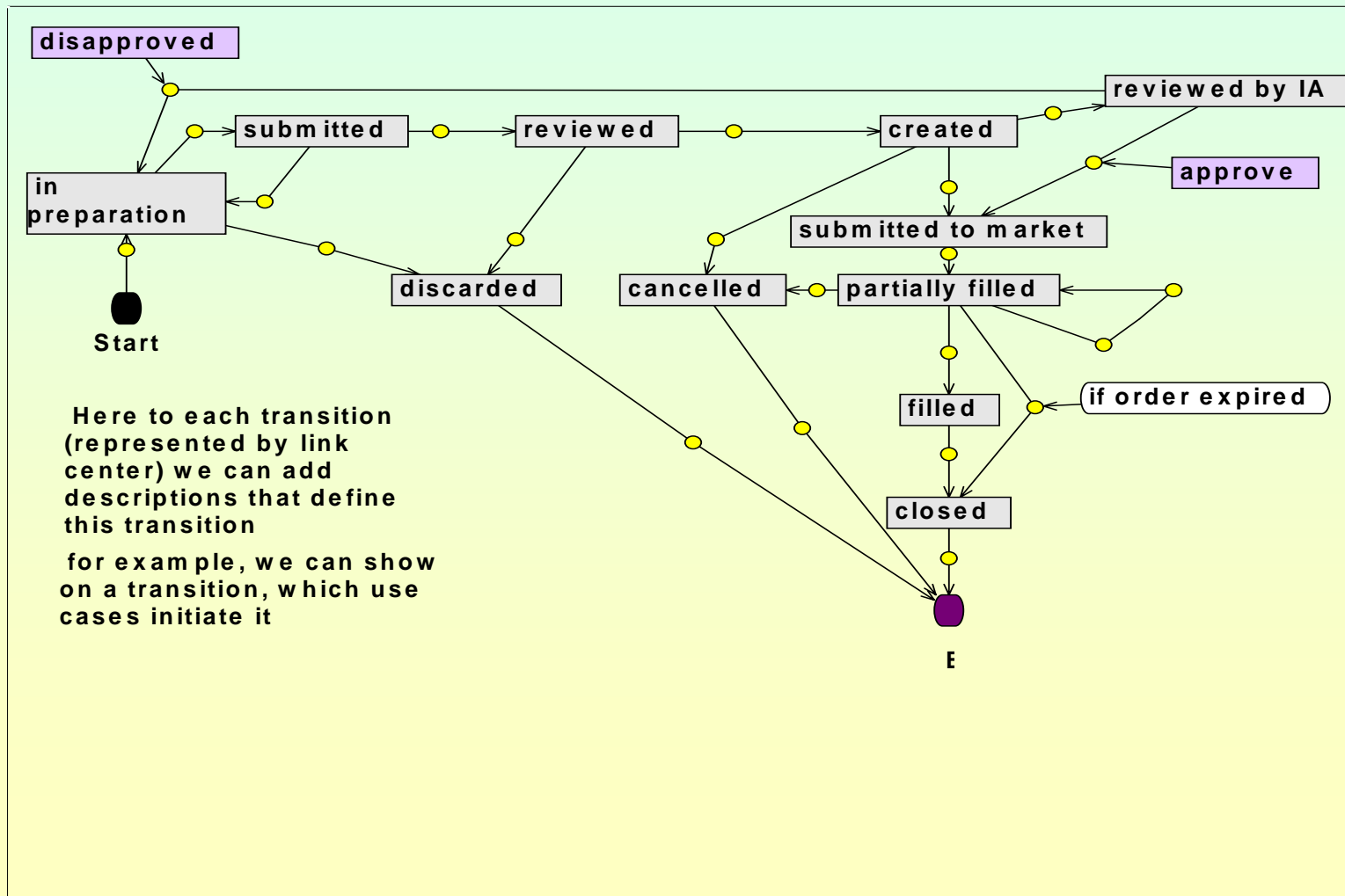
- Business views, such as Zachman's business logistics; business schedule; etc.
- CMMI: Stakeholder needs; stakeholder constraints; stakeholder interfaces
- Etc.

*Client shoots himself*  
*in the foot* example  
comes here

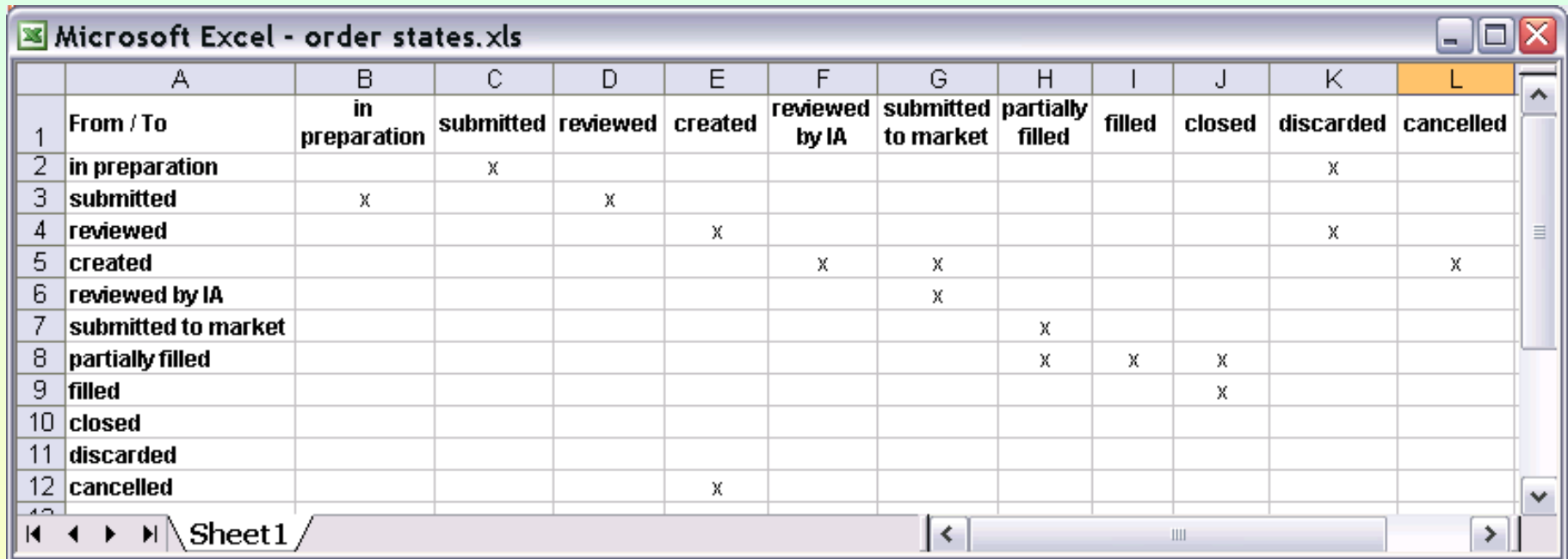
# Aspect: Model vs. representation

- Requirements is a model – an abstraction that serves a purpose
- Models can be presented in many ways
- We need to choose representations that fit the problem best
  - E.g. order states can be described in several ways

# Model vs. representation: Trade order states transition diagram



# Model vs. representation: Trade order states transition matrix



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - order states.xls". The spreadsheet displays a transition matrix for trade order states. The columns represent the current state (A-L) and the rows represent the next state (1-12). The matrix is lower triangular, indicating that an order can only transition to a state that is equal to or more advanced than its current state.

	A	B	C	D	E	F	G	H	I	J	K	L
1 From / To		in preparation	submitted	reviewed	created	reviewed by IA	submitted to market	partially filled	filled	closed	discarded	cancelled
2 in preparation			x								x	
3 submitted		x		x								
4 reviewed					x						x	
5 created						x	x					x
6 reviewed by IA							x					
7 submitted to market								x				
8 partially filled								x	x	x		
9 filled										x		
10 closed												
11 discarded												
12 cancelled					x							

# Model vs. representation: Trade order states transition textual description

**(Submitted)** After Client has entered the parameters of an order, he submits it to the system for verification

**(Reviewed)** After the system returns the results of the order verification, Client reviews the order

**(Created)** Client then either creates the order (sends it for execution) or

**(Discarded)** discards it

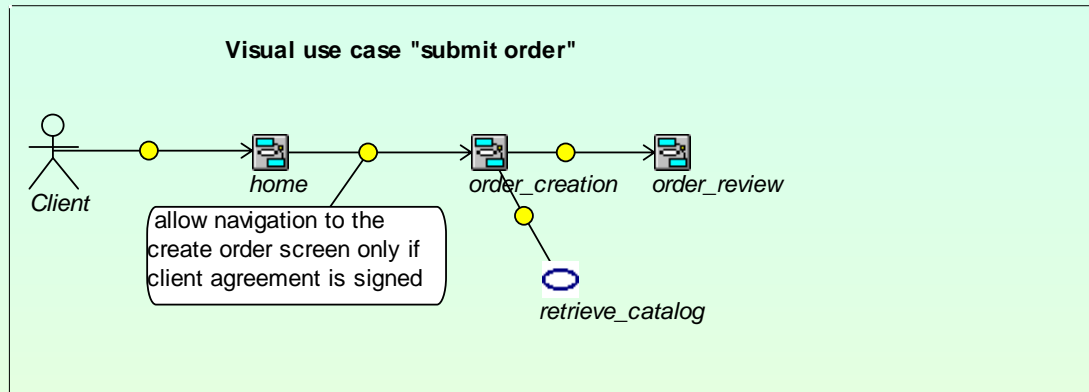
After an order is sent for execution:

**(Reviewed by IA)** 4.1. If Amount > \$10,000 the system sends the order to Investment Advisor for approval

**(Submitted to market)** 4.2. Otherwise, the system submits to the market the order

Etc.

# Aspect: Visualizing use cases



Home Order Product Info Preferences Help Contact Us Log out

home preferences

**Create order** order\_status

Account

account Account type: Individual Buying power: \$57900.43

Action: order op Item: ? Quantity: quantity Currency: currency

Submit Reset Cancel show\_catalog

order\_review catalog

testing\_for\_create\_order

# Visualizing use cases

- Use logical groupings of elements (concepts, requirements and texts) (as *pages*)
- Hide secondary details
- Lose procedural details where possible
- Add test (and other) details where necessary

*Understanding a problem is half the solution. Properly representing the problem is half of that.*

# Aspect: Multiple representations

- We may need to have various representations of the same content, e.g.:
  - Multiple perspectives
    - Business' perspective is different from the developer's one
    - Developer's perspective is different from the estimator's one
  - Product lines
    - Similar, but not identical sets of requirements
  - Needs of release
    - Implementing incomplete specifications
  - Etc.

# Multiple perspectives: Example

- The states model for trade order may be created from the perspective of:
  - Client: To explain how it works (order\_states\_main); will get reflected in messages and screens
  - Business: Will include client-system dialog (started\_)
  - High-level design: would include "Cancel issued" (order\_states\_cancel)
  - Detailed design: could include, for example, states for orders sent from handheld (e.g. lost signal) and from desktop

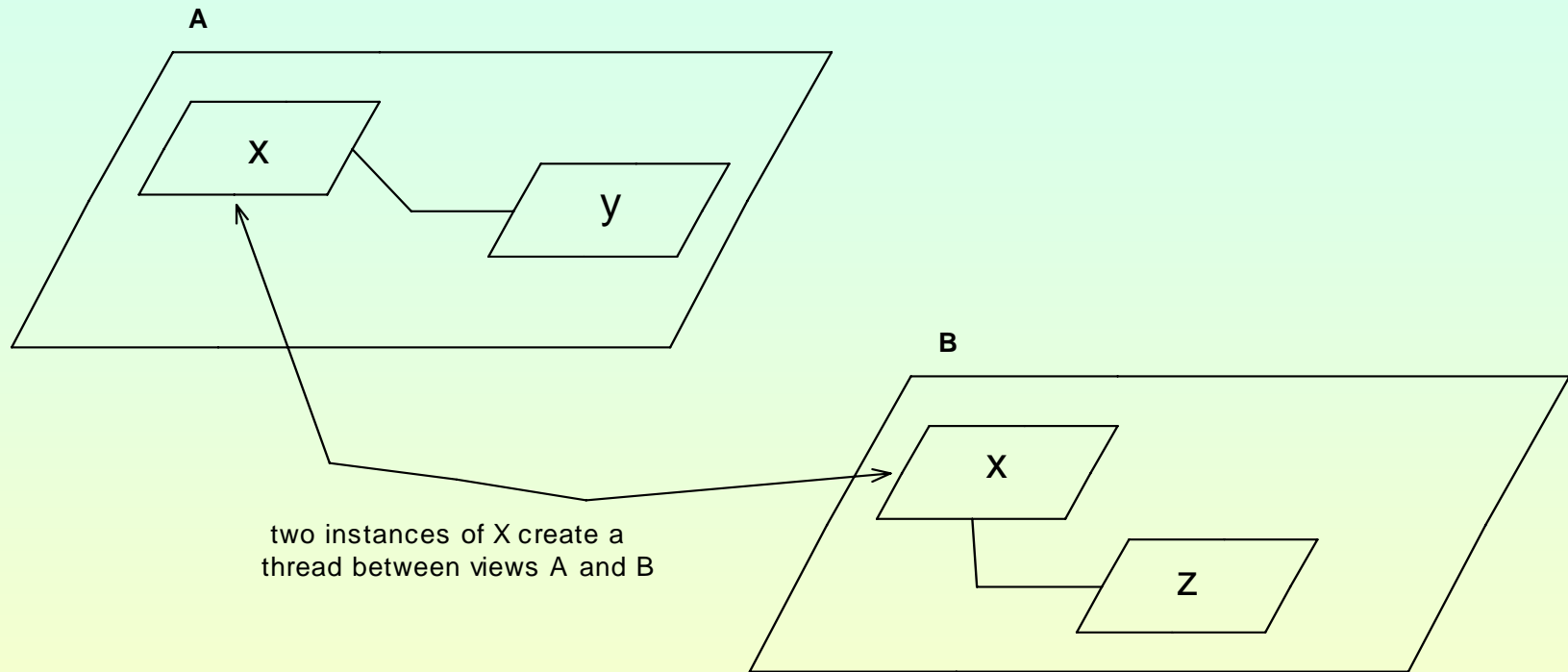
# Multiple representations, cont'd

- Can be created using
  - Different modeling techniques applied to the same subject
    - E.g. conceptual model and screen mock-up
  - Customized versions of a diagram
  - Elements shown differently depending on the goal
  - Elements that morph according to the context
  - ...

# Aspect: Threaded views

- Same element may be needed in several views
  - Requirement “*Proceeds from trading operation shall be deposited at the end of each quarter*” belongs to:
    - the description of the trading operation
    - the definition of the process scheduled for execution at the end of each quarter
  - Concept like account, security, are used in many places
  - Actor may be present on a use cases view and on an architecture view
  - Etc.

# Threaded views, cont'd



# Threading

- Exists whether we want it or not
- Valuable for consistency
  - One Change, One Place (OCOP)
  - Allows to work with various requirement's contexts
  - Helps to understand views and their relations
- Needs to be supported by the tool
  - Usual in modeling tools, but not in text processors or requirements tools, at least not at the level of requirements elements

# Bottom line

*Apply appropriate **M**odeling techniques*

*Define requirements **A**rchitecture*

*Choose appropriate **R**epresentations*

# References

1. Diev, Sergey. Requirements development as a modeling activity.  
ACM SIGSOFT Software Engineering Notes. 2007, Vol. 32, Issue 2 (March), 3 pages
2. Diev, Sergey. Structuring complex requirements.  
ACM SIGSOFT Software Engineering Notes. 2007, Vol. 32, Issue 2 (March), 5pp.
3. Diev, Sergey. Querying complex requirements.  
ACM SIGSOFT Software Engineering Notes. 2009, Vol. 34, Issue 1 (January), 7pp.