Functional Modeling – Requirement Patterns (Problem Frames)

Gregor v. Bochmann, University of Ottawa

Based on Powerpoint slides by Gunter Mussbacher
with material from:
K.E. Wiegers, D. Leffingwell & D. Widrig, M. Jackson, I.K. Bray, B. Selic,
Volere, Telelogic, D. Damian, S. Somé 2008, and D. Amyot 2008-2009

Table of Contents

The section on Problem Frames is based on the information in the book by Bray (Section 4.5)

Modeling in requirements engineering (1)

- "Analysis" in the context of requirements engineering has to deal with the analysis of the problems in the problem domain, and how they could be solved by a proposed new organization of the problem domain (the *problem domain to-be*), normally including a new "system" to be developed. It is not only analysing the "old system" – therefore it is sometimes called **Problem Domain Analysis (PDOA).**
- The development of the new organization of the problem domain is sometimes called "external design" or **e-design**. This activity then normally leads to the development of the specification of the new "system" (the **system-to-be**).
- Both of these activities assume that enough illicitation activities have been performed in order to obtain all pertinent information about the existing problem domain and the possible solutions to the problems.
- How can models help ?

Modeling in requirements engineering: Example A

- Example A
  - **Problem statement:** A large building is used by a government department as workplace for its employees. There are offices, one large meeting place, and a cafeteria. The employees often have to meet other employees and therefore have to move around the building. The building only has stairs to move between floors. The problem is that the stairs are often overcrowded and people (in 2010) are tired of walking stairs up and down. Some budget for renovating the building is available.
  - **Proposed solution:** Install one of the following systems:
    - Lifts
    - Escalators
    - Paternosters (see http://en.wikipedia.org/wiki/Paternoster)
  - The development of the specification of the system-to-be implies a design (external design) decision: which kind of system should be selected. Possible criteria: performance, user interface
  - For evaluating performance, one may establish a performance model of the problem domain (including the selected type of system), taking into account the structure of the building, and the capacity of the load in normal and peak hours (lunch time, or end of a large meeting).

Modeling in requirements engineering: Example B

- Example B
  - This is a continuation of Example A. Let us assume that the renovation included a set of elevators. And assume that the elevator hardware has already been selected. However, the behavior of the elevator control software must be determined in order to satisfy certain requirements.
  - **These requirements are (see book by Bray for more details):**
    - Certain requirements of the hardware (e.g. do not go from fast down movement to fast up)
    - Certain user requirements (e.g. a lift should arrive when one pushes the button at a given floor)
  - **Proposed specification (see book by Bray for more details):**
    - The system specification includes a state machine model showing in which order commands are sent to the motor of a lift in response to inputs from buttons – this is a model of the system-to-be.

Modeling in requirements engineering – different types (1)

One can distinguish different types of models:

- **Structural models:** define (essentially static) structure of the
  - **problem domain** as-is, or the problem domain to-be (including the system-to-be)
  - **system-to-be** including its interfaces and possibly some major components – but note: avoid internal design
- **Behavioral models:** define the dynamic behavior of the
  - **system-to-be** – it is defined as a black box – behavior is defined by the interactions at the interfaces and their possible order of execution
    - Note: The system-to-be is an “open system” that communicates with its environment
  - **problem domain** as-is, or the problem domain to-be (including the system-to-be)
    - Note: The problem domain can often be considered as a “closed system”, that is, a system that does not communicate with its environment. Therefore the behavior must be defined in terms of the interactions between the different components (entities) within the domain.

Modeling in requirements engineering – different types (2)

- The Behavioral models can be classified into
  - Definition of example behaviors - possible techniques:
    - Use cases
    - Use case maps or UML Activity diagrams
    - Sequence diagrams
  - Definition of complete behavior (for a certain aspect)
    - State machines
    - UML Activity diagrams
- **Performance models:** The complete behavior models can be extended to include performance characteristics – thus providing the possibility of evaluating the expected performance of the system-to-be

Problem Domain Analysis (PDOA) using Problem Frames

- The importance of considering the whole problem domain during requirements analaysis was stressed by Michael Jackson, who developed the **Problem Frame** approach to PDOA (1995, 2000)
  - "A problem frame is a kind of pattern. It defines an intuitively identifiable problem in terms of its *context* and the characteristics of its *domains*, *interfaces* and *requirements*. Domain and interface characteristics are based on a classification of *phenomena*."[1]

[1] Jackson, M.A., "Problem Frames", Addison-Wesley, 2000, p.76

Problem Frames

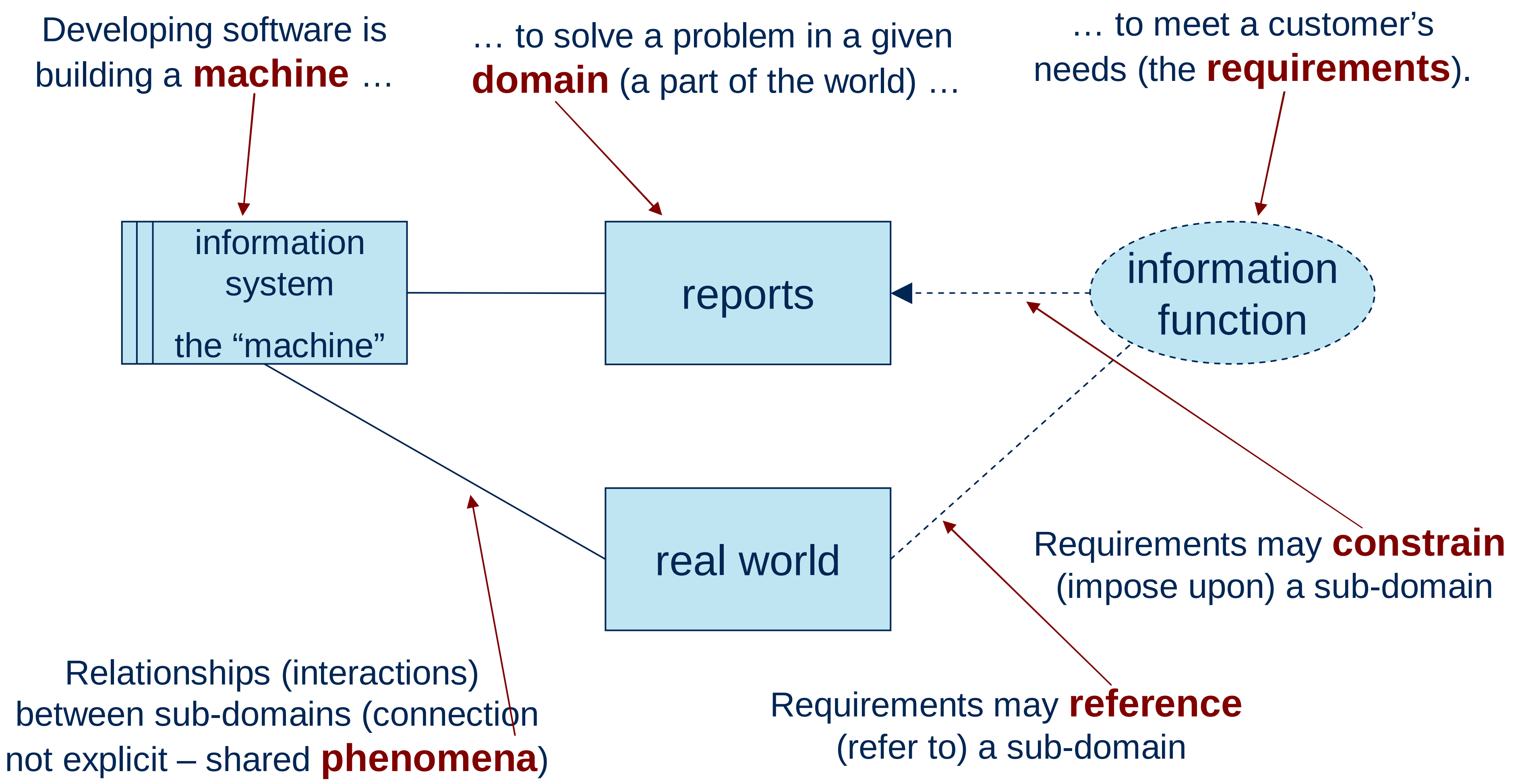
- Idea: categorize and characterize patterns of problems (called “Problem Frames”)
  - Useful for describing and understanding the problem domain
  - For each pattern, some guidelines are given about how to develop requirements for this type of situation
  - Basically, the idea is similar to the idea of design patterns
- Proposed methodology: given a problem, find the appropriate pattern and then develop requirements according to the guidelines for that pattern.
  - Rationale: Some methods work well for certain classes of problems

Requirements patterns

- The following Problem Frames have been identified by Jackson (there may be more) :
  - Information
    - Continuous display
    - Requested reports
  - Control
    - Required behaviour
    - Commanded behaviour
  - Workpiece
  - Transformation
  - Connection
- **Each Problem Frame is characterized by a diagram that involves a certain number of sub-domains of the problem domain, the system-to-be (a special sub-domain), relations between these sub-domains and requirements (that may reference sub-domains)**

Example: Information Frame

- An information system exists in order to provide information about some part of the problem domain.

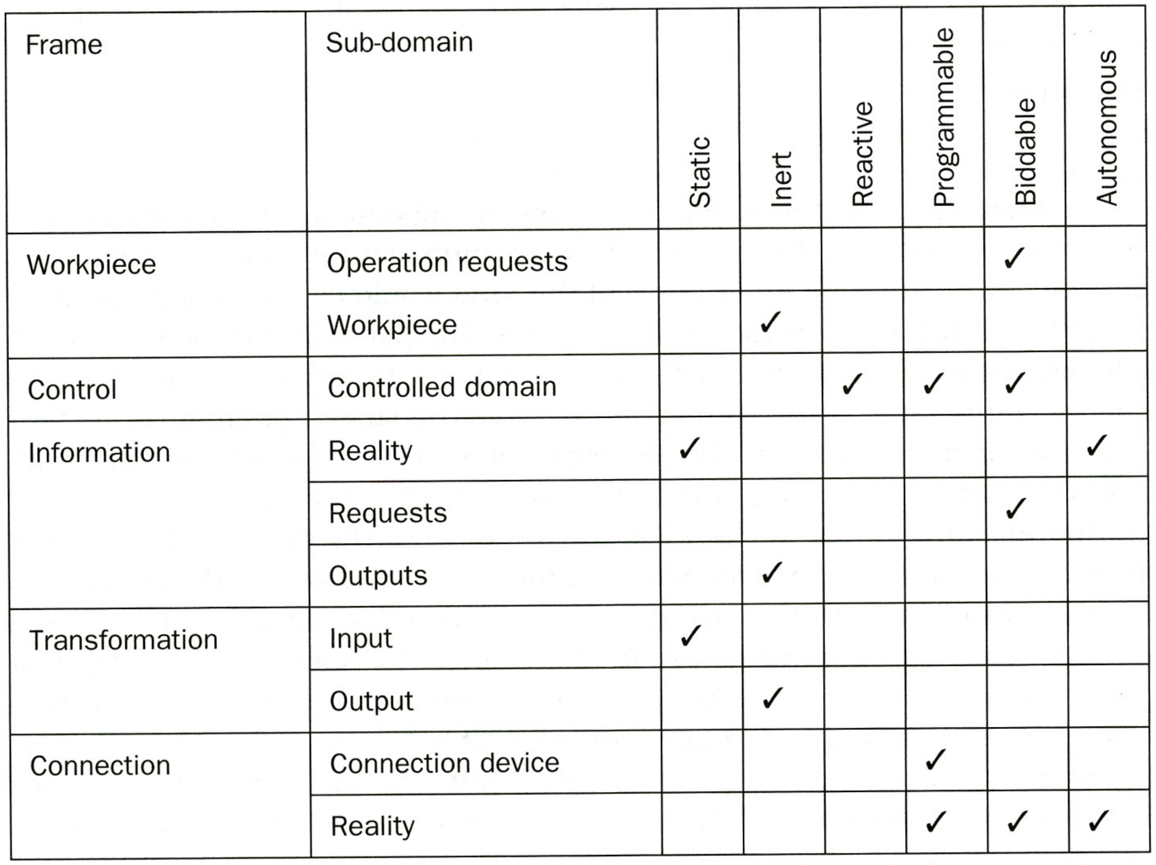

Developing software is building a **machine** …
… to solve a problem in a given **domain** (a part of the world) …
… to meet a customer's needs (the **requirements**).
information system
the "machine"
reports
information function
real world
Requirements may **constrain** (impose upon) a sub-domain
Requirements may **reference** (refer to) a sub-domain
Relationships (interactions) between sub-domains (connection not explicit – shared **phenomena**)

Problem Frames - generalities

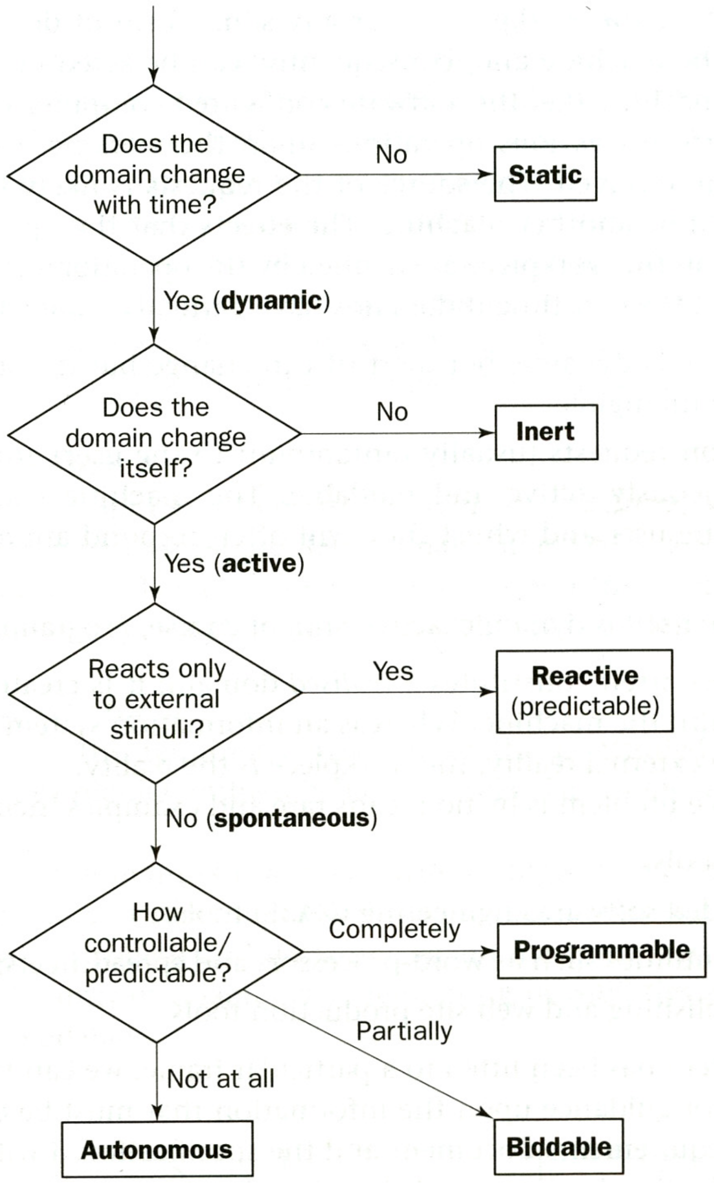
- One problem - one "**machine**" (agent that is part of the system-to-be)
  - The machine is a general purpose computer specialized by software
  - The machine may be distributed
  - One real computer may support many "machines"
- Problem decomposition may result in many sub-problems and therefore many machines – the we have a multi-frame problem.
- The machine and the problem domain interact at an interface of shared phenomena (events, states, etc.)
- The customer's requirement is for some effect (or property or behavior) in the problem domain
  - Requirements add a constraint to the sub-domains' intrinsic properties or behaviors

Types of sub-domains

- Different types of sub-domains are distinguished
- Only certain types are allowed as sub-domains in a given pattern:

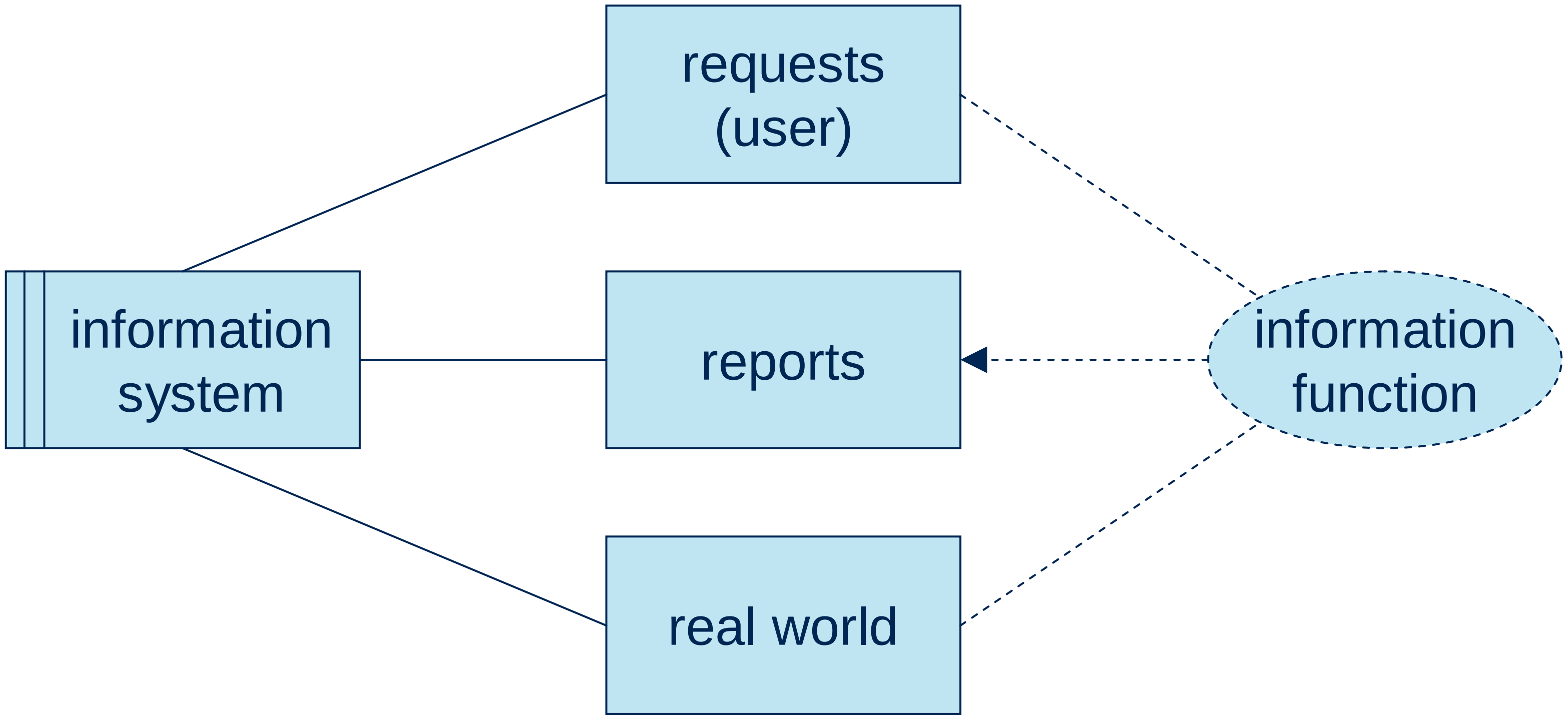


Frame	Sub-domain	Static	Inert	Reactive	Programmable	Biddable	Autonomous
Workpiece	Operation requests					✓	
	Workpiece		✓				
Control	Controlled domain			✓	✓	✓	
Information	Reality	✓					✓
	Requests					✓	
	Outputs		✓				
Transformation	Input	✓					
	Output		✓				
Connection	Connection device				✓		
	Reality				✓	✓	✓

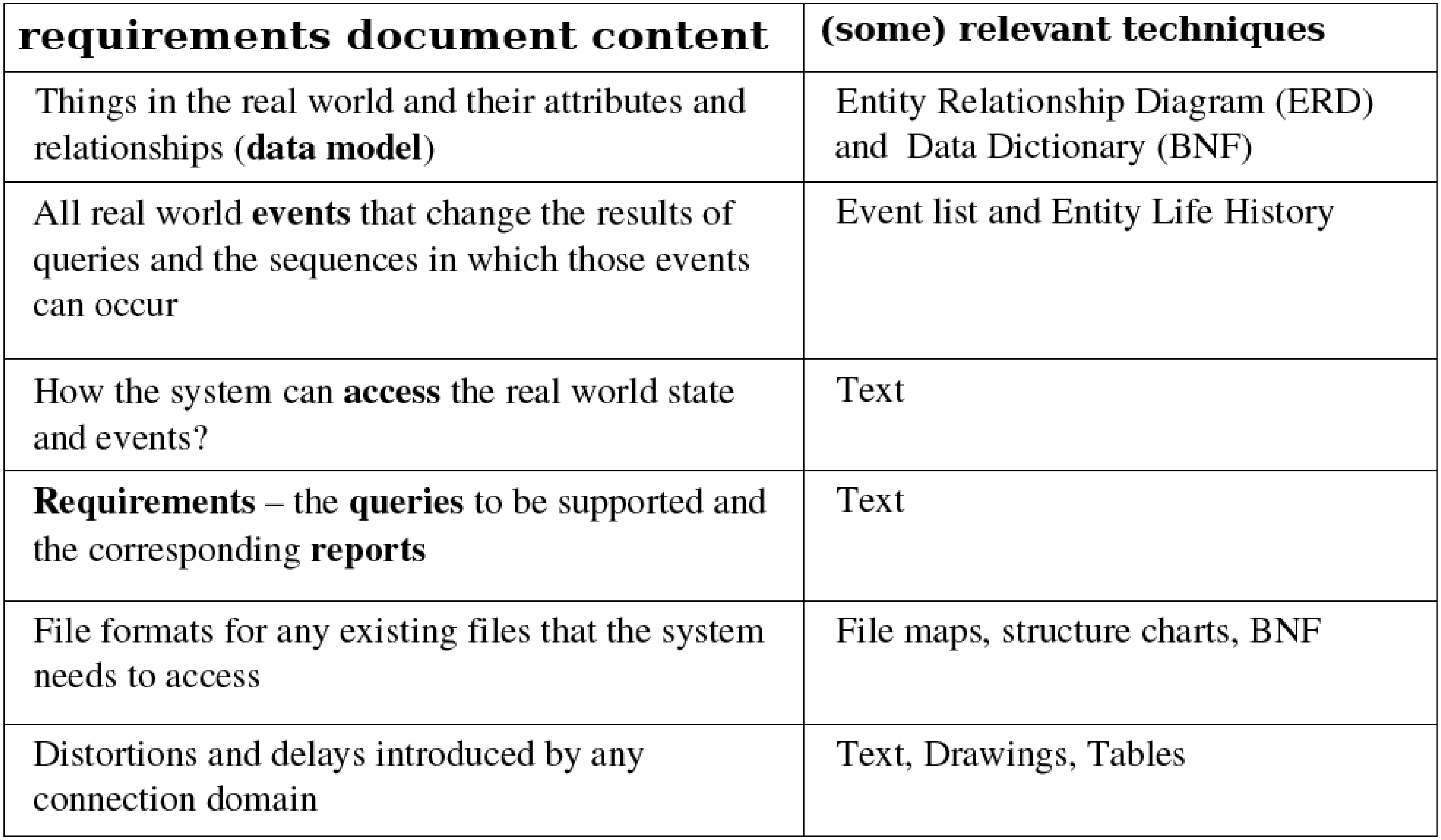


Information Frame - version (b)

- Pattern (a) - continuous display: automatically provide information about some part of reality
  - See above. Same as below but without the *requests* domain
- Pattern (b) - requested reports: provide information about some part of reality upon request



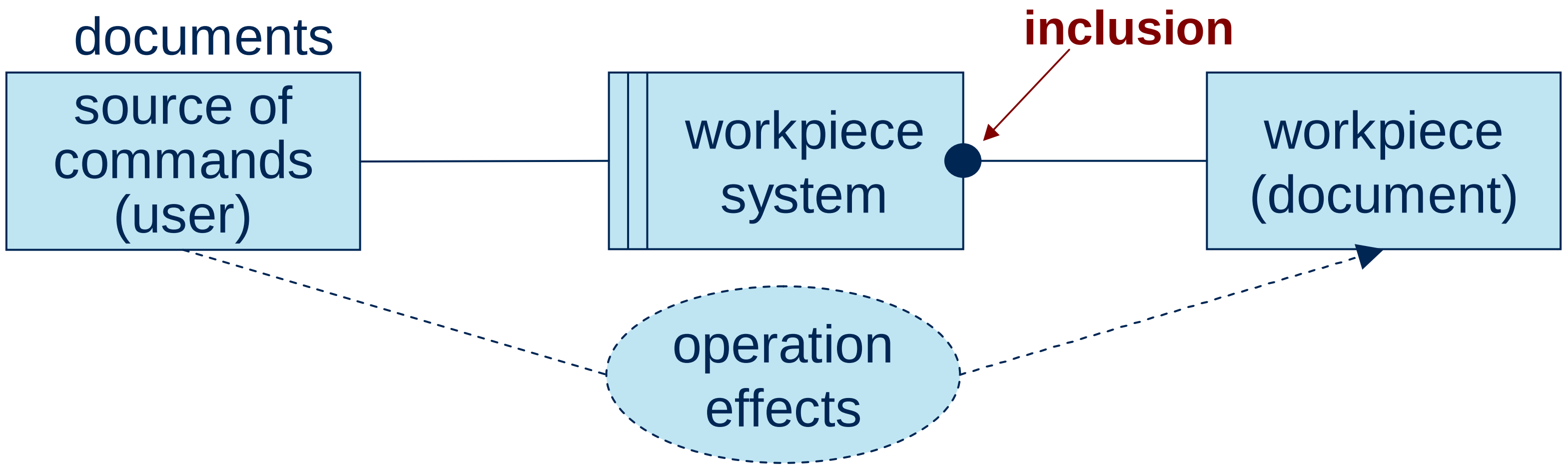
Guideline for Information Frame (called Kovitz Table)



requirements document content	(some) relevant techniques
Things in the real world and their attributes and relationships (**data model**)	Entity Relationship Diagram (ERD) and Data Dictionary (BNF)
All real world **events** that change the results of queries and the sequences in which those events can occur	Event list and Entity Life History
How the system can **access** the real world state and events?	Text
Requirements – the **queries** to be supported and the corresponding **reports**	Text
File formats for any existing files that the system needs to access	File maps, structure charts, BNF
Distortions and delays introduced by any connection domain	Text, Drawings, Tables

Workpiece Problem Frame

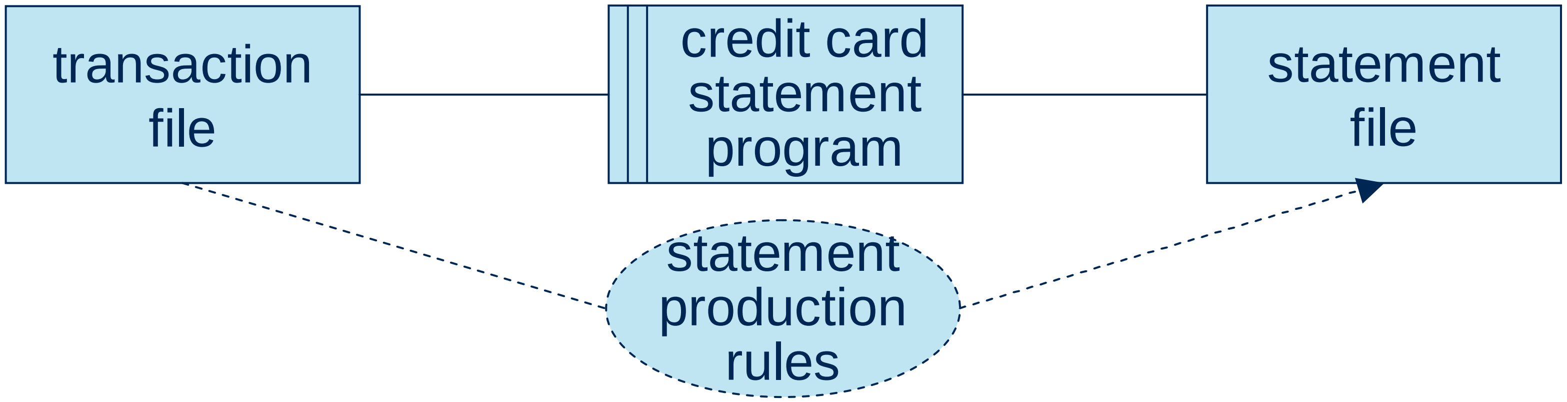
- Pattern: perform directed operations upon objects that exist only within the system
  - e.g., creating and editing documents, UML models, programs, other documents



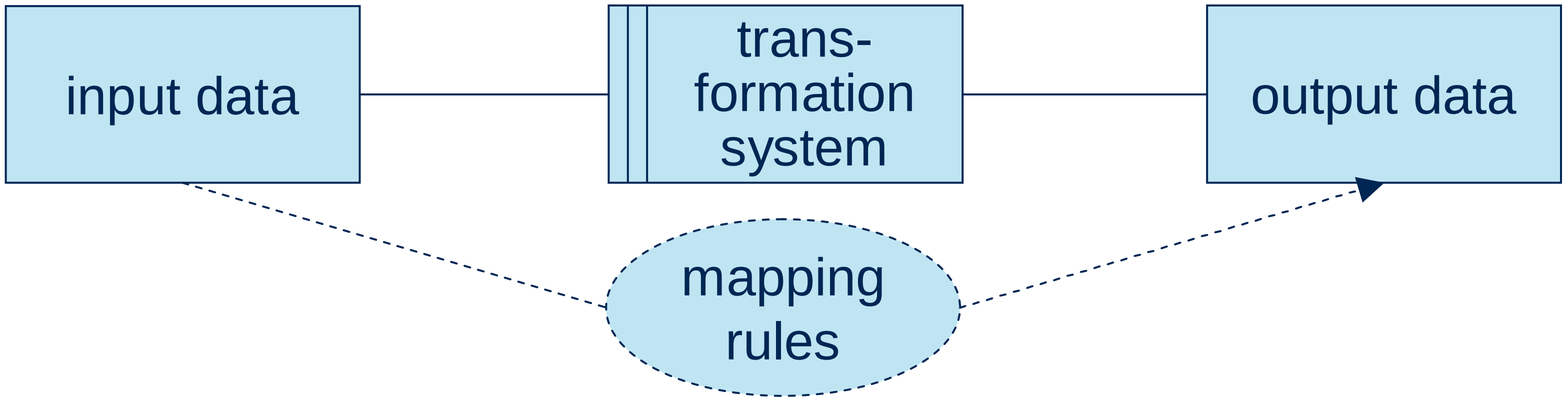
requirements document content	(some) relevant techniques
Workpiece – legitimate, logical structure as evident to outside world	BNF, File map, Structure chart
Requirements – the required commands/operations and the effects that the operations should have upon the workpiece	FSM/STD, text, Decision table, Use Case

Transformation Pattern

- Example: program that produces a credit card statement, given a file of transactions

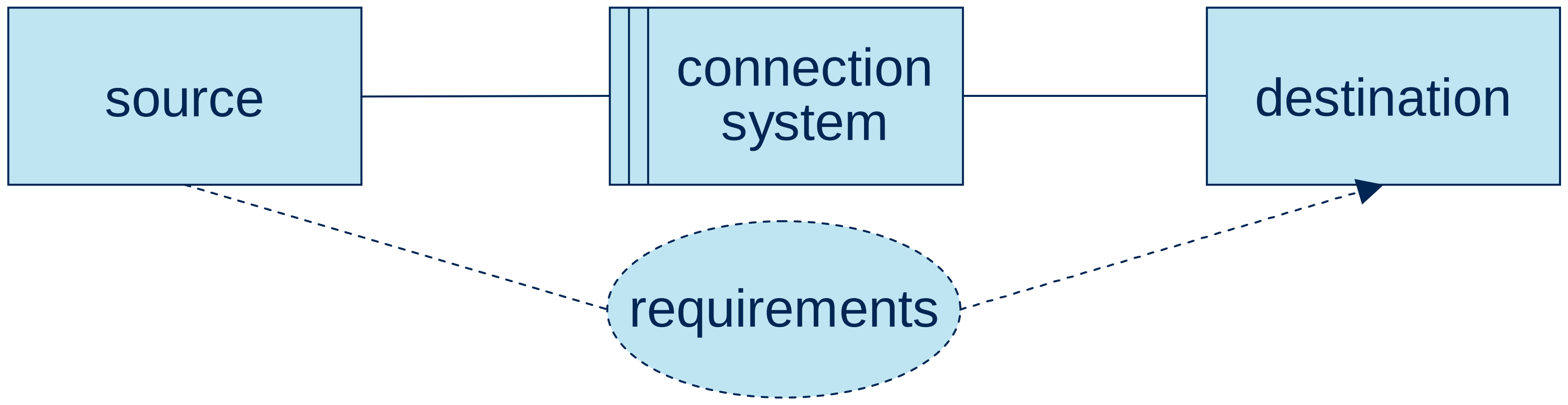


- Pattern: transform given input data to produce new output data according to some predefined rules



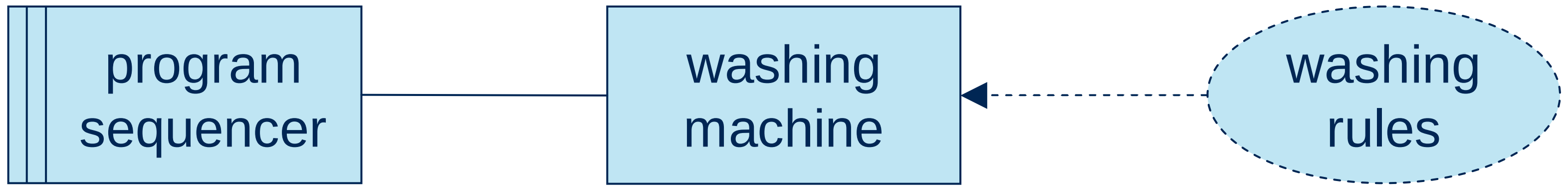
Connection Pattern

- Resembles the transformation frame but there is a crucial difference
  - Transformation system generates new data from old
  - Connection system just moves (logical) data from A to B
- The requirements will concern the acceptable delays and distortions that the system may introduce

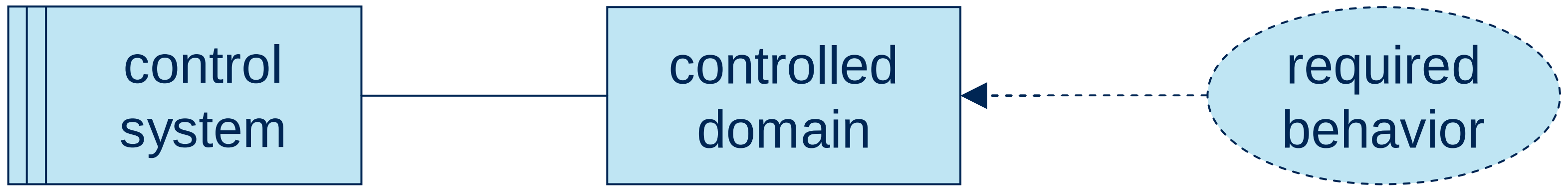


Control Pattern, version (a) – Required Behavior

- Example: system that controls the various programs of a washing machine

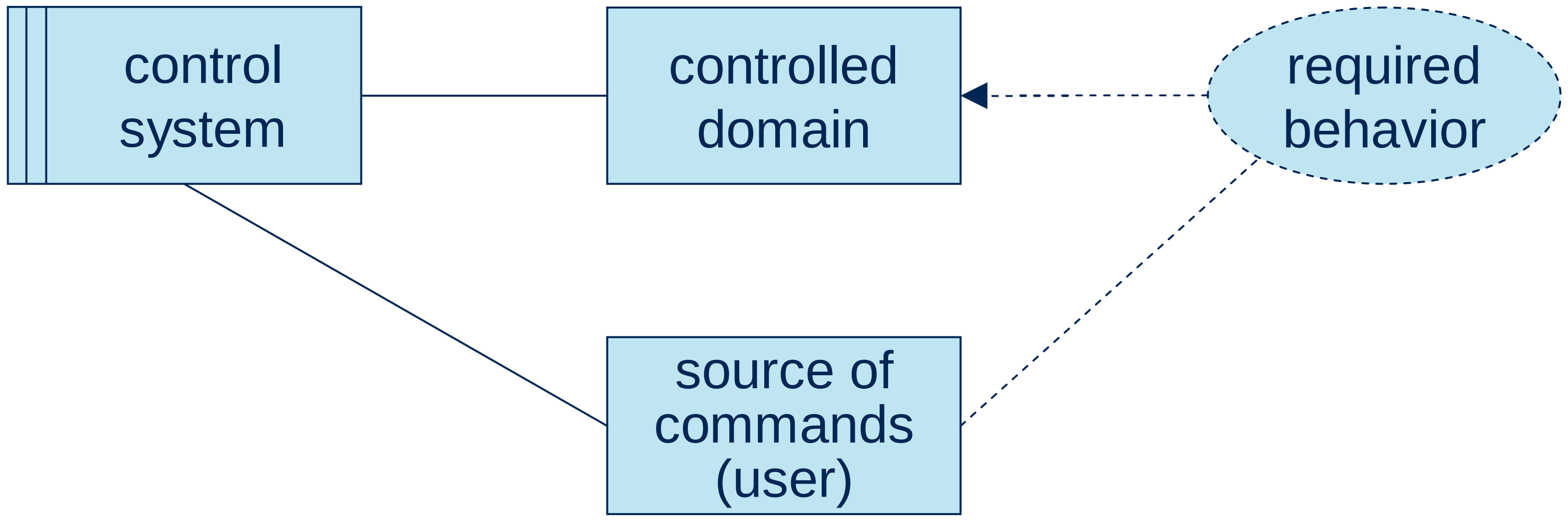


- Pattern: control some part of reality according to defined rules



Control Pattern, version (b) – Commanded Behavior

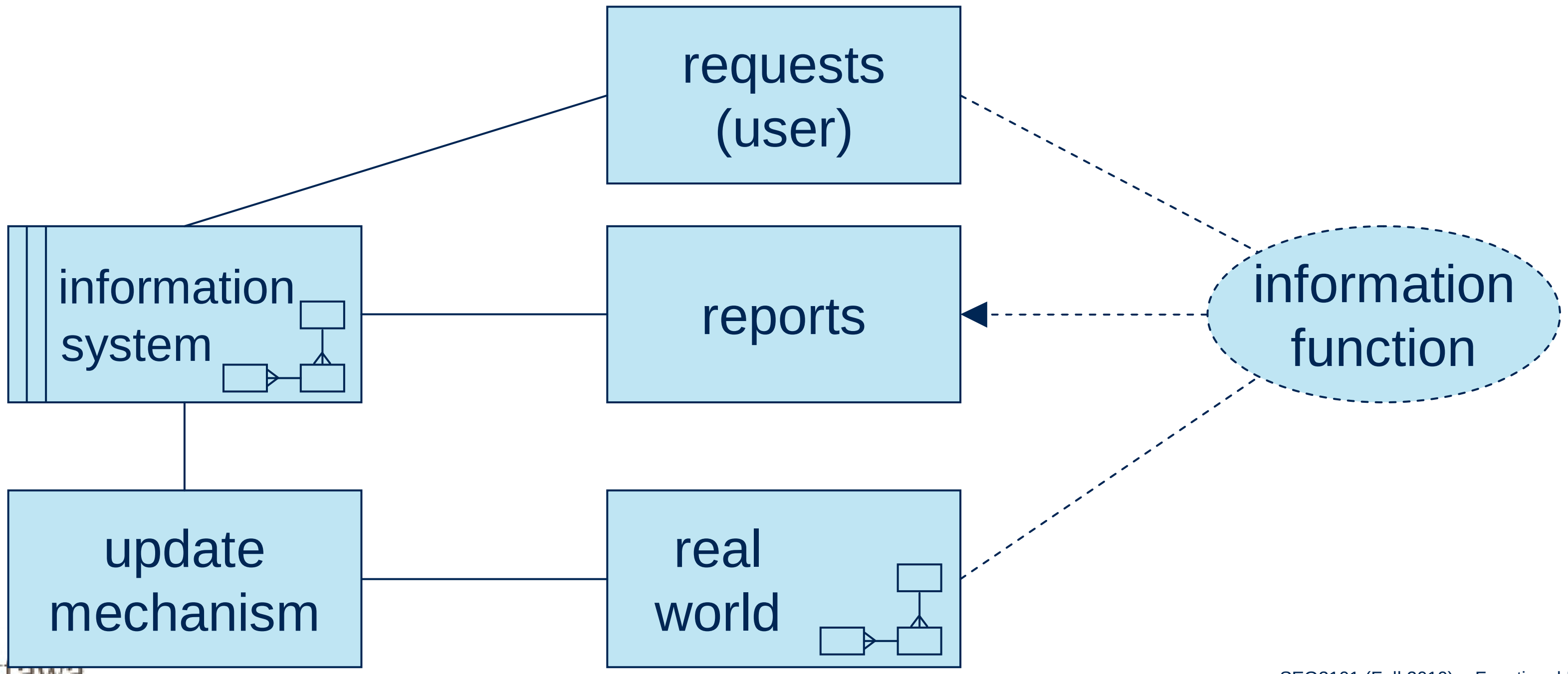
- Pattern: control some part of reality according to operator commands



- Examples: engine management system, process control...

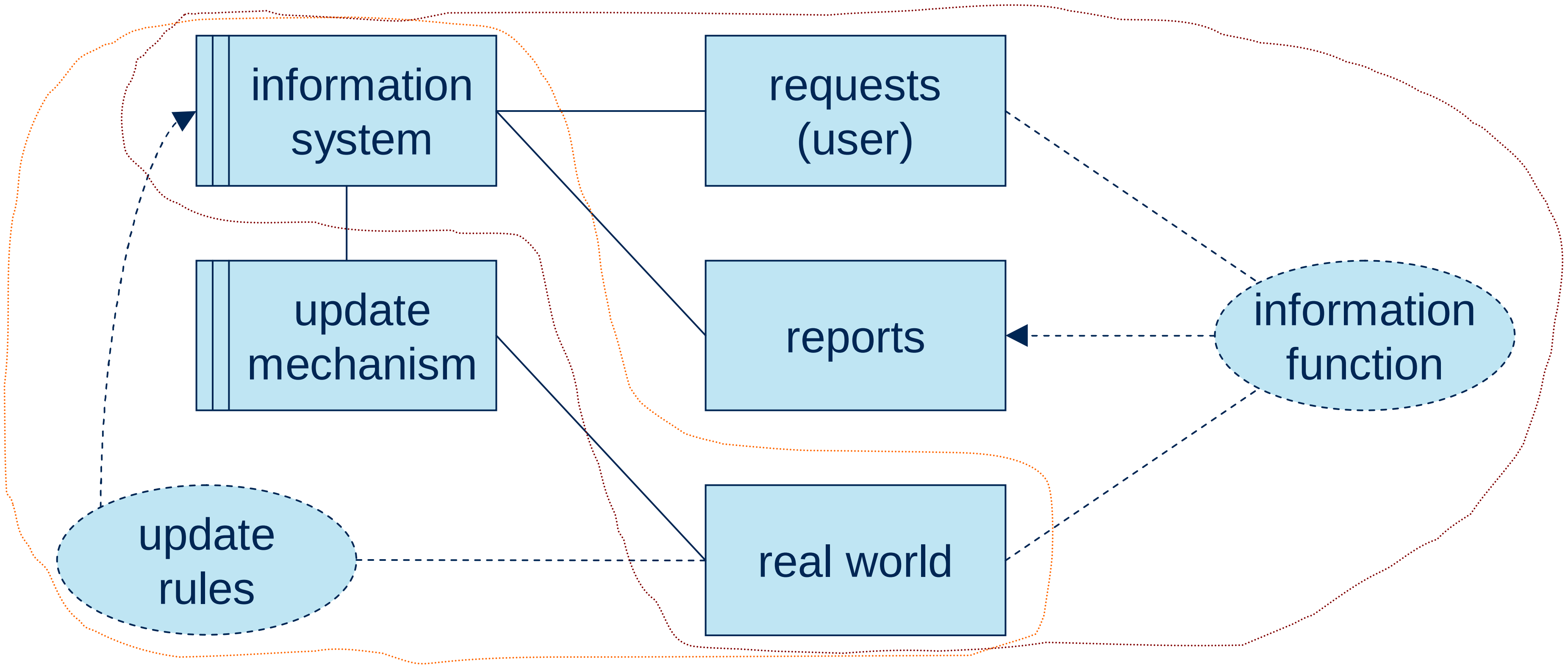
Information Problem – a multi-frame problem

- For an information system, it is the real world's data that is significant
- An information system will often contain a model of the real world data – but note that there are then two models of the real world during the development of the requirements: the real world (model) and the model of the real world in the information system. They may vary in structure and value!



Example of a Multi-Frame Problem

- The example of the previous slide can be seen as a multi-frame problem. The **update mechanism** represents a **connection pattern**.
  - The problem must be partitioned and (overlapping) frames fitted to the recognizable parts



Frame-Based Guidelines (Kovitz Tables)

- The classification into problem frames helps with the development process by
  - Providing specific guidance for each type of problem
  - Decomposing complex problems in a logical way
- Problem frames are used to characterise the **problem**
  - The problem frame approach concentrates on the problem context, that is, the whole problem domain – not only the immediate context of the system-to-be, sometimes called the "solution system context"
  - It also models the relationships between sub-domains
- Kovitz tables are used to determine what aspects need to be further investigated and documented
  - Depending on the type of problem frame, identifies the aspects to be documented (to be illicitated)
  - Suggest modelling techniques
- For more details, see book by Bray