

Conceptual modeling in agile information systems development

Antoni Olivé
Universitat Politècnica Catalunya-BarcelonaTech



Questions without definitive answers

What is it?

Is conceptual modeling
a necessary or an optional
activity in information systems
development?
agile



Motivation

“The main purpose of conceptual modeling is to *improve communication* between the parties involved in the development process”

“Conceptual data modeling is an *indispensable part* of information system design and development”

Motivation

“...*might be* used to facilitate the design and implementation of an information system”

“Once you have the conceptual design, all the other design and implementation activities can and *should* be grounded in it...”

Manifesto for agile software development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to **value**:

...

Working software over comprehensive documentation

...

That is, while there is value in the items on the right, we **value the items on the left more**.



Motivation

To: antoni.olive

From: a student

Body:

Dear professor,

I've started to develop my final career project in the company.

Here they use agile development, and they do not generate any kind of documentation, and the **requirements engineering** and overall design are **not very deep** (I understand that this is their development method).

My idea of the project is to approach it in this way...



Objective of the talk

Is conceptual modeling
a **necessary** or an **optional**
activity in information systems
development?

agile

Implications

In (agile) information systems development

Practice

Do we need to perform conceptual modeling?

Teaching

Do we have to teach conceptual modeling?

Research

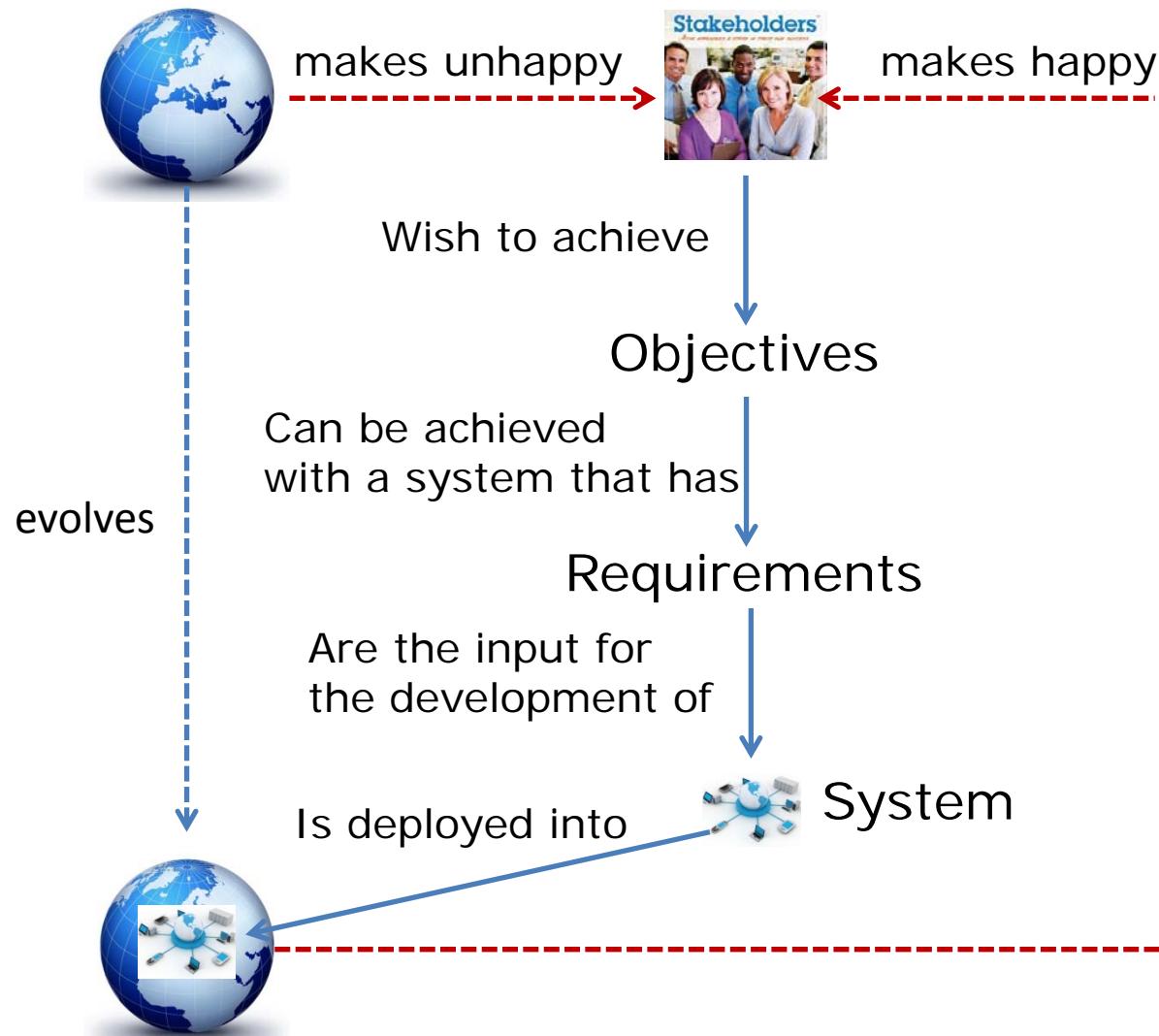
Which is the nature and role of conceptual modeling?



Outline

- Back to basics: the need of **requirements**
- What are the **conceptual schemas**? Do we really need them?
- The main driving force of **formal** conceptual schemas
- Conceptual modeling in **agile** development. Is it needed?
- Conclusions

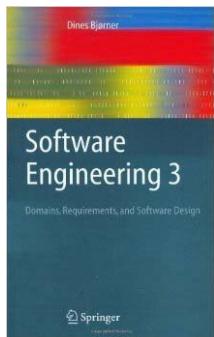
The system's lifecycle



You cannot develop a system unless you know its requirements

The principle of necessity of requirements

To develop an information system
it is necessary
to define its requirements



Before software can be designed, programmed, coded,
its requirements **must** first be reasonably well understood.

Are there exceptions?



Infinite monkey theorem



A monkey

- hitting keys at random on a typewriter keyboard
- for an **infinite** amount of time
- will almost surely type a given text,
- such as the **complete works of William Shakespeare.**

Infinite programmer theorem



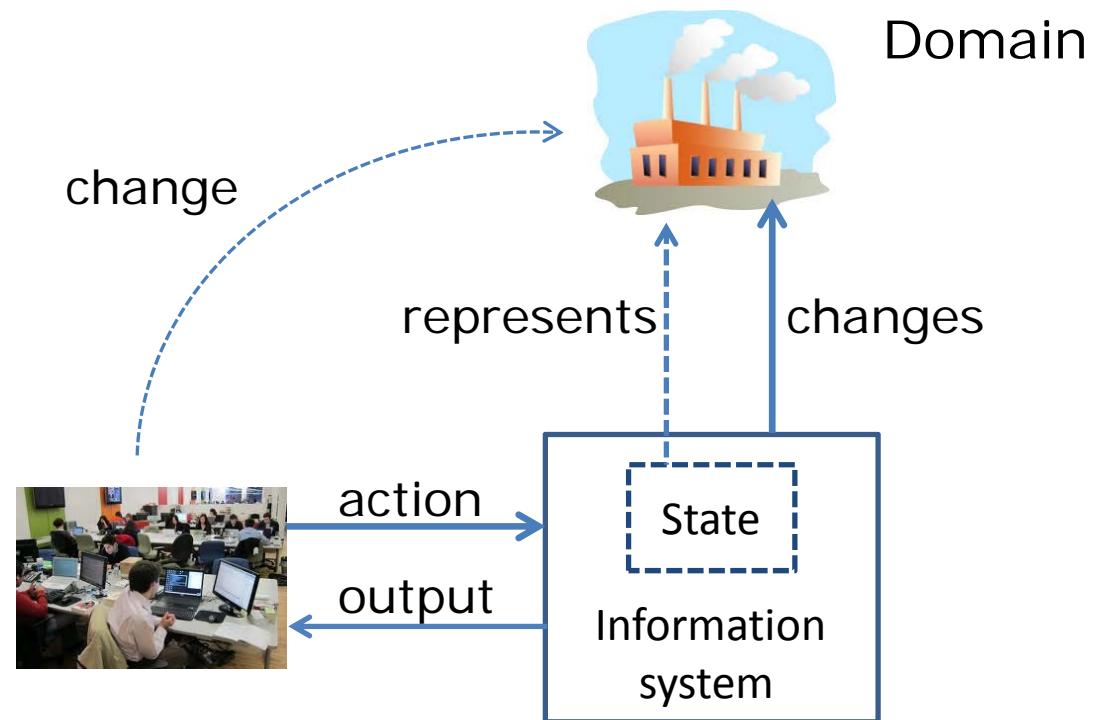
A programmer

- hitting keys at random on a typewriter keyboard
- for an **infinite** amount of time
- will almost surely type a program,
- that **satisfies the stakeholders' needs**

Two types of requirements

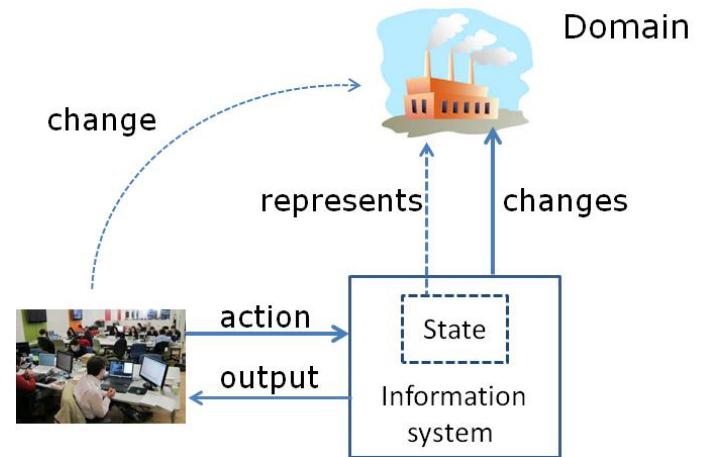
- Functional
 - Quality
- 
- The focus of this talk

The functions of an information system



Functional requirements

- Domain concepts represented in the IS
- Definitions
- Integrity Constraints
- Events/Actions:
 - Triggering conditions
 - Constraints
 - Effect on the state
 - Output



The principle of necessity of functional requirements

To develop an information system
it is necessary
to define its functional requirements

Outline

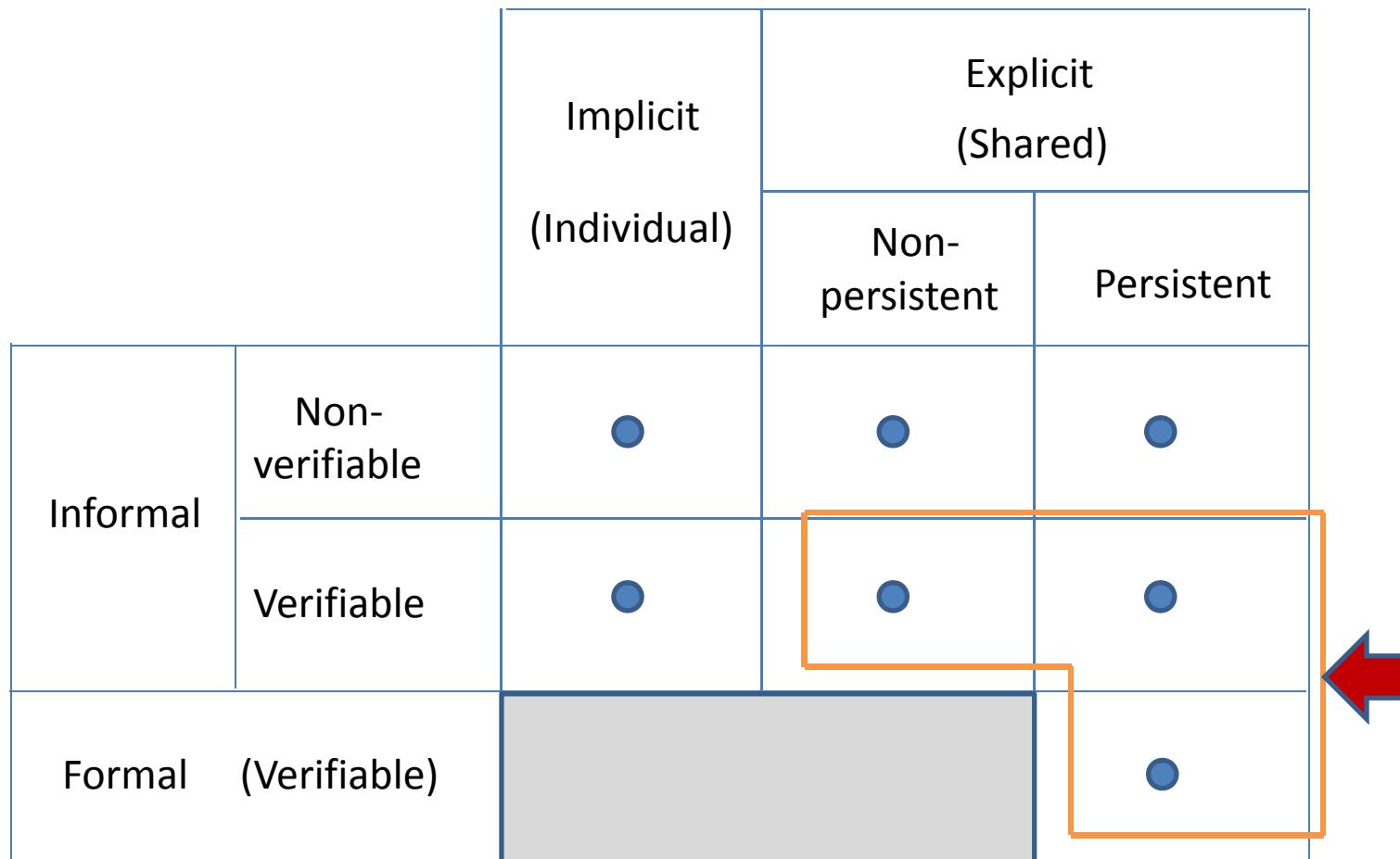
- Back to basics: the need of **requirements**

 What are the **conceptual schemas**? Do we really need them?

- The main driving force of **formal** conceptual schemas
- Conceptual modelling in **agile** development. Is it needed?
- Conclusions

Classification of functional requirements

		Implicit (Individual)	Explicit (Shared)	
			Non-persistent	Persistent
Informal	Non-verifiable	●	●	●
	Verifiable	●	●	●
Formal (Verifiable)				



Four key classifications of functional requirements

- Formality
- Verifiability
- Explicitness
- Persistency



Formality

Natural language	User stories	Structured natural language	UML/OCL	Logic
------------------	--------------	-----------------------------	---------	-------

Informal

Semi-formal

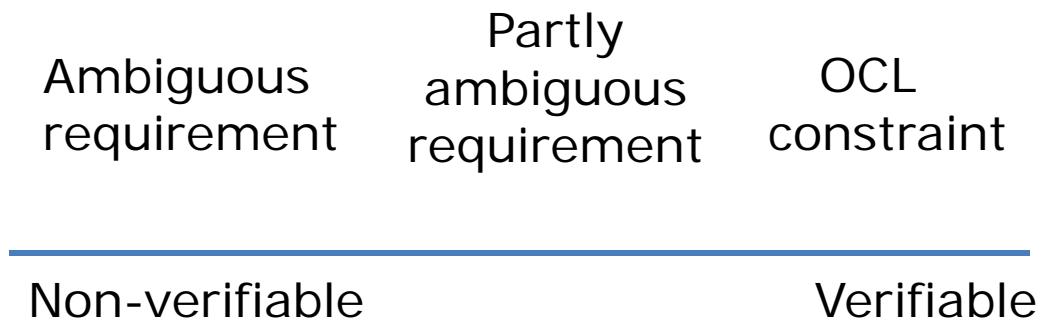
Formal

Degree of formality of the language

Verifiability

Verifiable:

A person or a machine can check that the software meets the requirement



Explicitness

Explicit: Made public in some form

Assumed
requirements

SRS
document

Implicit

Explicit



Persistency

Verbal

Paper document

Computer file

Non-persistent

Persistent

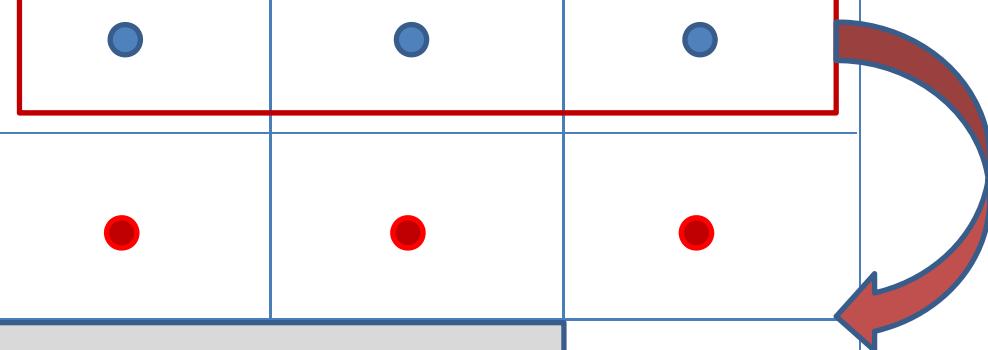


Classification of functional requirements

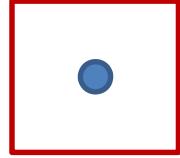
		Implicit (Individual)	Explicit (Shared)	
			Non-persistent	Persistent
Informal	Non-verifiable			
	Verifiable			
Formal (Verifiable)				

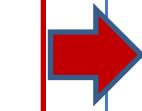
Functional requirements to be implemented must be made verifiable

		Implicit (Individual)	Explicit (Shared)	
			Non-persistent	Persistent
Informal	Non-verifiable	●	●	●
	Verifiable	●	●	●
Formal (Verifiable)				●



Functional requirements to be implemented must be made explicit

		Implicit (Individual)	Explicit (Shared)	
			Non-persistent	Persistent
Informal	Non-verifiable			
	Verifiable			
Formal (Verifiable)				



Conceptual schema = Explicit, verifiable functional reqs.

		Implicit (Individual)	Explicit (Shared)	
			Non-persistent	Persistent
Informal	Non-verifiable			
	Verifiable		●	●
Formal (Verifiable)		●		

The diagram illustrates a conceptual schema classification matrix. It consists of a 5x5 grid where the first two columns and the first row are defined by the axes of the matrix. The remaining three columns and two rows represent specific schema types. The matrix is divided into four quadrants by a diagonal line from the top-left to the bottom-right. The bottom-right quadrant contains three blue dots, each enclosed in a small square. An orange rectangle surrounds the top-right quadrant (the second column and third row). A red arrow points to the bottom-right dot in this quadrant.

The principle of necessity of conceptual schemas

To develop an information system
it is necessary
to define its **functional requirements**



To develop an information system
it is necessary
to define its **conceptual schema**



“Official” definitions

...conceptual modelers:

- (1) describe structure models in terms of entities, relationships, and constraints;
- (2) describe behavior or functional models in terms of states, transitions among states, and actions performed in states and transitions; and
- (3) describe interactions and user interfaces in terms of messages sent and received and information exchanged.

<http://www.conceptualmodeling.org/ConceptualModeling.html>



“Official” definitions

... conceptual-model diagrams:

- ...
- in some cases automatically generate (parts of) the software application.

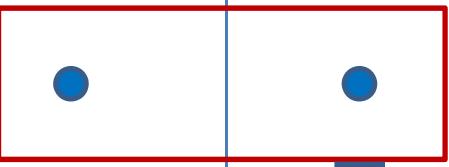
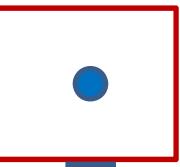
<http://www.conceptualmodeling.org/ConceptualModeling.html>



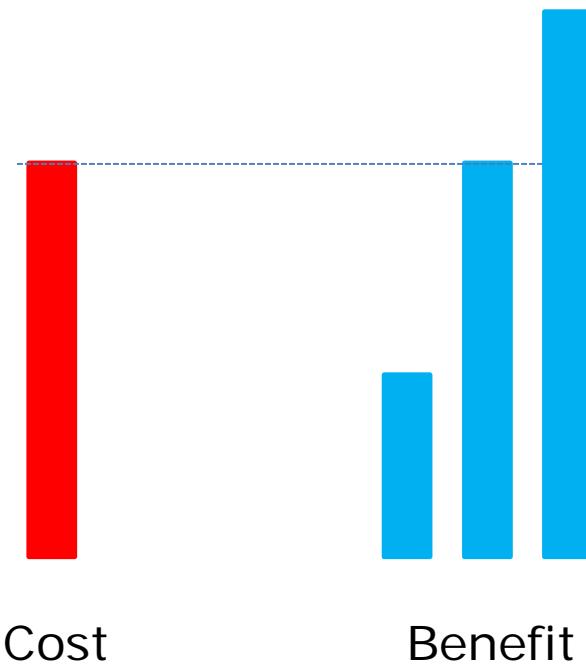
Outline

- Back to basics: the need of **requirements**
- What are the **conceptual schemas**? Do we really need them?
 The main driving force of **formal** conceptual schemas
 - Conceptual modeling in **agile** development. Is it needed?
 - Conclusions

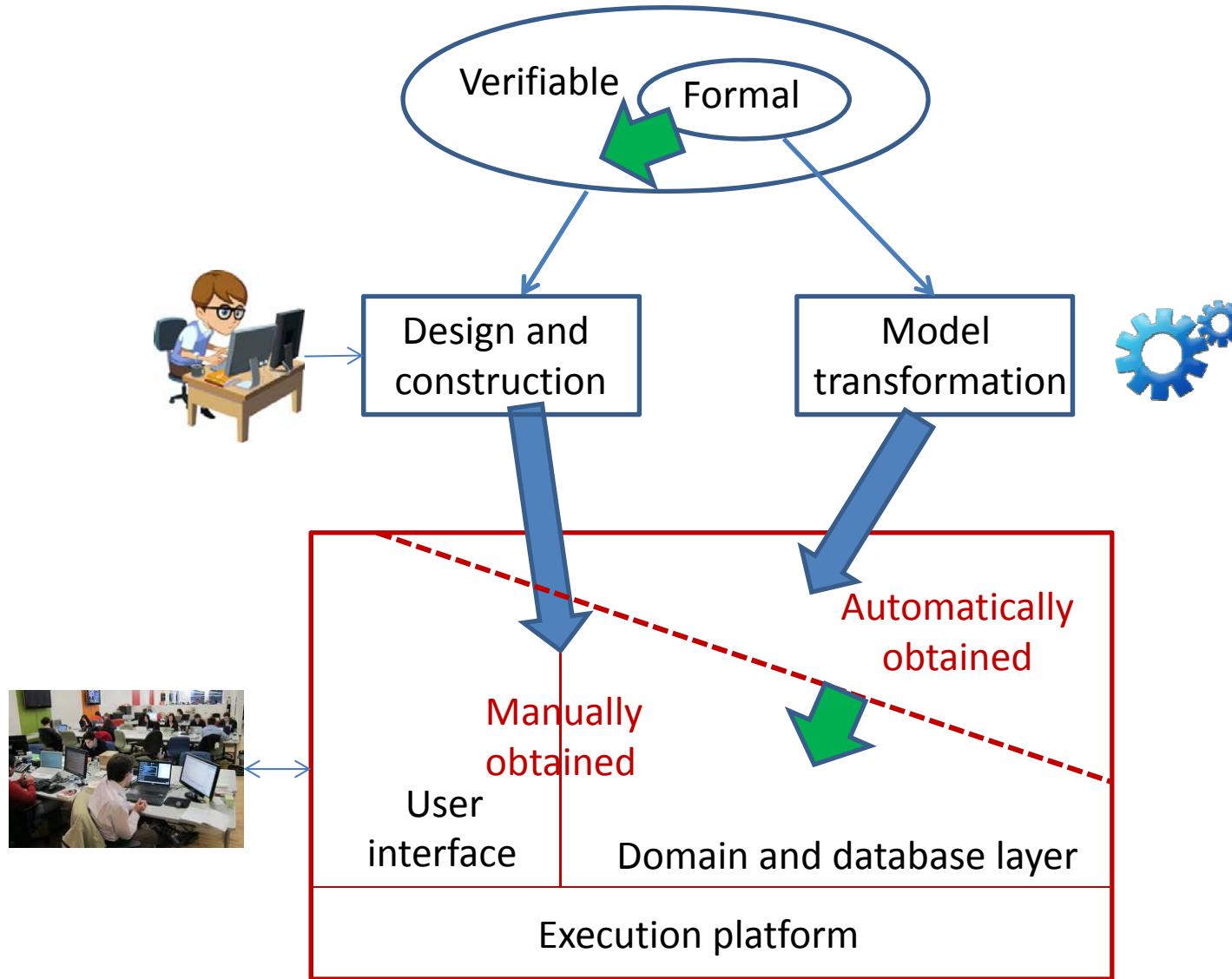
Conceptual schemas: Formal or informal?

		Implicit (Individual)	Explicit (Shared)	
			Non-persistent	Persistent
Informal	Non-verifiable			
	Verifiable			 
Formal (Verifiable)				

Formalization? A cost/benefit analysis



From requirements to implementation



Outline

- Back to basics: the need of **requirements**
 - What are the **conceptual schemas**? Do we really need them?
 - The main driving force of **formal** conceptual schemas
-  Conceptual modeling in **agile** development. Is it needed?
- Conclusions

Manifesto for agile software development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to **value**:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we **value the items on the left more**.



Conceptual modeling in agile software development

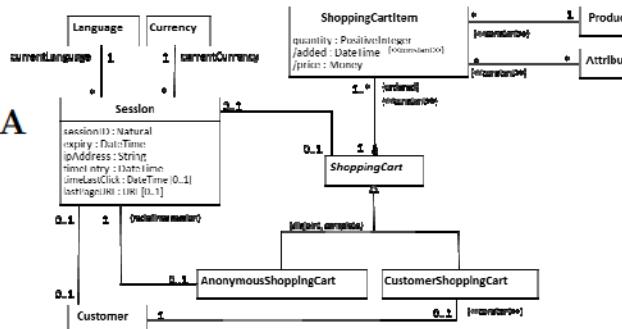
Two views

Working software over comprehensive documentation

The focus of this talk

Conceptual-Model Programming: A Manifesto

David W. Embley, Stephen W. Liddle, and Oscar Pastor



Functional requirements in agile development: User stories



In consultation with the customer or product owner, the team **divides up the work to be done into functional increments called “user stories”.**



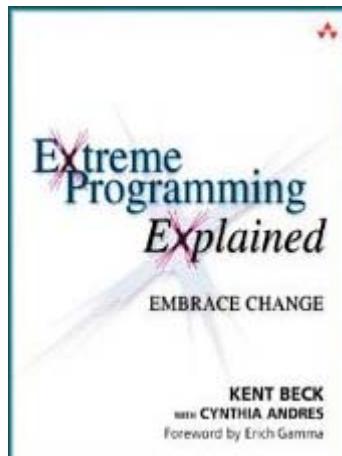
User stories

As a **role** I want **feature** so that **reason**

As a **bank account holder**
I want to be informed if my monthly balance
is projected to go to zero or below
so that I can arrange for an overdraft



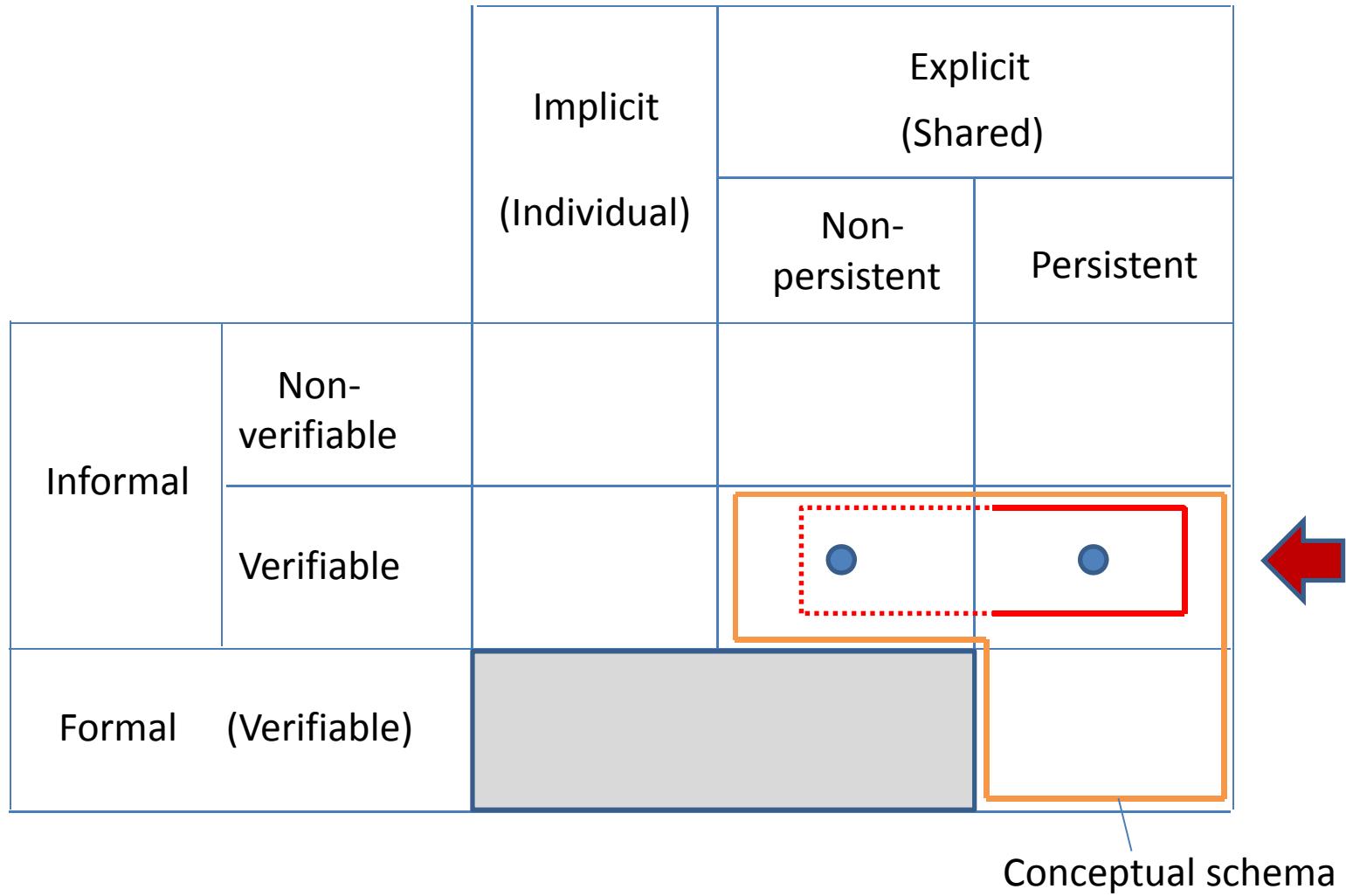
User stories



One thing the customer wants the system to do.
...
Should be **verifiable**.



User stories



The principle of necessity of conceptual schemas

in an agile way

To develop an information system
it is necessary
to define its conceptual schema

Alistair Cockburn's Cooperative game principle

Software development is a cooperative game...

The **primary** goal of the game is to deliver useful,
working software.

The **secondary** goal is to set up for the next game.

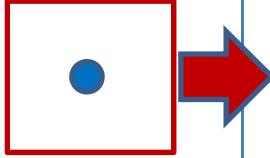
The next game may be:

- to alter or replace the system or
- to create a neighboring system.

Documentation

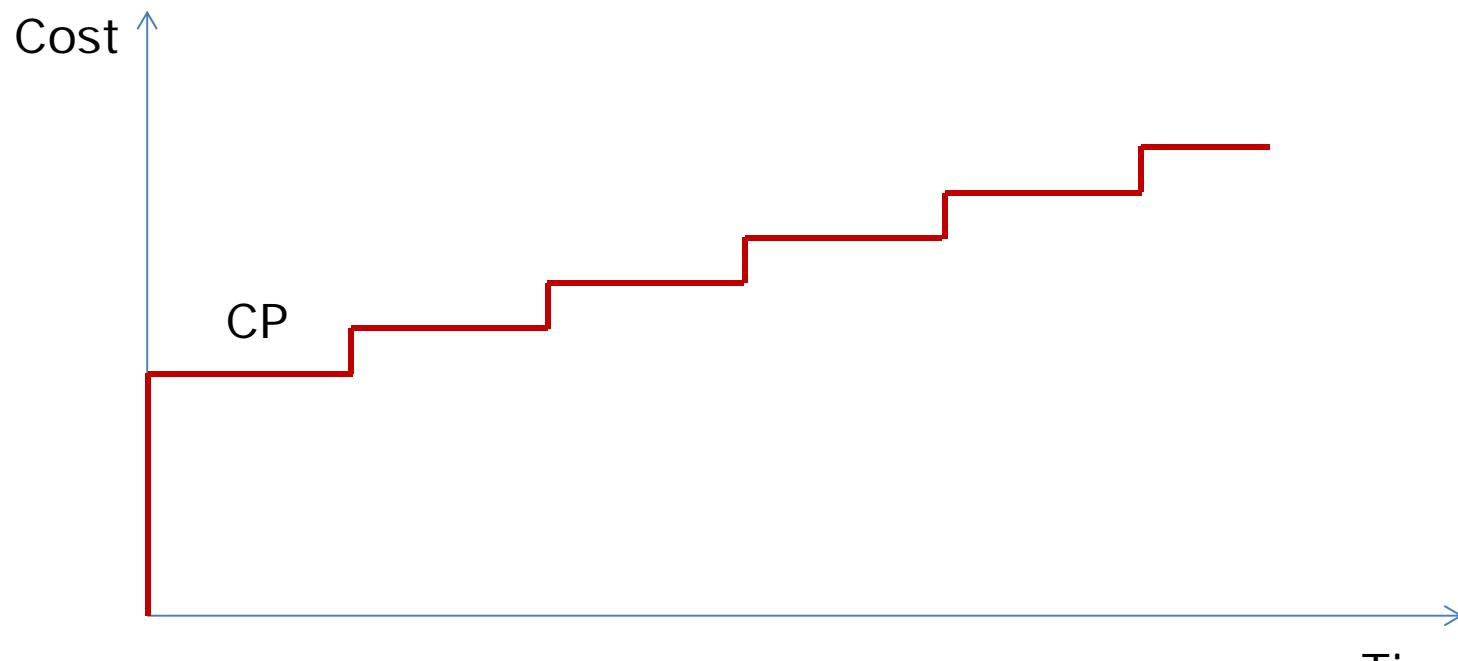


Conceptual schemas: Persistent or non-persistent?

		Implicit (Individual)	Explicit (Shared)	
			Non-persistent	Persistent
Informal	Non-verifiable			
	Verifiable			
Formal (Verifiable)				

Persistent or non-persistent? A life-cycle cost analysis

$CP = \text{Cost of initial documentation} +$
 $N * \text{Average cost of updating documentation}$
N : Number of system's functional updates



Persistent or non-persistent? A life-cycle cost analysis

$CP = \text{Cost of initial documentation} + N * \text{Average cost of updating documentation}$

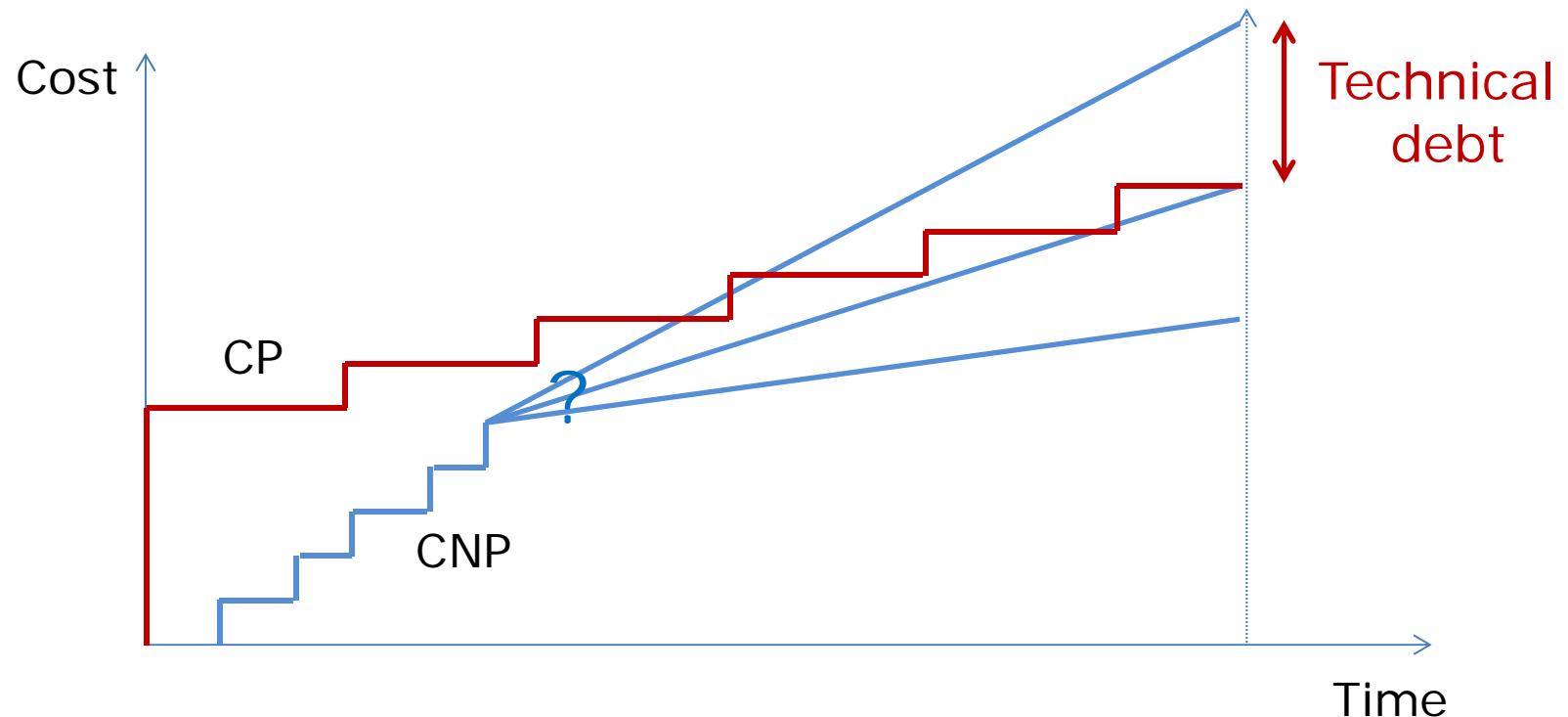
$CNP = M * \text{Average cost of recovering requirements}$

N : Number of system's functional updates

M : Number of times functional requirements are needed and not available



Persistent or non-persistent? A life-cycle cost analysis



Technical debt



WIKIPEDIA
The Free Encyclopedia

... A metaphor referring to the eventual consequences of poor ... software development. The debt can be thought of as work that needs to be done before a particular job can be considered complete. If the debt is not repaid, then it will keep on accumulating interest, making it hard to implement changes later on.

Common causes of technical debt :

Lack of documentation, where code is created without necessary supporting documentation. That work to create the supporting documentation represents a debt that must be paid.

Non-persistent may be better when...

$$CNP = \frac{M * \text{Average cost of recovering requirements}}{\text{ }}$$

- Short life cycle
- Few people
- Stable team
- Team remembers everything always

- Low cost

M : Number of times functional requirements are needed and not available

How to decrease the initial cost

May remain **non-persistent** requirements that:

- Can be easily observed in the system
- Can be easily recovered
- Are common-sense
- Are obvious (dependent)

Sometimes requirements are easily observed in the system



15. ICEIS 2013: Angers, France

- [□](#) [Download](#) [Cite](#) Slimane Hammoudi, Leszek A. Maciaszek, José Cordeiro, Jan L. G. Di **Enterprise Information Systems, Volume 1, Angers, France, 4-7**

Invited Speakers

Keynote Speakers

- [□](#) [Download](#) [Cite](#) Stephen J. Mellor: **Agile Model Driven Development.** IS-5
- [□](#) [Download](#) [Cite](#) Fabien Gandon: **Semantic and Social (Intra)Webs.** IS-7
- [□](#) [Download](#) [Cite](#) Ulrich Frank: **Multi-Perspective Enterprise Modelling as a Found**
- [□](#) [Download](#) [Cite](#) Henderik Alex Proper: **Architecture-based Services Innovation.** I

Databases and Information Systems Integration

Full Papers

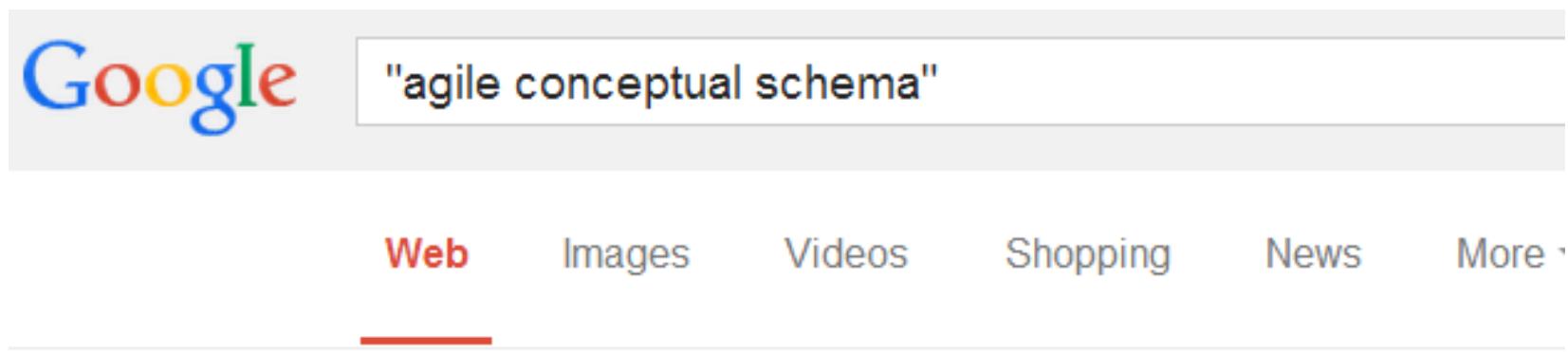
- [□](#) [Download](#) [Cite](#) Mai-Lun Chiu, Chu-Ying Fu, Ing-Long Wu: **An Analysis of Supply Ch** **Social Capital, Justice, and Technology Use.** 5-12
- [□](#) [Download](#) [Cite](#) Michael Henderson, Ramon Lawrence: **Are Multi-way Joins Actual**

But not always

The screenshot shows a Google search results page for the query "UML". The search bar at the top contains "UML". Below it, there are tabs for "Web", "Books", "Images", "Videos", "News", "More", and "Search tools". The "Web" tab is selected. The search results page displays approximately 277,000,000 results in 0.39 seconds. The first result is a link to the Wikipedia page for "Unified Modeling Language". The second result is a link to the official UML website. The third result is a link to the University of Massachusetts Lowell's website. The fourth result is a link to ArgoUML. The fifth result is a link to an IBM tutorial on UML. The right sidebar contains sections for "Unified Modeling Language" (with a link to its Wikipedia page), "Related topics" (including "use case", "ArgoUML", "University of Massachusetts Lowell", and "IBM Rational Unified Modeling Language"), and a "See results about" section for "University of Massachusetts Lowell".



Agile conceptual schemas?

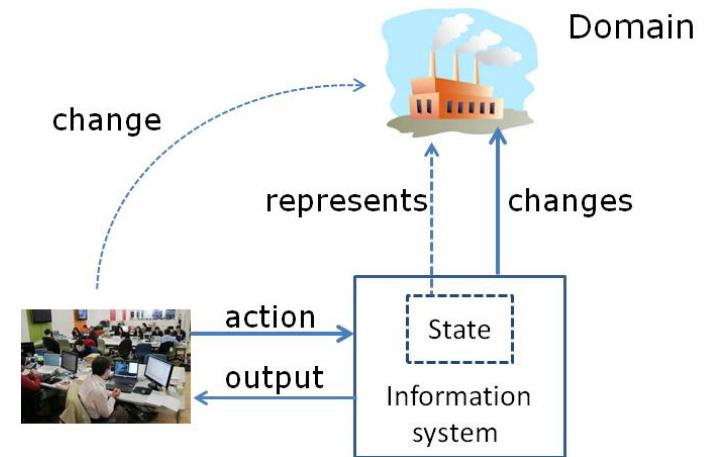


No results found for "agile conceptual schema".

Results for **agile conceptual schema** (without quotes):

Agile persistent conceptual schemas?

- ✓ • Domain concepts represented in the IS
- (*) ✓ • Definitions
- (**) ✓ • Integrity Constraints
- (***) ✓ • Events/Actions:
 - Triggering conditions
 - Constraints
 - Effect on the state
 - Output



(*): When may not be known/accessible

(**): When not externally visible

(***) When not externally visible and independent

Outline

- Back to basics: the need of **requirements**
- What are the **conceptual schemas**? Do we really need them?
- The main driving force of **formal** conceptual schemas
- Conceptual modeling in **agile** development. Is it needed?



Conclusions

Conclusions (I)

The principle of necessity applies also to agile methods:

To develop an information system
it is necessary
to define its conceptual schema



Conceptual modeling
is a necessary activity
in information systems development

Conclusions (II)

Key distinctions in conceptual schemas:

- Informal/formal
 - Formalization enables automatic processing
- Non-persistent/persistent
 - Has non-obvious cost implications
 - Agile conceptual schemas may be an appropriate persistence degree.

Thanks for your attention

