Expressing Business Semantics

Tony Morgan

Professor of Enterprise Informatics

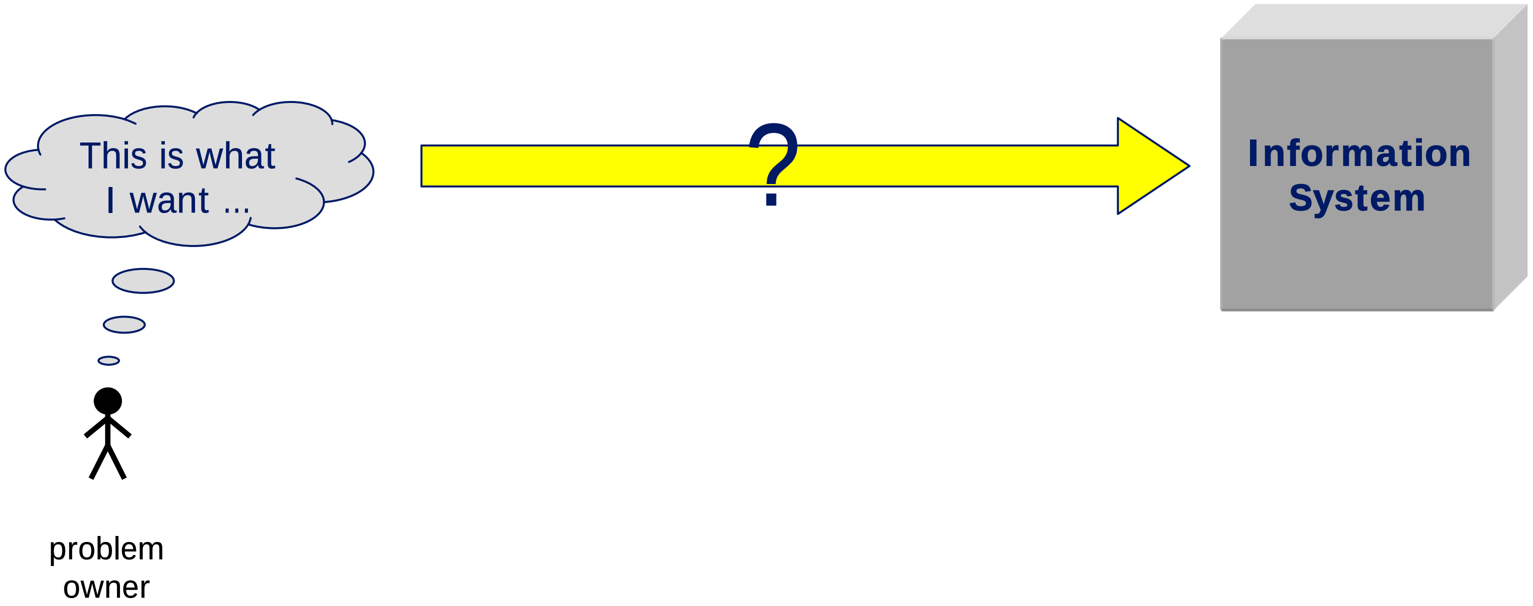
Northface University

What's the problem?

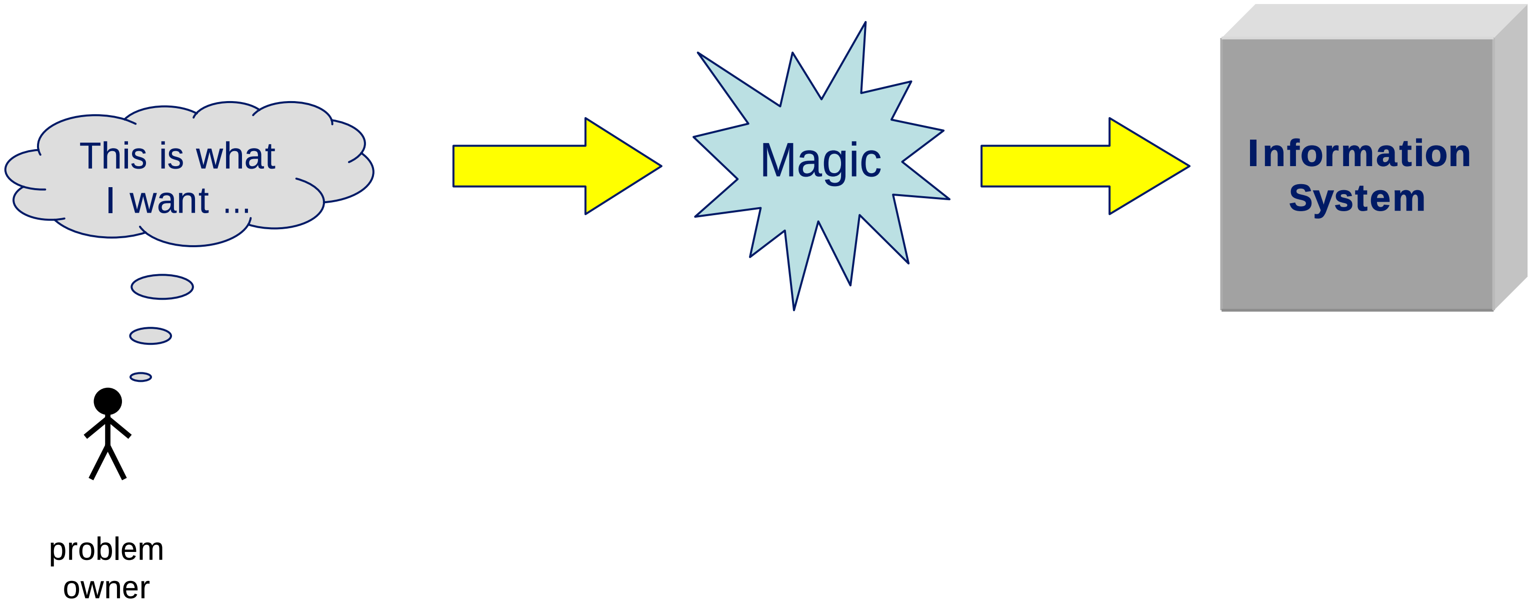
- At least $350 Billion is lost each year on IT, worldwide*
  - abandoned projects, rectification of errors, consequential losses, ...
  - not all of this could be saved by a better development process
    - operational errors, changing business conditions
- But potential savings are <u>at least</u> $200 Billion per year
- This requires a radical re-think of the way we build systems
- Producing an Information System today is:
  - costly (expensive resources over extended periods)
  - much too slow for modern business conditions
  - very risky (hard to control, high failure rate)
  - highly unsafe (introduces hidden failure points)
- Maybe there’s a better way of doing things ...

*Sources: Giga Group, Gartner Group, Standish Group, CCTA, Brunel University

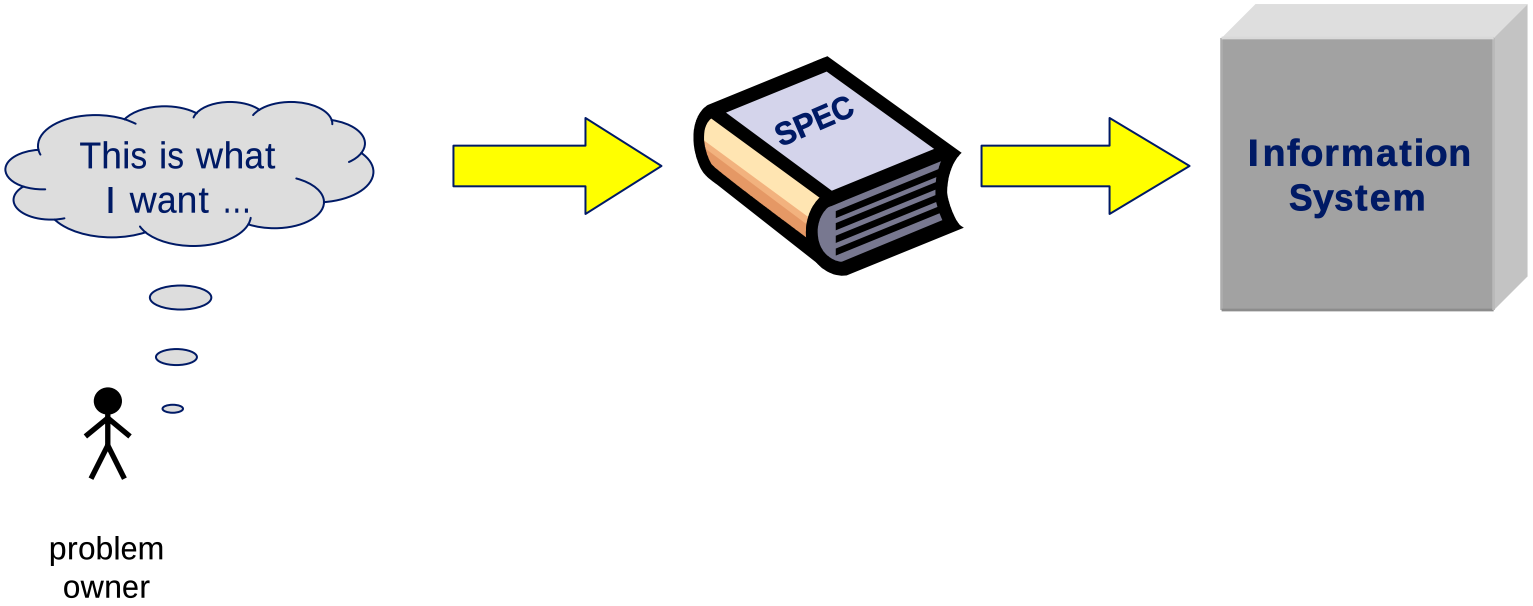
Software development (simplified!)



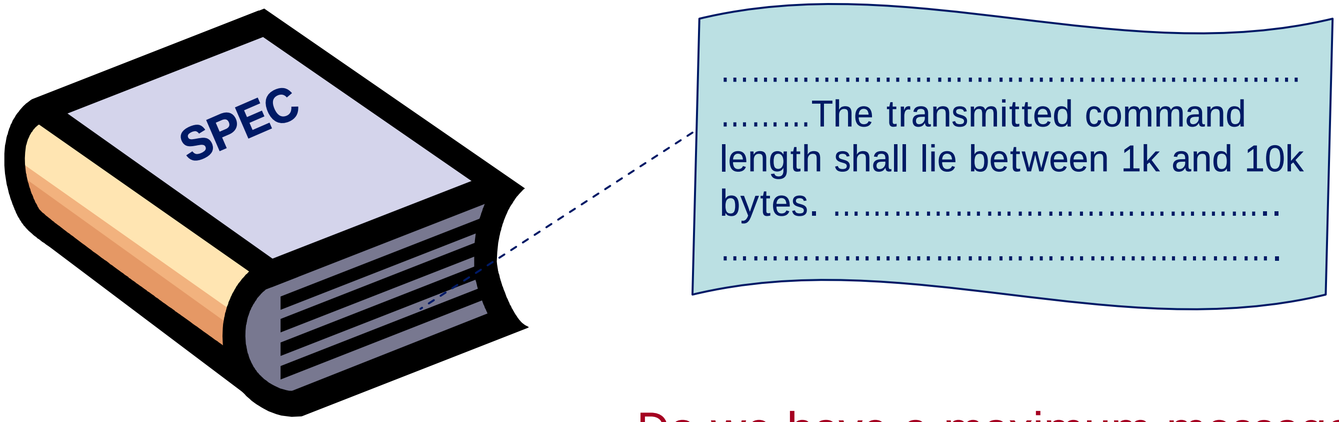
Software development (simplified!)



Software development (current)



The problem with specifications



……………………………………………………
………The transmitted command length shall lie between 1k and 10k bytes. …………………………………….
…………………………………………………..

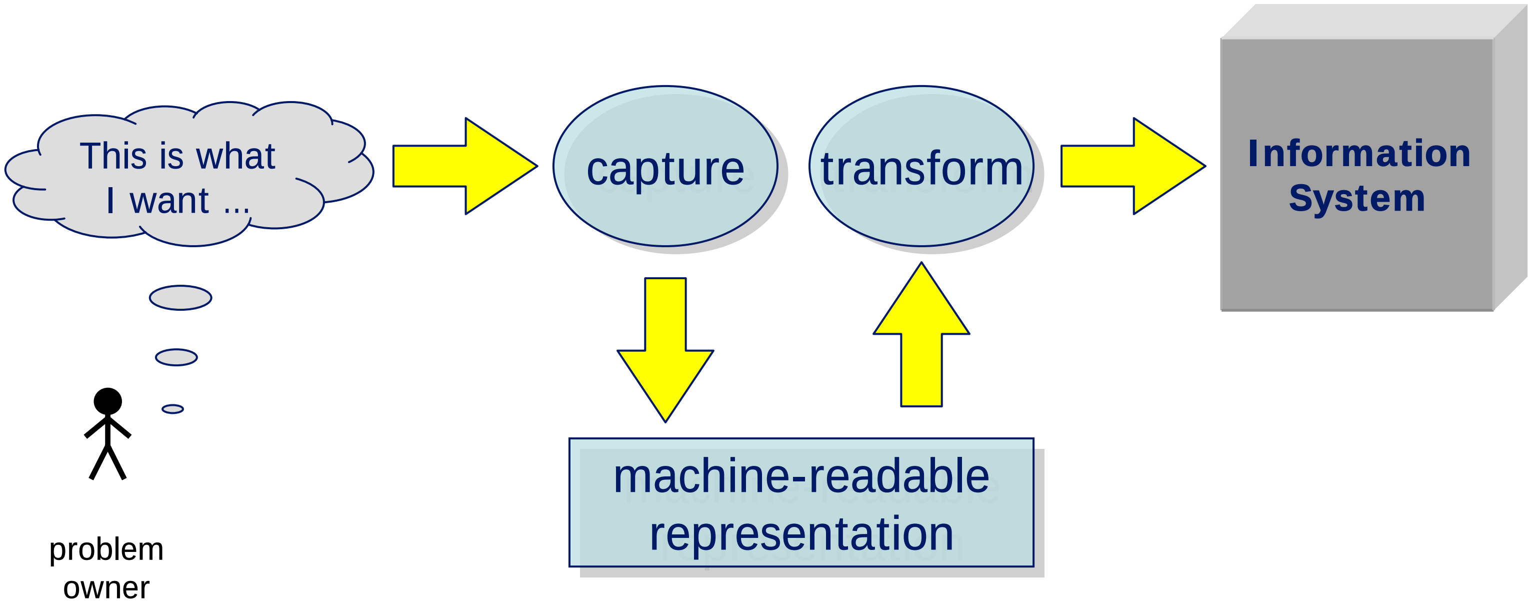
Do we have a maximum message size?

The semantics are in the mind of the reader and writer

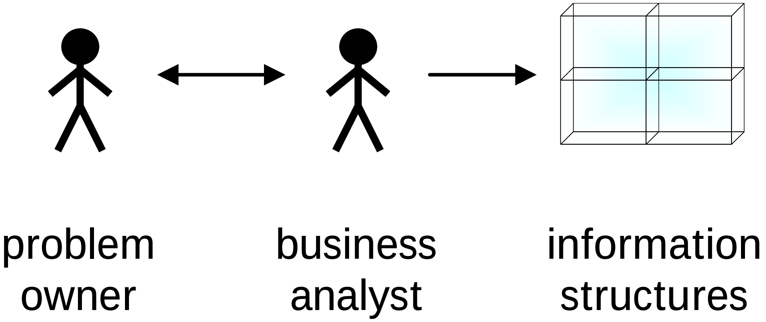
Unreliable semantics (dependent on human perspectives)

Search tools (etc.) are not geared to semantics

Software development (possible)



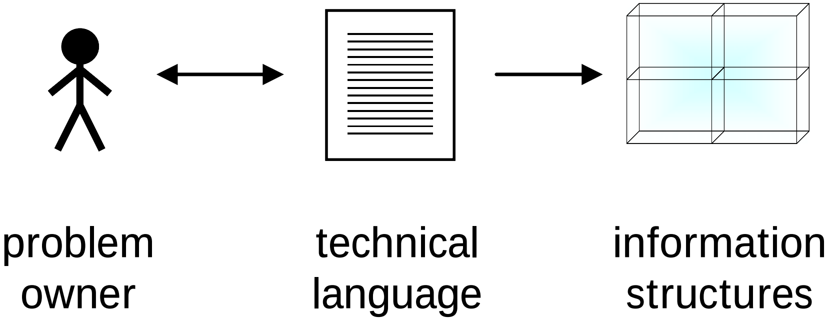
Unsuccessful approaches


problem owner

business analyst

information structures

Issue: problem owner remote from structure


problem owner

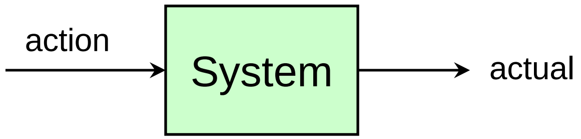
technical language

information structures

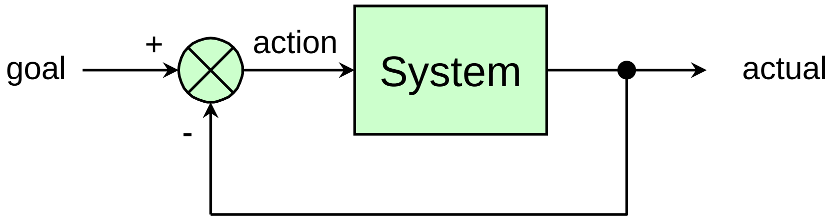
Issue: problem owner can't use technical language

Open loop, closed loop

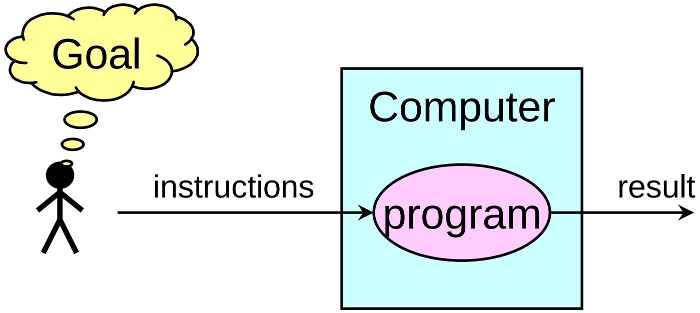
Open Loop Control



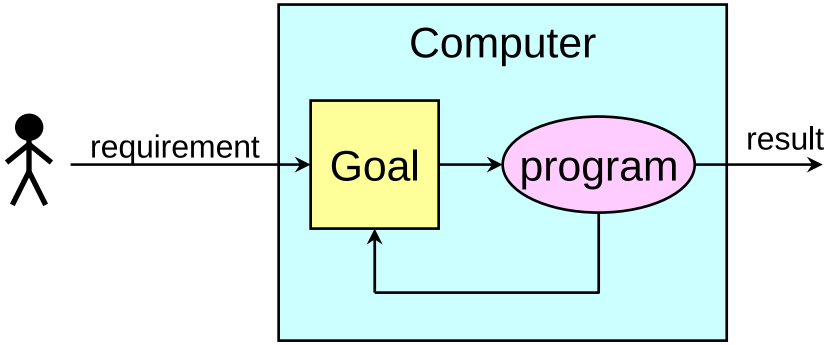
Closed Loop Control



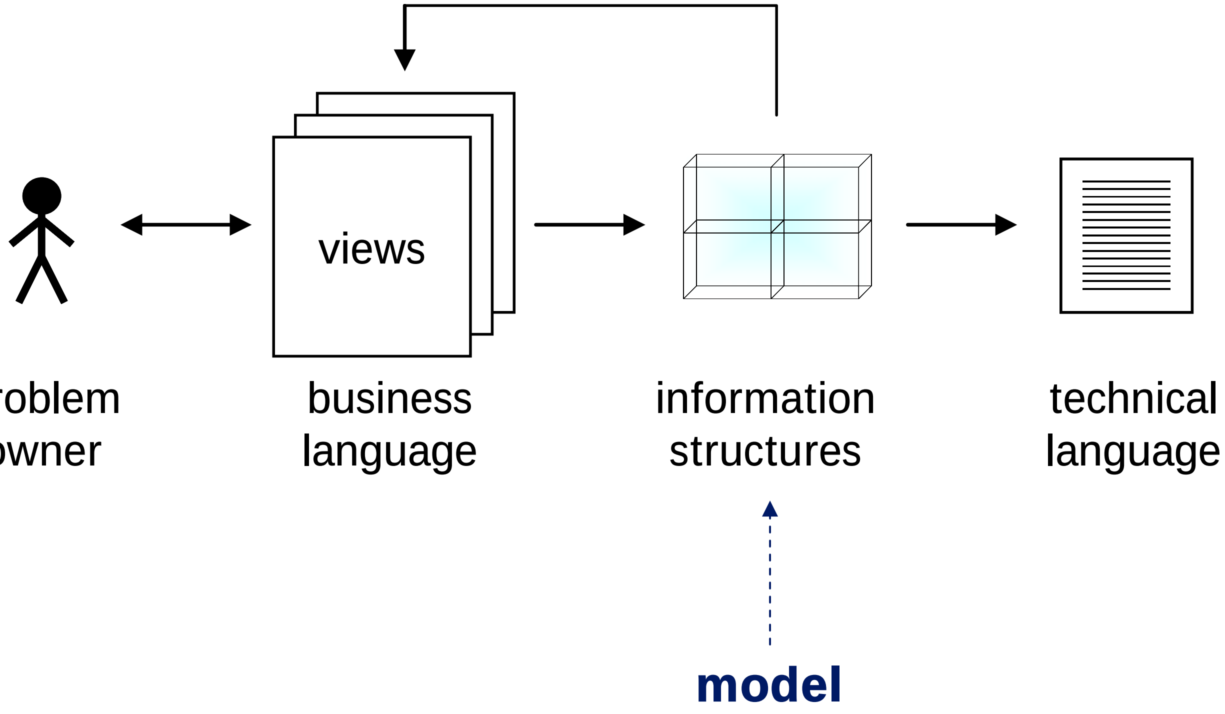
Open Loop Programming



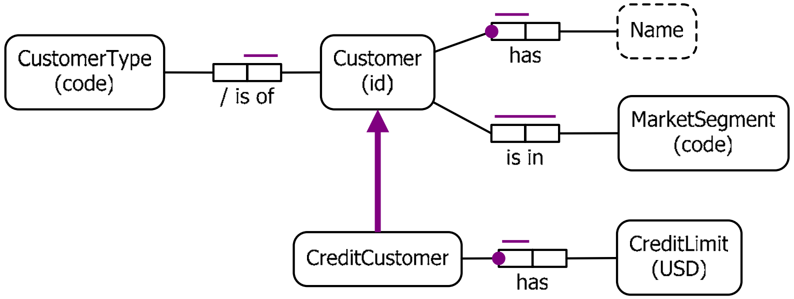
Closed Loop Programming



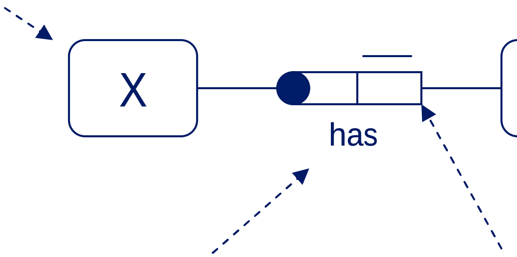
A more promising approach



A simple model



object type



fact type
("an X has at least one Y")

fact type role

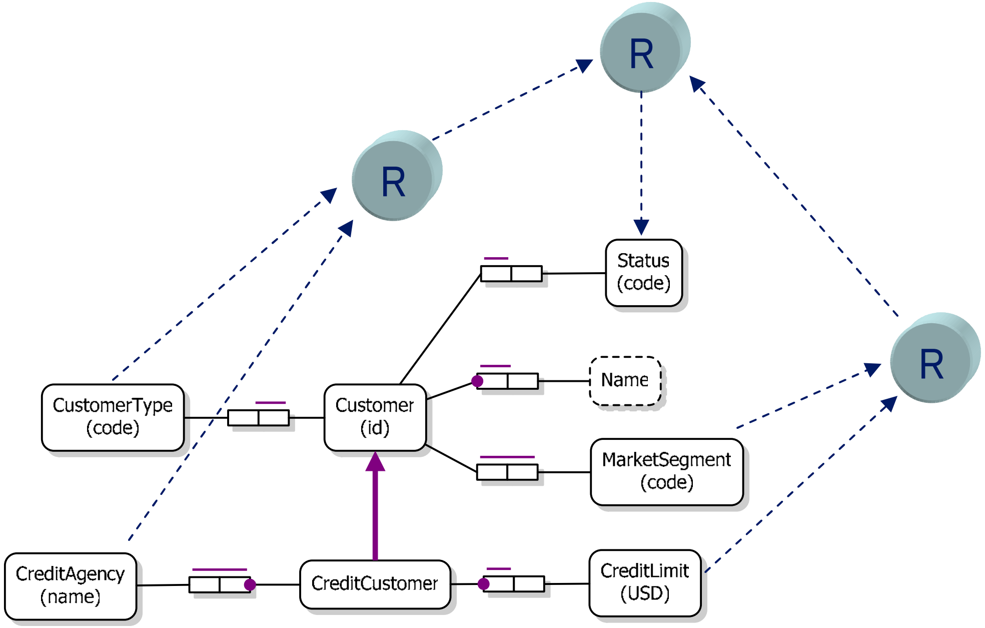
object instance
(of type X)

x1 has y1
x1 has y2
x2 has y3

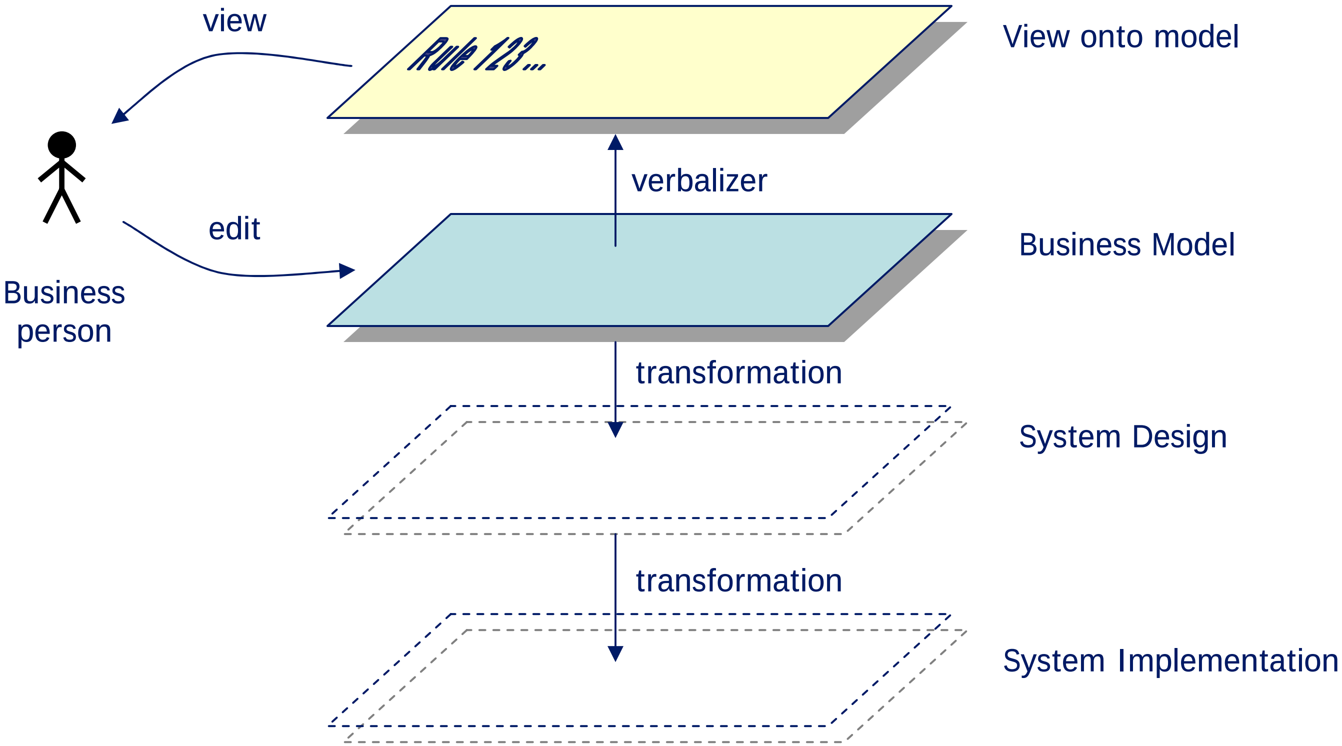
fact type instance
(i.e. a fact)

(ORM2 notation)

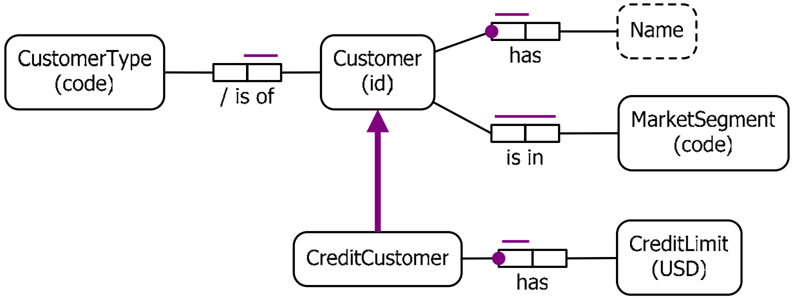
Rules and models



Defining rules with a model



Generated verbalizations



Customer is of CustomerType

Each Customer is of at most one CustomerType.

Customer is in MarketSegment

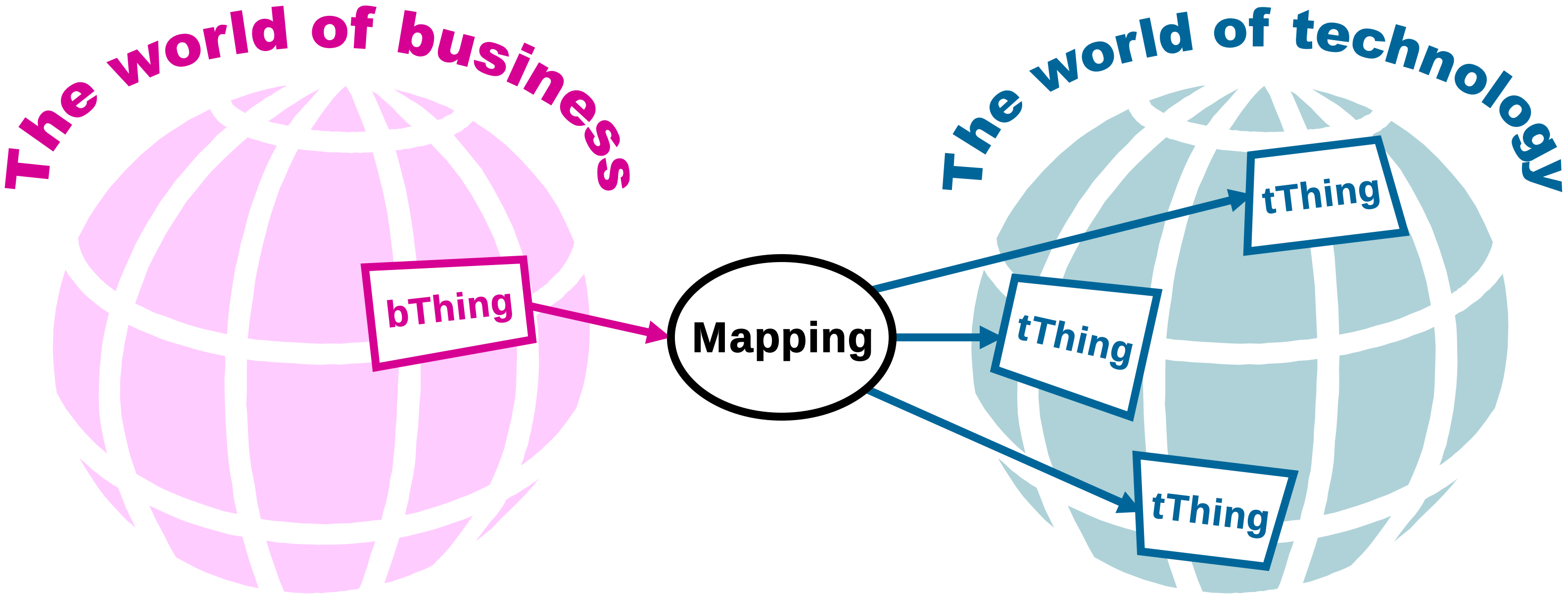
It is possible that some Customer is in more than one MarketSegment and that more than one Customer is in some MarketSegment.

CreditCustomer has CreditLimit

Each CreditCustomer has some CreditLimit.

Each CreditCustomer has at most one CreditLimit.

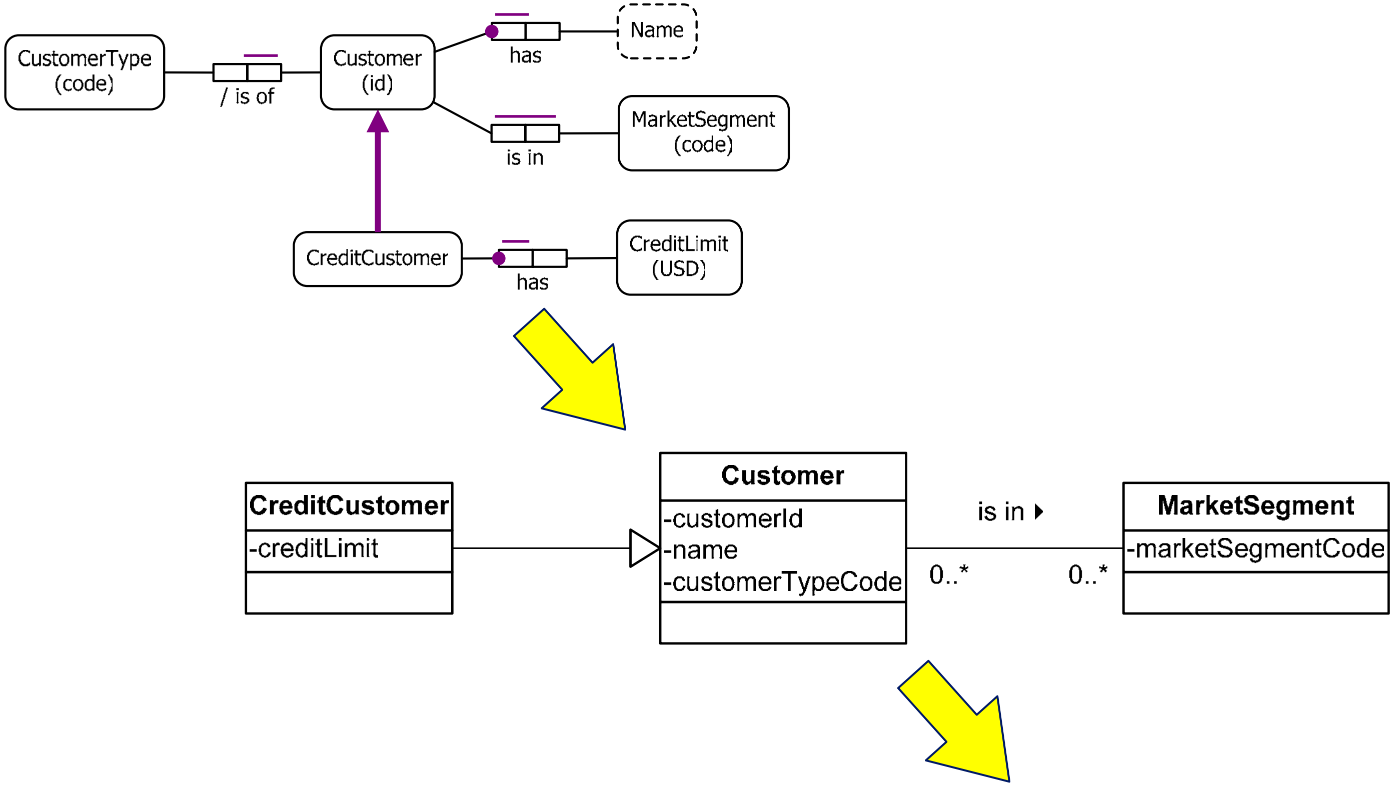
Mapping from business to technology



bThing:
Defines some element(s)
of the business

tThing:
Technology realization
of some element(s)
of the business

Automated transforms



```
public class Customer {
   private int customerId;
   private string name ;
   // and so on }
```

What's different this time?

- How is this different from previous approaches like "4GLs"
- No expectations that there will be "a language'
  - except in a very general sense
  - a multi-faceted approach is needed
- Business models are the starting point
  - understanding the business is crucial
  - business involvement is essential
- Technology framework is not determined by the approach
  - independent of OS, programming language, database, etc.
  - choose what you like, then map to it
- Code is just a by-product
  - it's generated from models by a series of transforms

Some software implications

- Choice of programming language is unimportant
  - relegated to technical architecture / model compiler
- Very few programmers needed (if any)
  - mainly needed for model compilers and architecture definition
- Implementation can be very fast
  - a new system in minutes instead of months
- Software quality improved dramatically
  - automated process introduces no bugs
- Software (code) has almost no value
  - it costs almost nothing to produce
  - software re-use is pointless -- throw it away and generate afresh
- The business owner is in full control
  - systems include exactly what's been specified
  - but full control also implies full responsibility!

Some Semantic Web implications

- Business models assume one consistent interpretation
  - possible in business, but who controls the Web?
- Must use "natural' reference schemes
  - e.g. can't rely on a unique 'id', as in database systems
- What happens if assumptions are broken?
  - e.g. we assumed a one:many relationship, but then find out later that it is really many:many
- How are rules controlled in a Web environment?
  - authorization? version control? retirement?
- How can we identify dependencies?
  - my service depends on your service
  - you change your rules
  - should I care? and how would I know about your changes?

Standardization initiatives

- Several initiatives are under way to define standards for business rules
- In particular "the Semantics of Business Rules and Vocabulary (SBVR)" – under the auspices of OMG
  - other related standards include "Production Rule Representation (PRR)"
- Some SBVR features:
  - captures business semantics
  - business-centric rule expression
  - underpinned by formal logics
  - mapping to XML Schema for transfer

Summary

- The way we develop software will change dramatically over the next few years
- Model-Driven Development will become the norm
- Standardization initiatives will rapidly increase acceptance
- Solutions that work within the bounds of a single organization may not work in a Web environment
    - interesting areas for research
- The main obstacle to change is mind-set, not technology!

NORTHFACE UNIVERSITY

You're serious about software. So are we.

www.northface.edu