Open Models Initiative 3rd International Workshop
Modeling Methods in Motion Vienna Sept 13, 2012

Some poor remarks from a humble modeler

An Integrated Theory of Informatics

from Technology to Science

W. Reisig
Humboldt-Universität zu Berlin

Humboldt Universität Informatik

T o P
Institut für Informatik
Humboldt-Universität zu Berlin

Theory of Programming

Let’s start fundamentally …

What constitutes a science?

The example: Physics
A system of notions
and relations among them,
stating laws of nature,
described in terms of *models*.

Models in Science

Typical example:

The term “energy”
+ all laws about energy.

There is nothing like *energy* in nature.
The notion of “energy” is an *abstraction*
from many aspects in nature

Example: What remains invariant?

sum of energy
first residing in gasoline,
then in acceleration,
then in deformed metal sheet.

What physicists *really* did:
Searched a notion, general enough
to describe what remains invariant
… and called it *energy.*

Scientific theories

Physicist do accept intuitively hard models ("*theories*")
if they offer convincing explanations,
in particular *invariants*.

Invariant in Chemistry

$CH_4 + 2\ O_2 \rightarrow CO_2 + 2\ H_2O$

Search for good models
= Search for comprehensive invariants. $e = mc^2$

We should learn from this!

Even *Theoretical Biology* is behind (biological) invariants!

Our task

We must develop
integrated conceptual theories and models
in the style of the exact sciences
for … ??

for strictly organized flow
of information and items,

supporting nontrivial analysis and verification
by help of invariants.

Models and invariants in Informatics

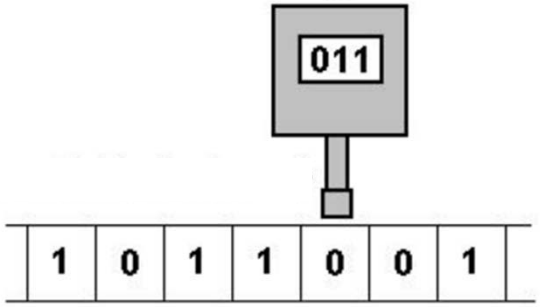
Informatics has models.

Some models have invariants.

Symbol Processing Models

“The computable functions”

Turing machines



very expressive, no invariants.

Logical Models

“symbols and what they denote”

PROLOG

relational data model

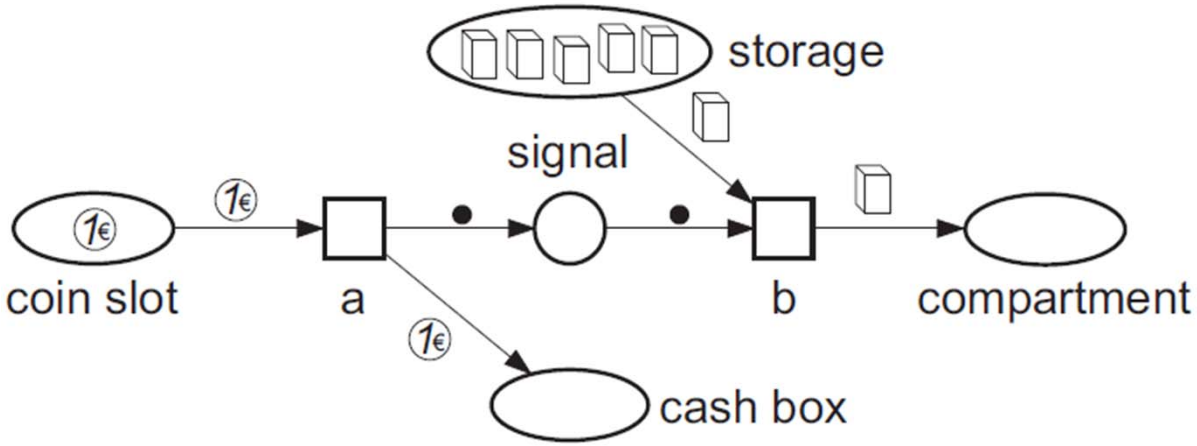
analysis: first order logic
still very expressive,
hence with limited analysis techniques

Programs as models

$$\textbf{while } (x < 10)\ x := x + 1$$

loop invariant:
Hoare Logic

Behavioral Models

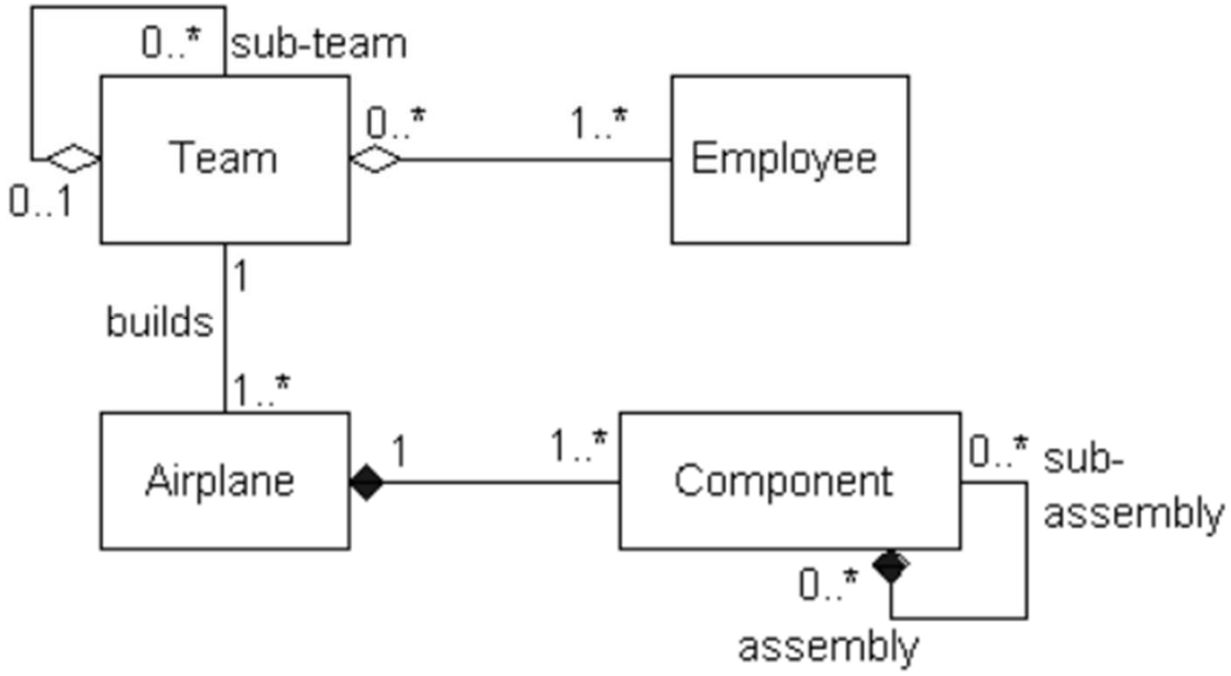


invariant:

|cash box | + |storage| = |signal| + 5

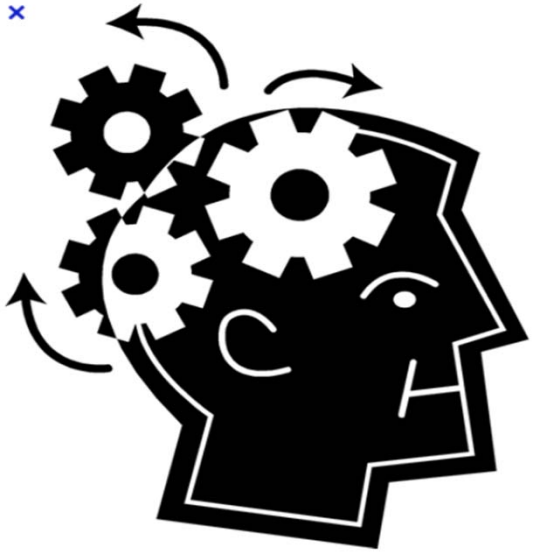
Software models

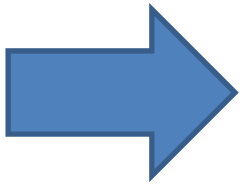
UML



not formal,
hence no invariants.

… the blunt Reality





The SW industry widely does without modeling.

Why?

The software industry doesn't benefit substantially.

… because models are complicated?

No! because a software developer don't get much out of a model.

Needed: *Comprehensive* models …

- for entire *systems*,
  not (only) for software components;
- allowing free choice of level of abstraction;
- shaping "the implementable"
  (not "the computable");
- including a comprehensive notion
  of "algorithm";

… *and nontrivial analysis techniques!*

… with comprehensive invariants

What remains invariant

when using

a cash machine ?

account
+ in hand

What remains invariant during running a garbage collector?
… while fulfilling a contract?

… and what, precisely is gained ?

How achieve all this?

Combine existing ideas;

generalize them substantially.

Start with basic notions,
extend domain specific aspects later

Search for basic notions

for strictly organized discrete flow
of information and items:

What notions may be subject
to rigorous formalization?

- static notions
- dynamic notions

Data / messages / documents / contracts / information / items

ownership, access rights ~

to dispatch, store , disseminate, communicate, compose ~

computer-mediated

When copied / composed:

What aspects remain?

What aspects change?

What are the properties of a copy / a compositum?

Activities / tasks

what means

to execute~

to cancel ~

to authorize ~

to delegate ~

to synchronize ~

to re-organize ~

Prospective theorems

Theorem 1: In each computerized system holds:

While computing

– without communicating –

the amount of *information* (?)

remains constant.

Theorem 2: To decide an alternative =

to consume a piece of information

Domain specific models

… not stand-alone,

but on top of basic notions

Always remember the aim

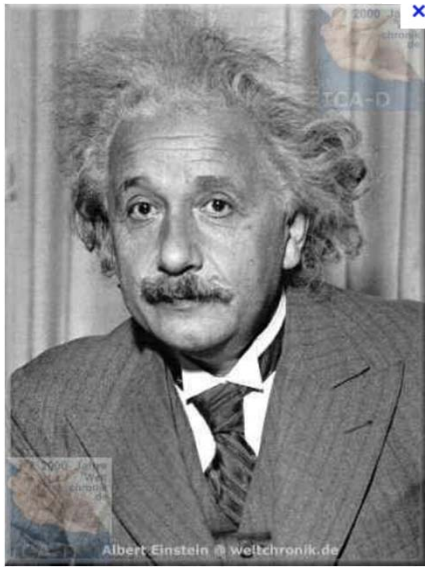
formulated by Grady Booch, the founnder of the UML:

Elevate models as to a first class citizenship ...
a peer of traditional text languages
(and potentially a master)

support this idea with open models, as suggested by the OMI

State of the art

Certainly, we are pre-Einstein



Probably, we are pre-Newton.

Open Models Initiative 3rd International Workshop
Modeling Methods in Motion Vienna Sept 13 , 2012

A Concerted Theory of Informatics

from Technology to Science

let's do it!

Theory of Programming

W. Reisig
Humboldt-Universität zu Berlin