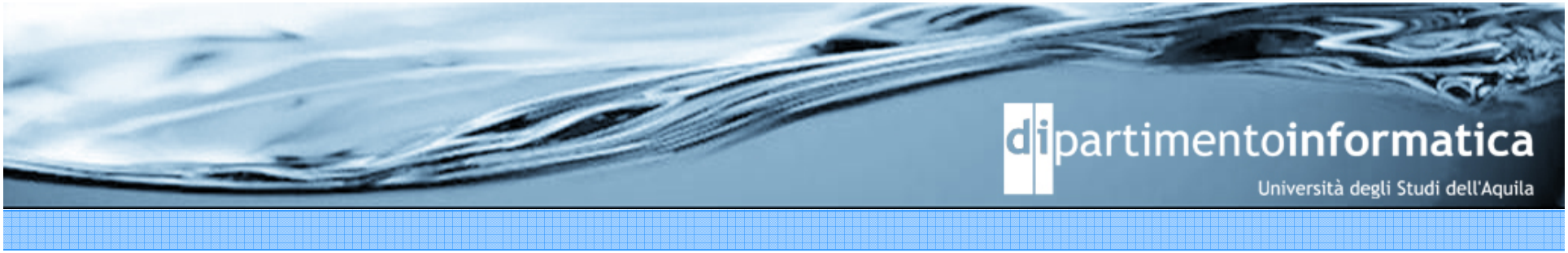


MODELS for ADAPTABILITY

Paola Inverardi

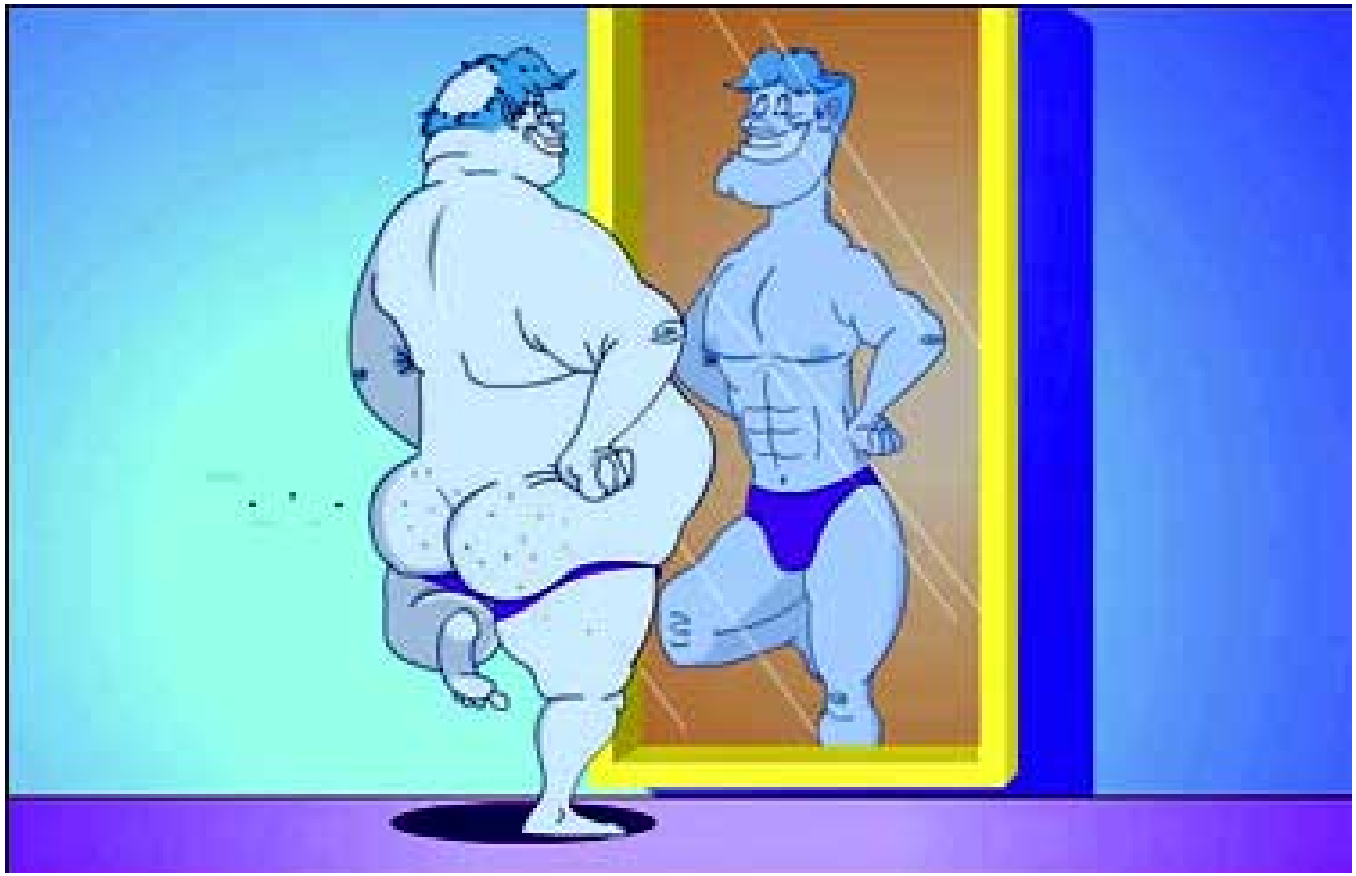
Software Engineering and Architecture Group
Dipartimento di Informatica
Università degli Studi dell'Aquila
I-67100 L'Aquila, Italy

What are Models?

- An *idealized* view of the system suitable for reasoning, developing, validating a *real* system
- Better if *formal*, e.g. rigorous, mathematical-logical flavour, etc.

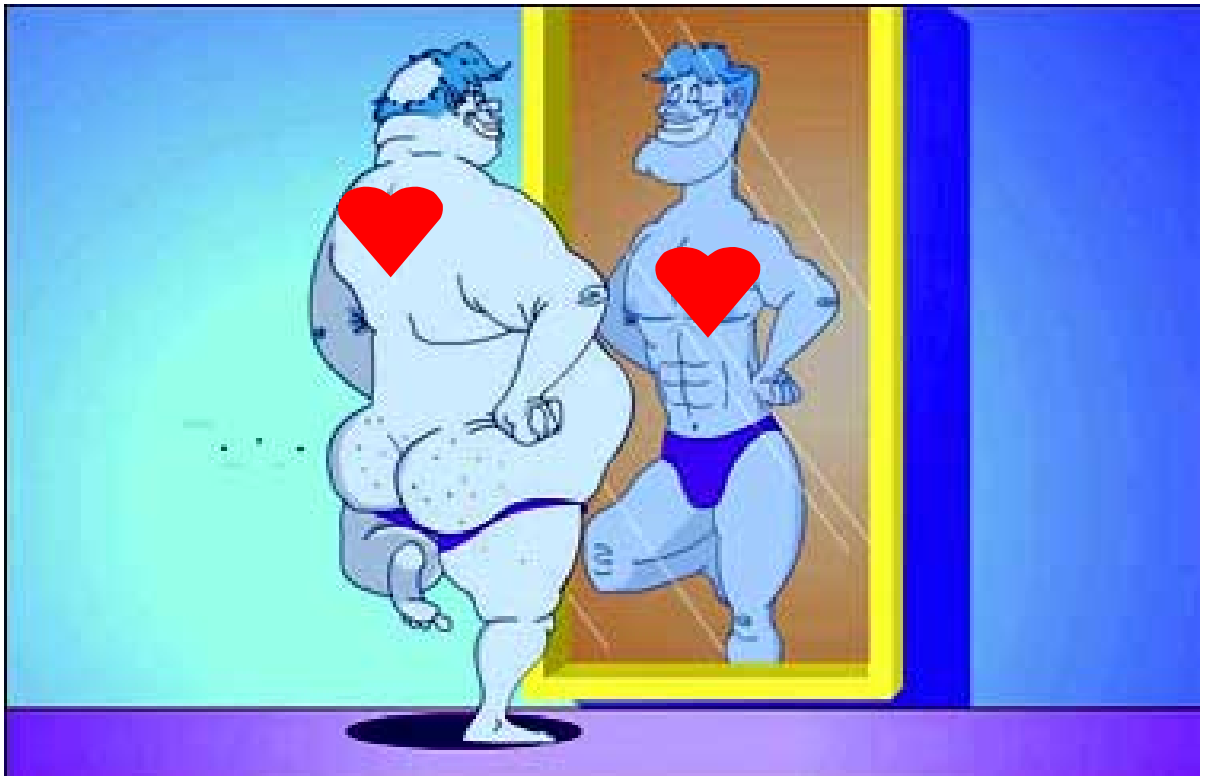
WHAT is ADAPTABILITY?

» The ability to *change* a system according to context variations, e.g. driven by QoS requirements



ADAPTABILITY II

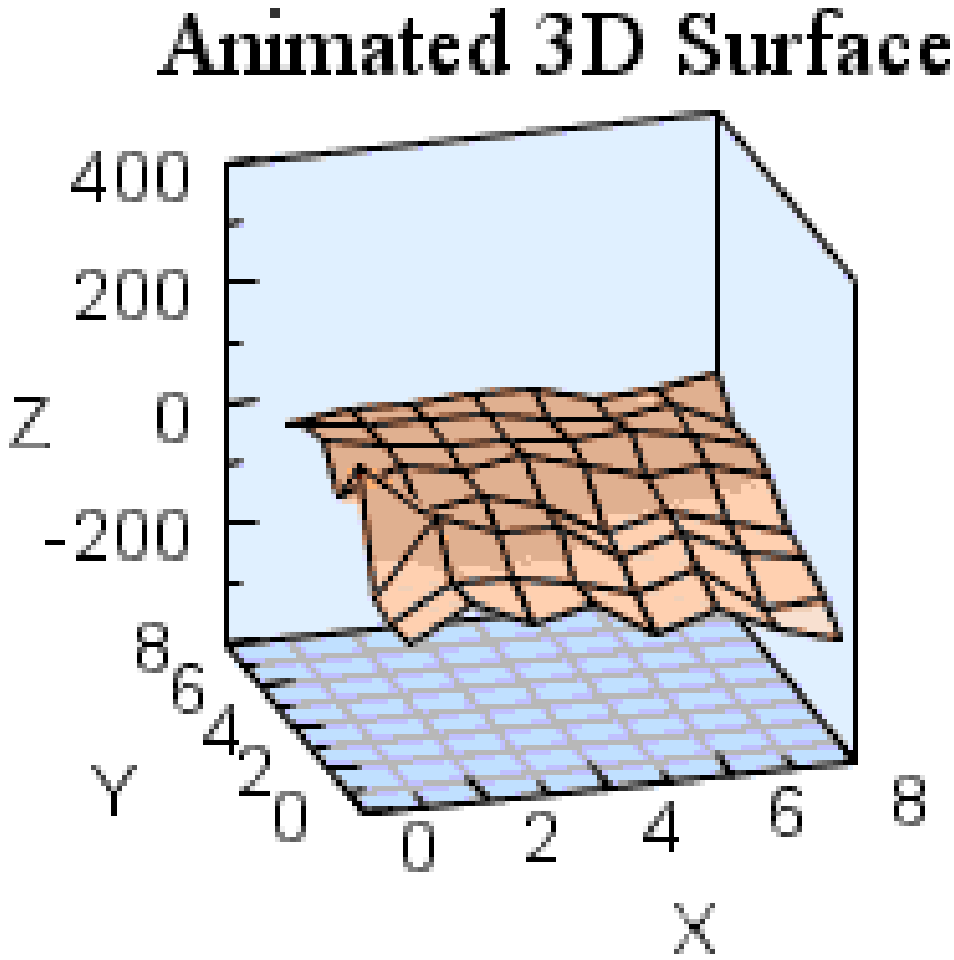
- But …Still remaining the *same*
- Adaptability makes sense only if it preserves something …the *Invariant*



A more serious example

» What is the *invariant* here?

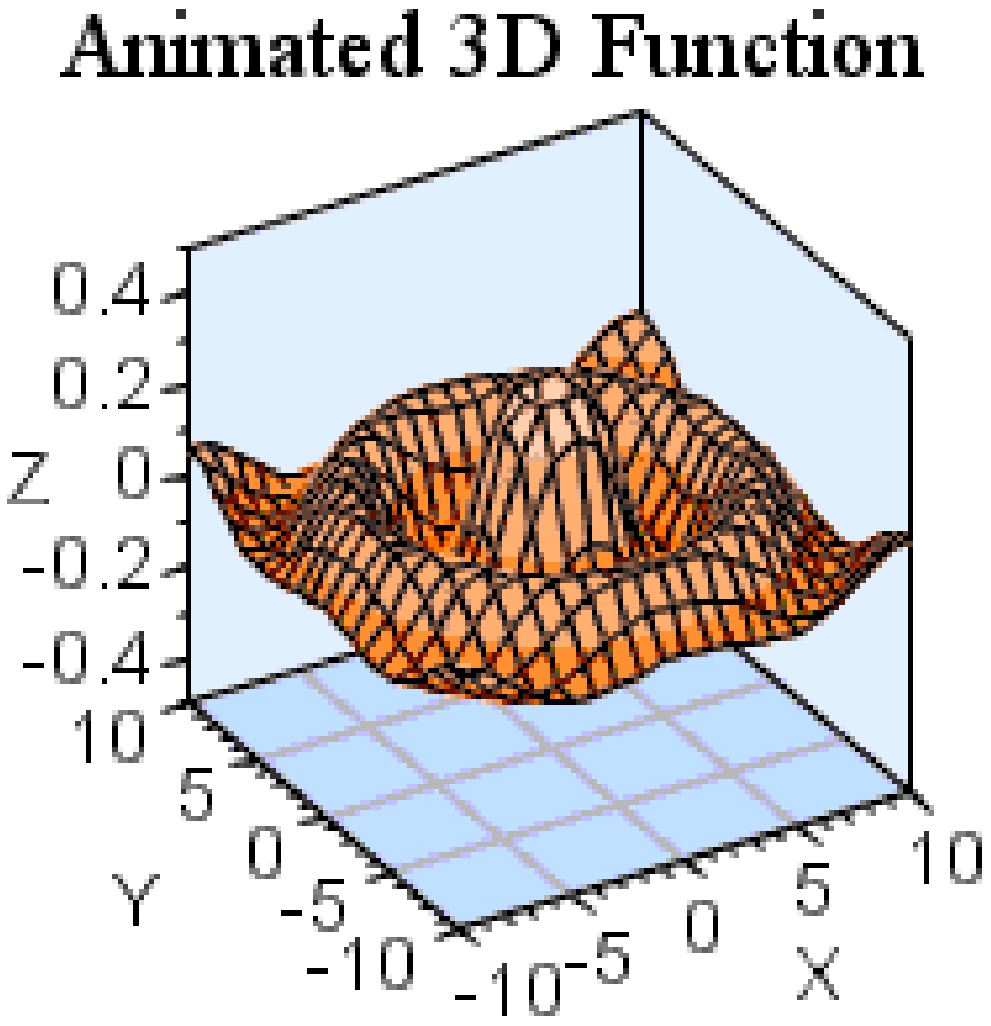
The ***surface***



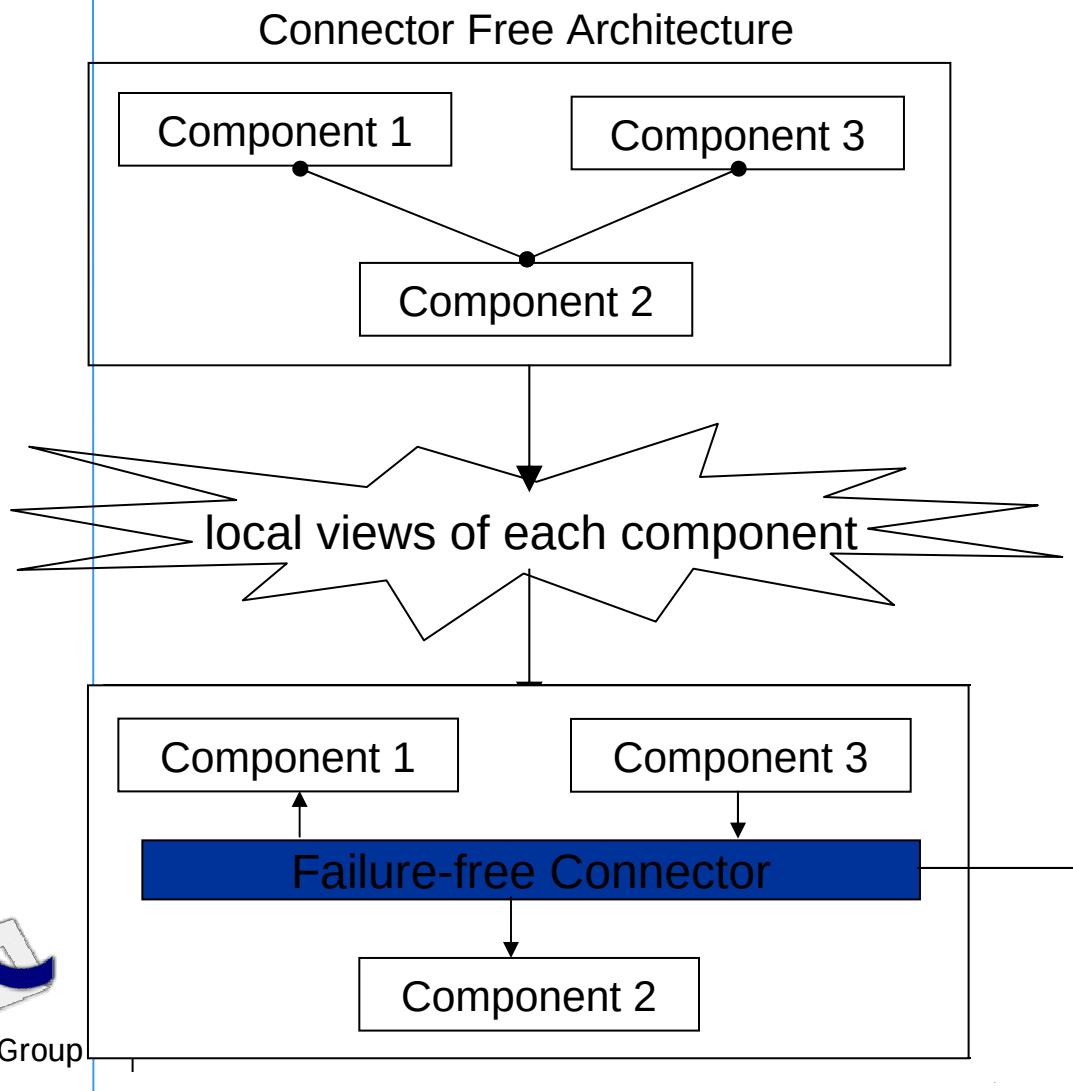
Even better/worse

» *Invariant* ???

The 3D ***function***



A more familiar example … (Tivoli-Inverardi)



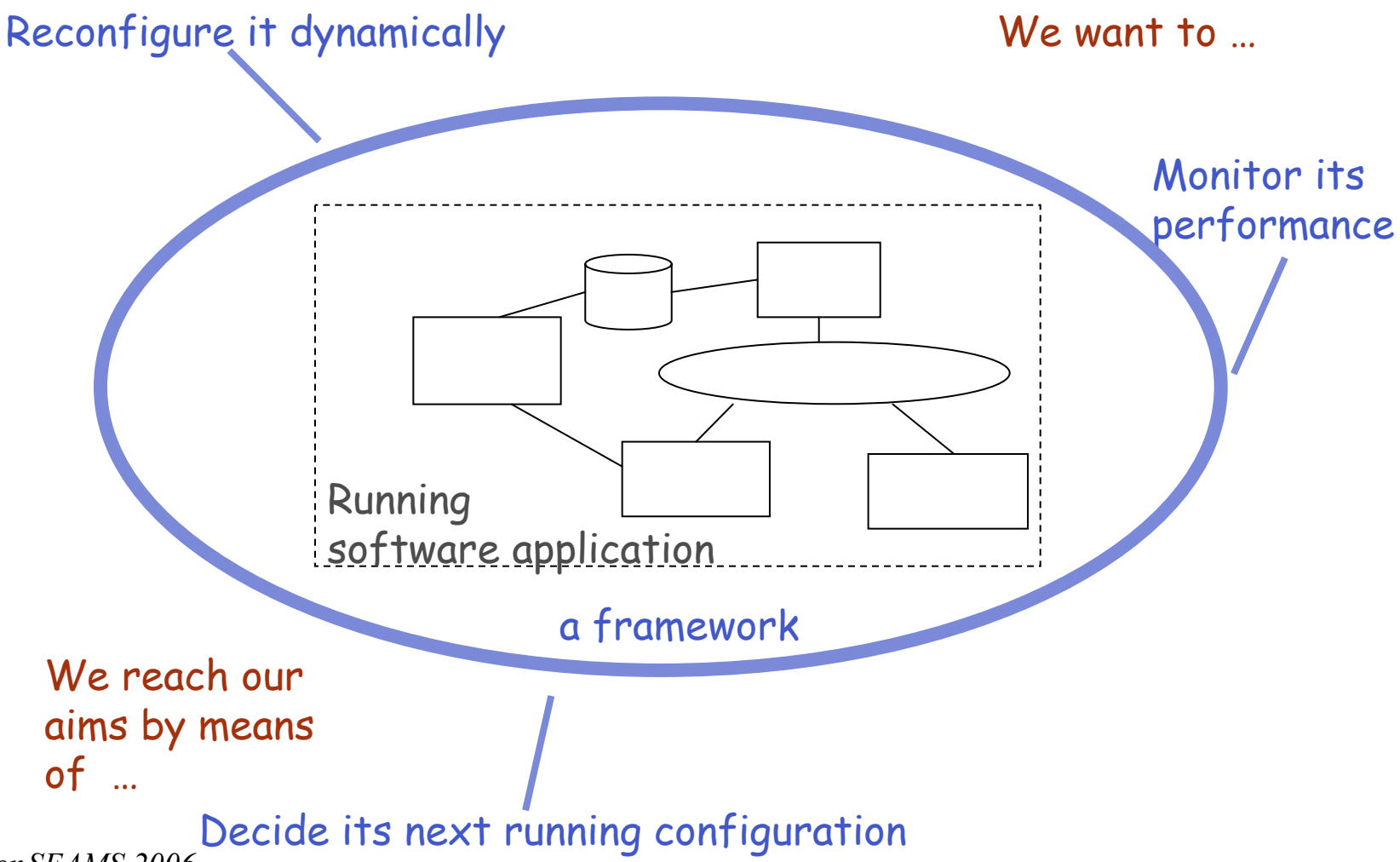
Structure changes –
equivalent behavior

What are the models and fomalisms

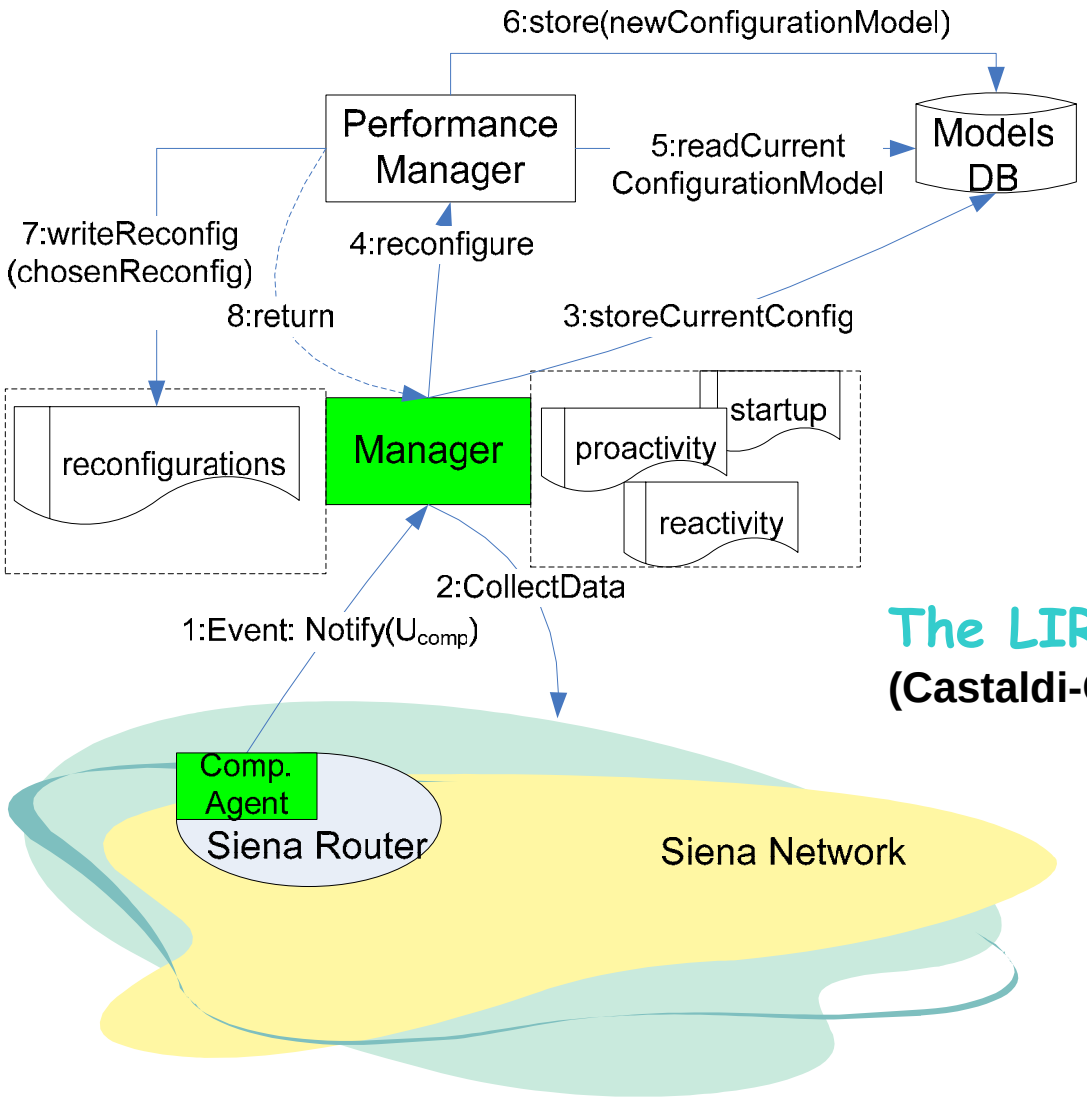
- An architectural model i.e. constraints on the way components can interact
- Behavioural model for components -- LTS
- Behavioral equivalence on LTS
- Temporal logic – Buchi Automata
- Model Checking

Ex. 2 - PERFORMANCE : system reconfiguration

Caporuscio-Di Marco-Inverardi



PERFORMANCE : system reconfiguration



The LIRA framework
(Castaldi-Carzaniga-Inverardi-Wolf)

Which models?

- System dynamic model (LTS etc)
- Queueing Network models (+-extended) derived from the dynamic models
- Models analysis
- Performance indices evaluation

First conclusion – 1 --

» Models for adaptability must be able to express the ***invariant*** essence of the system not the ***variable*** one …

» Easier for structure :

- topological constraints (Jeff&Jeff)
- Graph grammars (Le Metayer, etc.)
- Category Theory (Fiadeiro-Maibaum etc.)

What about behavior?

Invariant -- continue

Behavioral/Semantics invariance

Difficult in general: non-computable → restrictions

Examples:

- Type systems (can be also structure … ArchJava)
- Behavioral equivalence checks (Allen&Garlan , process algebras) better preorders
- Models checking and evaluation
- Constraints programming
- Code/component certification – Proof Carrying Code

An orthogonal issue: Static VS Dynamic

» Is adaptability static or dynamic?

The system adapts at run time *how* and *when* the adaptation is computed does not change the problem it is just a matter of ***cost***. Cost of the adaptation that maintains the invariant.

At the end it is just a pointer in the control link stack …

The real issue is what is the invariant and how do we maintain it?

Ex. Functional Languages and Higher order functions

(a' la ML) (ponlimorphic types, type inference)

Following the 3D function example

Build the n-dimension space → fix in the context the variable points that matter

Design the overall system with all its possible fluctations.

extract/ elicit the function, i.e. the ***non variable*** essence of the system

Example:

If we consider a service and a QoS space that can dynamically vary then the function is the "optimal " correlation among the points in the space to achieve the "best" overall QoS

An initial attempt to rephrase all this

- S = software system
- *SS* = Static description of S { the code, the structure of the code, the language it has been written, the developing artifact, the language in which it is described and all the models that can from these information be deduced (control flow graphs, slicing models, etc.)}
- *D* = {behavior of S} dynamic description of S
- *C* = {description of the running context} c ∈ C (might not be under the system control, otherwise just an input)
- <i> = input (known variability points)
- ***R*** suitable equivalence relation on *D*. ***R*** ⊆ *D*x*D*

Formalization 2 --- Re-configuration

Re-Configuration

Let Γ be the set of configurations $\gamma = <SS, c, <i>>$

Def. Let $\rightarrow_{reconf} \subseteq \Gamma x \Gamma$ such that $\gamma \rightarrow_{reconf} \gamma'$ iff $SS_\gamma \neq SS'_{\gamma'}$

re-configuration requires a **change** in the structure of S, formalizes the context and monitors the execution environment.

Formalization 2 --- Adaptability

Adaptation

S can be adapted to S' wrt an equivalence ***R*** *if*

$\langle SS, c, \langle i \rangle\rangle \rightarrow_{reconf} \langle SS', c, \langle i \rangle\rangle$ $(SS \neq SS')$ *and* $D \; \mathbf{R} \; D'$.

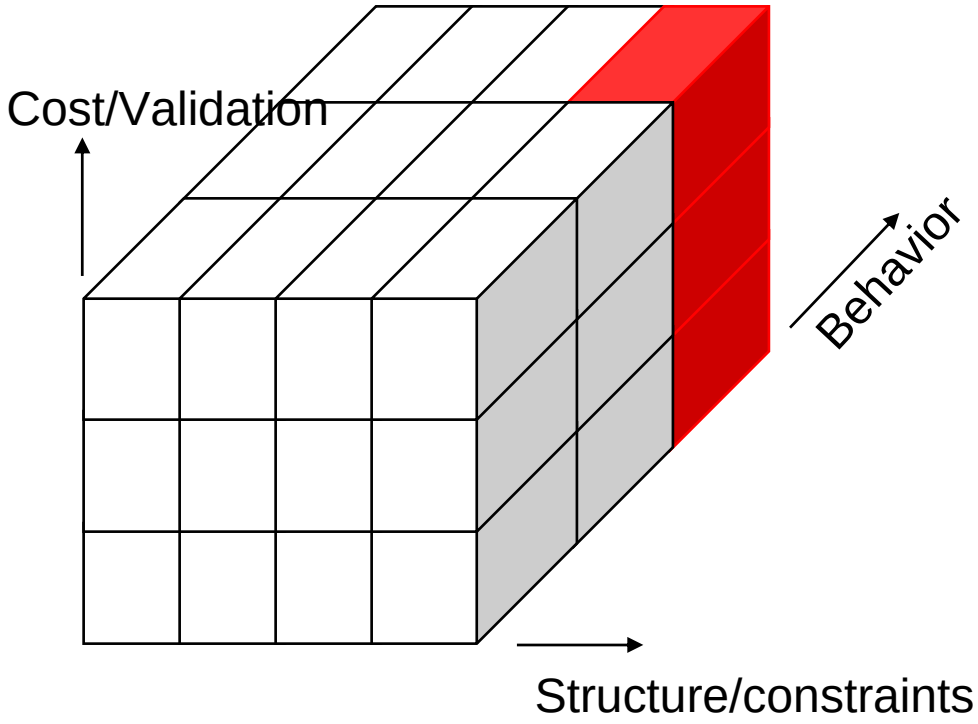
R can assess functional or non functional properties

More appropriately ***R*** should be a **congruence** relation that preserves contexts of use of the system thus modeling the user's observable view

In other words we require that adaptation implies a change in the static structure of the system. (e.g. we do not consider weak adaptation as adaptation)

Conclusions

Many dimensions to consider



My opinion: Focus on *Invariants*

- **Structure**

  Software architectural models/Styles, patterns, etc.

- **Behavior**

  Abstract behavior, Types/signatures, Behavioral equivalences

- **Cost/Validation**

  Interplay between static and dynamic analysis, clients and servers, compiled and interpreted

Do not stop looking for models … may be that …

Under the formal suite also models …

…. have an heart …

References

- Woss proceedings