Software architecture for reactive systems (introduction)

Luís Soares Barbosa José Proença

HASLab - INESC TEC
Universidade do Minho
Braga, Portugal

February 2018

For today

Overview of Software Architecture

Its view by MFES profile

Pragmatics (evaluation, etc.)

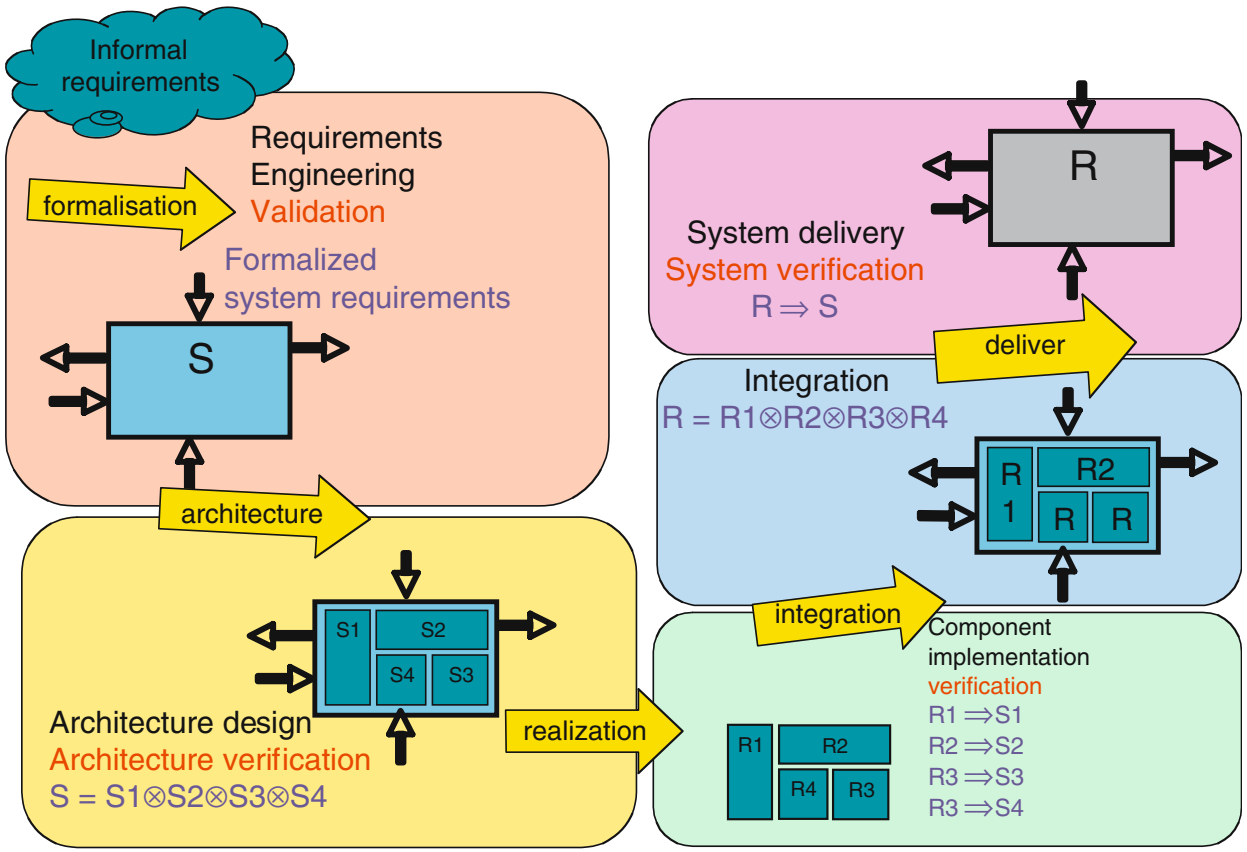
http://arca.di.uminho.pt/ac-1718

Software Engineering

Software development as one of the most complex but at the same time most effective tasks in the engineering of innovative applications:

- Software drives innovation in many application domains
- Appropriate software provides engineering solutions that can calculate results, communicate messages, control devices, animate and reason about all kinds of information
- Actually software is becoming everyware ...

Software Engineering



(illustration from [Broy, 2007])

Software Engineering

So, ... yet another module in the MFES profile?

Software architecture for reactive systems

characterised by

- a methodological shift: an architectural perspective
- a focus: on reactive systems

Challenges

Such trends entails a number of challenges to the way we think about SA

- new target: need for an architectural discipline for reactive systems
  (often complex, time critical, mobile, cyber-physical, etc ...)
- from composition to coordination (orchestration)
- relevance of wrappers and component adapters: integration *vs* incompatible assumptions about component interaction
- reconfigurability
- continued interaction as a first-class citizen and the main form of software composition

Reactive systems

Reactive system

system that computes by reacting to stimuli from its environment along its overall computation

- in contrast to sequential systems whose meaning is defined by the results of finite computations, the behaviour of reactive systems is mainly determined by interaction and mobility of non-terminating processes, evolving concurrently.
- observation $\equiv$ interaction
- behaviour $\equiv$ a structured record of interactions

Reactive systems

Concurrency vs interaction

$$x := 0;$$
$$x := x + 1 \mid x := x + 2$$

- both statements in parallel could read x before it is written
- which values can x take?
- which is the program outcome if exclusive access to memory and atomic execution of assignments is guaranteed?

Our approach

There is no general-purpose, universally tailored, approach to architectural design of complex and reactive systems

Therefore, the course

- introduces different models for reactive systems
- discusses their architectural design and analysis
- with (reasonable) tool support for modelling and analysis

Syllabus

- Introduction to software architecture
- Background
  - Introduction to transition systems (mCRL2)
  - Introduction to modal, hybrid and dynamic logic (mCRL2)
- Models and calculi of reactive systems
  - Timed (with real time constraints) (Uppaal)
- Architecture for reactive systems
  - Coordination-oriented architectural design
    - Paradigm: The Reo exogenous coordination model
    - Method: Compositional specification of the glue layer
  - Resource analysis of concurrent systems
    - Analysis of a Java-based language (ABS tool-set)

Pragmatics ...

- **Assessment:**
  - Test in June - 70 %
  - Group projects (2x) - 40 % (10+20)

http://arca.di.uminho.pt/ac-1718

- **Research context:** Projects
  - DALI — 2016-18
    on Dynamic logics for cyber-physical systems
  - TRUST — 2016-18
    on Trustworthy Software Design with Alloy

possible GRANTS available!
(with INL, U. Aveiro, CWI, INESC TEC)

Model checking

Recall "Especificação e Modelação":

- Modelling reactive systems – Kripke structures and NuSMV
- Specification – Temporal logics (LTL ~~and CTL/CTL*~~)
- Verification – Check if a formula holds in a system

SMV model checker

What we will see

- Labelled transition systems (LTS) as Kripke structures
  - Process algebra (not ~~Petri Nets~~ SMV) to define LTS
  - mCRL2 toolset to model (not SMV)
  - Equivalence of LTS
- Modal logics – generalising temporal logics (CTL*,CTL,LTL)
- Using mCRL2 toolset to verify properties

- Later: Timed-automata and UPPAAL model checker (CTL)

Model

$\mathfrak{M}, w \models \phi$ – **what does it mean?**

Model definition

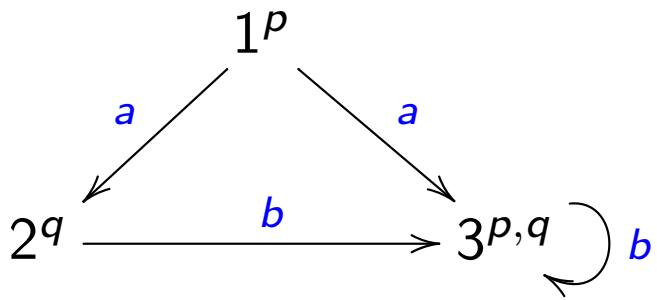
A model for the language is a pair $\mathfrak{M} = \langle \mathfrak{F}, V \rangle$, where

- $\mathfrak{F} = \langle W, \{R_m\}_{m \in \mathrm{MOD}} \rangle$ is a Kripke frame, ie, a non empty set W and a family R_m of binary relations (called *accessibility relations*) over W, one for each modality symbol $m \in \mathrm{MOD}$. Elements of W are called points, states, worlds or simply vertices in directed graphs.
- $V : \mathrm{PROP} \longrightarrow \mathcal{P}(W)$ is a valuation.

Kripke structures from last semester

- $\mathrm{MOD} = \{\mathbf{1}\}$
- (S, I, R, L) where $S = W$, $I = \{w\}$, $R = R_{\mathbf{1}}$, $L = V$
- $\mathfrak{F} = \langle W, R \rangle$ instead of $\mathfrak{F} = \langle W, \{R_m\}_{m \in \mathrm{MOD}} \rangle$

Example

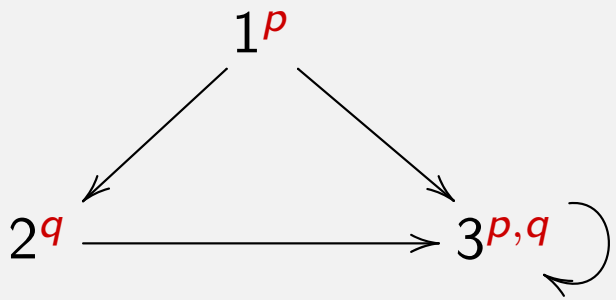
$\mathfrak{M} =$ 

$$
\begin{aligned}
W &= \{1, 2, 3\} \\
MOD &= \{a, b\} \\
R_a &= \{(1, 2), (1, 3)\} \\
R_b &= \{(2, 3), (3, 3)\} \\
V &= \{1 \mapsto \{p\}, \\
&\quad 2 \mapsto \{q\}, \\
&\quad 3 \mapsto \{p, q\}\}
\end{aligned}
$$

- $\mathfrak{M}, 1 \models p$
  means p holds in state 1
- $\mathfrak{M}, 2 \models [b]\, p$
  means p holds in every state reachable with b from 2.

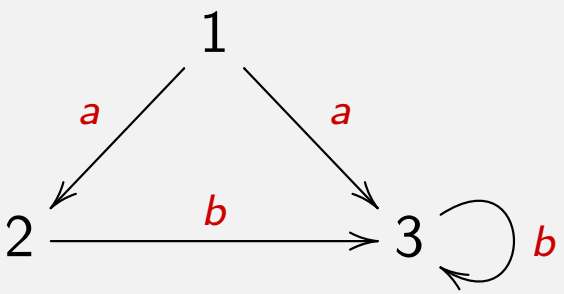
Key differences

Before



- emphasize on states - desired/forbidden states
- SMV language to generate models
- $\mathfrak{M}, 1 \models p$, $\mathfrak{M}, 1 \models F\,G\,p$

Now



- emphasize on actions - desired/forbidden sequences
- Process algebra to generate models
- $\mathfrak{M}, 2 \models [a]\,\text{false}$