# Better Abstractions for Reusable Components & Architectures

## *XBSE Before CBSE*

Christos Kloukinas
Department of Computing
City University London
http://www.soi.city.ac.uk/%7Ekloukin

CITY UNIVERSITY LONDON

Current modelling languages for MDE, e.g., UML, AADL, SysML, have been developed with the CBSE paradigm in mind. As such they offer very little support for connectors.
This is unfortunate, given how reusable connectors are and how important they are for deriving the non-functional, system properties that are of most interest at the architectural level - a level where components are usually barely defined at any detail.
This work attempts to review the situation and hopefully help refocus the architectural design support towards the use of connectors.

## 1 Introduction

Current CBSE languages do not support the definition of complex connectors and focus instead on specifying as many component characteristics as possible; given two resistors they will attempt to describe their impedance, weight, size, ... However, the overall impedance is a property of how these two resistors are connected and how they communicate, i.e., a property of the *connector* used.
Luck of full support for the *easy*, *explicit* definition of *complex* connectors hinders the understanding and analysis of our systems.

## 2 XBSE - *Changing* the Paradigm

**Algorithms + Data Structures = Programs** (1)

**Connectors + Components = Architectures** (2)

CBSE is great for lower level designs and development but at higher levels it is the connectors that matter most - so we need to develop an ***XBSE*** approach (Connector-Based...) to designing system architectures.
XBSE essentially follows Wirth's statement (1), turning it into (2) with connectors standing for algorithms and components for data structures. Indeed, in a top-down design approach, one starts with the connectors desired and then tries to select/develop the components that can be used with them. In a bottom-up design approach one starts with the components at hand and tries to select/develop connectors that can make use of these (and in reality, different parts of a system may well be designed following different approaches). Our insistence on the importance of connectors should, therefore, not be taken as an attempt to disparage the importance of components. It is only because current approaches have shown a quite imbalanced interest in the latter that we attempt to bring connectors back into the light. Both are needed equally - but connectors need highly more support at this point.

## 3 More Support for XBSE's Connectors

Connectors for XBSE need better support. They need *at least*†:

**Roles**

- Who participates in the connector's protocol?
- How should they interact?

If component's actions are first-order predicates, connector's are higher-order ones...

**Goals**

- What is the overall goal of the connector?
- What is the local goal of each role?

**Structural constraints**

- Who can assume which/how many role(s)?

†More details in the accompanying paper.

## 4 XBSE's Configuration & *Control Strategies*

Architectural *configuration* should go beyond simple port-role mappings; it needs to add support for the *control strategies* to be used by components in order to meet the goals of the roles they have assumed.

## 5 Control Strategies & Reusability - An Example

```
Phil(N) = sit
  ; f[N].take
   ; f[N+1].take
  ; eat
  ; f[N+1].release
   ; f[N].release
  ; think ; Phil
```
(a) Dining Philosopher

```
Phil(N) = sit
  ; (f[N].take)
   || f[N+1].take)
  ; eat
  ; (f[N+1].release)
   || f[N].release)
  ; think ; Phil
```
(b) Reusable Dining Philosopher

```
CS(N) = ( when (N%2=0) f[N+1].take ; f[N].take
        | when (N%2=1) f[N].take ; f[N+1].take )
```
(c) A control strategy for (b)

Combining control with behaviour as in (a) leads to problems and reduces reuse, while separating them as in (b) & (c) solves the problem and allows the component to be used in more contexts, e.g., without a butler component.

## 6 How Can All These Help?

- Better separate:
  - Behaviour (components), from
  - Interaction (connectors), and
  - Control (strategies).
- Increase component reuse.
- Reduce over-specification.
- Ease the system's control.
- Better connector description.
- Re-balance the architectural description of systems:
  - Ease communication - no more a million "wires" but a small set of well understood communication protocols.
  - Analysis depends on connectors - components sometimes abstracted to a small set of numbers, e.g., period, computation time, mean time between failure, etc.

**Connectors *impose* analyses - Components *endure* them...**

  - Explicitly identifying connectors makes the selection of analyses methods and their application simpler.

## 7 Future Work - What's Missing?

Almost everything... Some points in particular are:

- Should *channels* and *connectivity patterns* (star, bus, hypercube, etc.) be included? Seems so - but how best to do it?
- Provide a link with CBSE at the level below.
- Explore the connections with Aspect-Orientation - cross-cutting system properties, interaction patterns (might provide the link to CBSE).
- Model real systems - what are their connectors? Not the wires or basic communication technology like blackboard/bus but the real protocols, like Model-View-Controller (i.e., feedback control), Master-Slave, ...
- Good tool support (for more than just drawing boxes).

## Acknowledgments

This work has been funded by the European Commission Information Society Technologies Programme, as part of the projects:

- SERENITY (contract FP6-27587); and
- SLA@SOI (contract FP7-216556).

The author would also like to thank the anonymous reviewers for their very helpful feedback.