(Embedded Control Systems, Spring 2011)

Separation of Concerns: the "5Cs"

—

Levels of Complexity

Herman Bruyninckx
Mechanical Engineering, K.U.Leuven, Belgium

23 February 2011

Abstract

Not all embedded systems have the same level of complexity, but how does the complexity level influence the design and implementation of an embedded system? Is an industrial robot, a car, or an airplane a complex embedded system? What can we expect in the coming decade? In other words, what should you prepare yourself for as the future innovators in our industry, and what knowledge and skills will that require? This lecture also explains the role of the 5Cs in embedded systems design: Computation, Connectivity, Communication, Coordination, and Configuration.

1 Introduction

Embedded systems are getting more and more *complex*, integrating more and more functionalities, and these functionalities contain more an more *domain knowledge*. In addition, the state of the art in wired or wireless communication has reached a level of performance, reliability, and price/quality ratio that makes very distributed systems a *possibility*. However, this potential has not yet been really exploited, because the embedded control for such *systems-of-systems* has turned out to be extremely difficult. This document provides some insights about *where* these difficulties come from, about *how* the complexity can be *structured*. This structure in itself is, of course, not a solution, but it facilitates the description and design of possible solutions.

Figure 1 depicts the *running example* of the course: the *virtual traffic light*. This is a system-of-system consisting of multiple cars (and possibly also some *infrastructure* systems!), that come and go in rather unpredictable ways; in addition, each crossroad is *not a closed world* in itself, since the traffic that has to be *coordinated* on this crossroad comes from, and travels towards, other crossroad. Or, more in general, to and from other parts of the traffic system to which the *access* of vehicles has to be *coordinated*.

This problem of *access coordination to scarce resources* has become the number one design challenge for complex embedded systems of this decade.

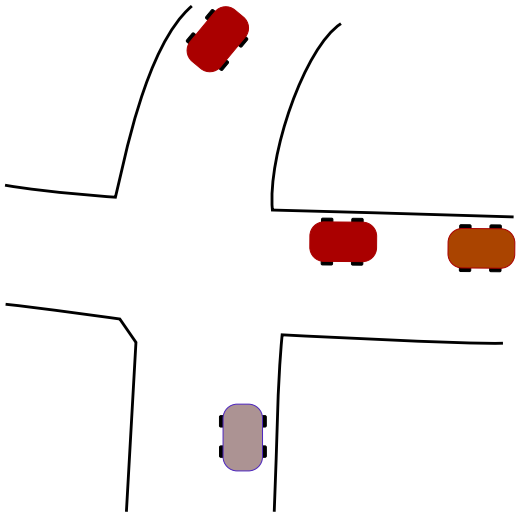


Figure 1: Sketch of the virtual traffic light situation: a crossroad lacks traffic lights to coordinate the access of the approaching cars to the *shared resource* of the crossroad, and that role has to be filled in by a distributed embedded control system over all vehicles.

2 Separation of Concerns — the "5Cs"

This Section contributes a *methodological approach* to structure knowledge along the lines of the *separation of concerns* idea, [1]. That idea has proven its efficiency in the hardware design and the software engineering of complex (not necessarily embedded!) system, and this document introduces only a small extension: it provides *five*[1] separate design "concerns" that (i) each focus on complementary aspects of the component-based system design, and (ii) have a systematic way of *composing* individual components into larger systems. These are the "five Cs" (Fig. 2):

- *Computation*: this is the core of a system's functionality, and provides an implementation (in hardware and/or software) of domain knowledge that provides the real added value of the overall system. The knowledge processing activity typically requires "read" and "write" access to *data* from sources outside of the component, as well as some form of synchronization between the computational activities in multiple components.

- *Communication*: this brings data towards the computational components that require it, with the right *quality of service*, i.e., time, bandwidth, latency, accuracy, priority, etc.

- *Connection*: this is the responsibility of system designers to specify which Computation components should Communicate with each other.

- *Coordination*: this functionality determines *how* the *activities* of all components in the system should work together.

- *Configuration*: this functionality allows users of the other above-mentioned concerns to influence their behaviour and performance, by giving concrete values to the provided *component-specific*

[1] Instead of four in the original work, [1]: their "Configuration" aspect is split into "Connection" (i.e., the knowledge representing how components are interconnected) and "Configuration" (i.e., the knowledge representing what are the "best" values to give to all configurable parameters in a component).

parameters; e.g., tuning control or estimation gains, determining Communication channels and their intercomponent interaction policies, providing hardware and software resources and taking care of their appropriate allocation, etc.

Similarly as already mentioned before, each of these 5C aspects to structure knowledge representation typically has its own knowledge to be represented. For example, the Communication between two components in a knowledge-driven embedded system-of-systems most often has its own Coordination and Configuration knowledge, such as: what communication protocol to use, and what are the best configurations of the messages (length, encoding, timing,...) that are being used in the communication.

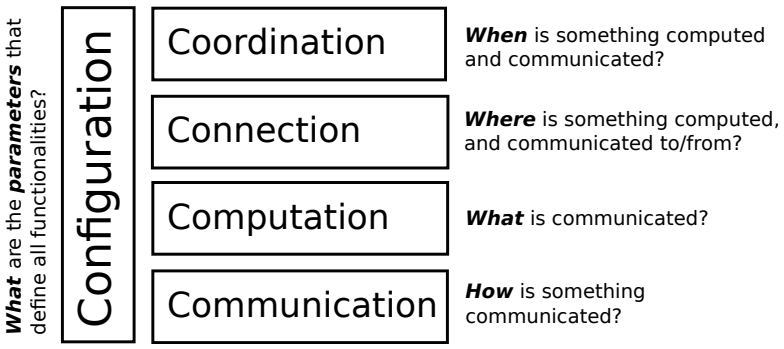


Figure 2: The "Five Cs" of separations of concerns in the design of complex robotics systems.

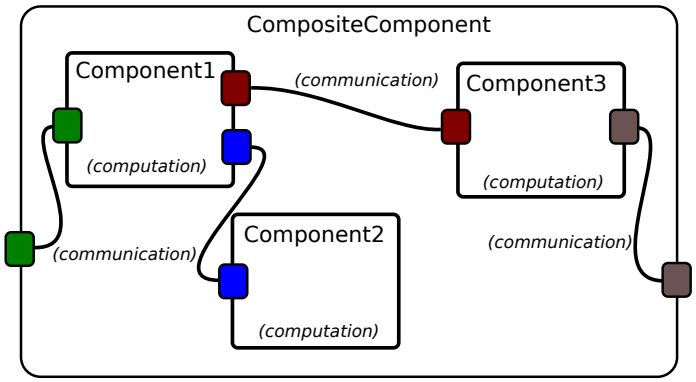
The Figures below illustrate the different ways *to compose* robotic systems from individual knowledge components:

- *structural composition*: that is, the *Connections* between components, and the *Communication* between the interconnected components.
- *discrete behaviour* of a component, that is, the internal "Coordination" of a component's own activities. The Coordination of the available between different components appear naturally as the Coordination of a *composite* component.
- *continuous behaviour* of a component, that is, the "dynamics" of the algorithms running in a component's activity.

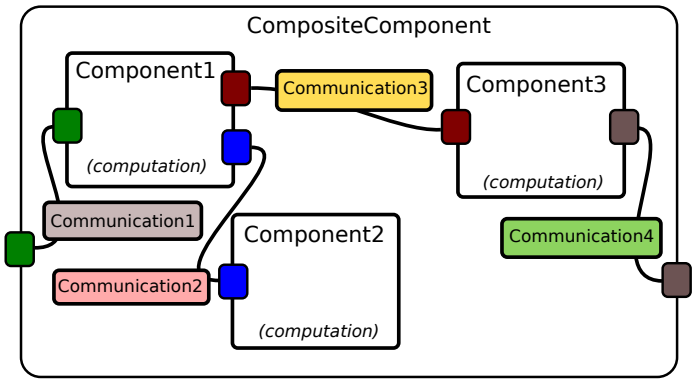
Of course, each of the above-mentioned composition operations must be given a specific *Configuration*.

***Coordination*, *Connection* and *Configuration* are the most important aspects where you will be able to add economic value to your embedded control system design.**

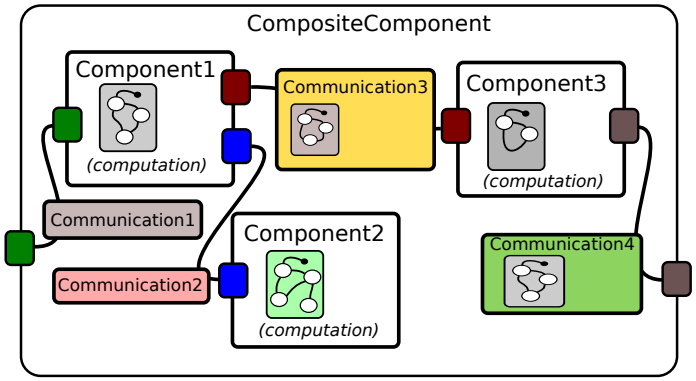
From your employees, sub-contractors, vendors,..., you should demand components that are pure *Computation* and *Communication*.



Structural component model



Structural component model + *Communication*



Structural model + *Communication* + *Coordination*

3 Levels of system coordination complexity

Figure 3 shows a sample of modern robots and systems of robots. Each of the individual robots in themselves is already a rather complex embedded control system, but, as mentioned before, the major current challenge is to solve the *coordination* problem. "Solving" the problem is a challenge with many different "design objective functions" to be optimized: robustness (agains a miriad of different "disturbances" the system is confronted with), safety, performance, efficiency, effectivity, etc. Different objective functions will drive the design in different directions, but from a coordination point of view, the complexity of the solution can be *structured* along the following lines:

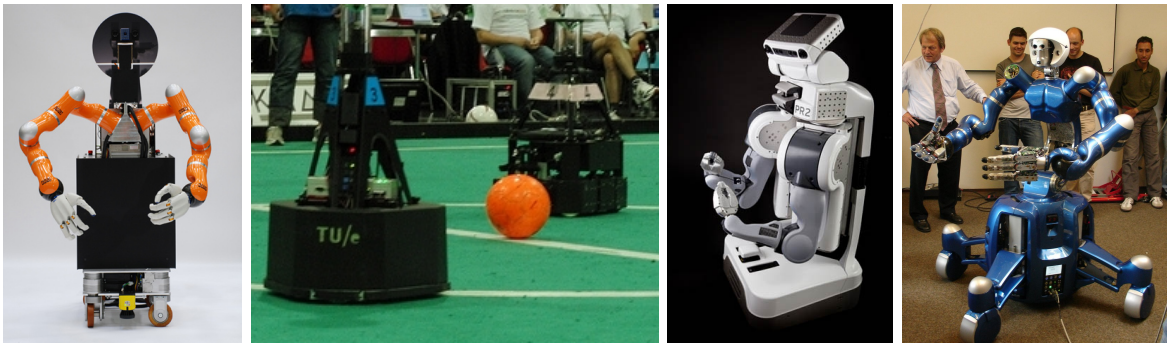


Figure 3: Modern robots and systems of robots.

S1 **All 5Cs centralised** E.g., (older) industrial robots.

S2 **Centralised coordination**. E.g., humanoid and mobile robots.

S3 **Decentralised coordination, but shared task**. E.g., RoboCup team.

S4 **Decentralised coordination, with individual tasks**. E.g., fleet of *robotised* wheelchairs in *"ambient intelligence"* building; the virtual traffic light.

Each higher level of system coordination complexity requires, roughly, and *order of magnitude* more efforts to design.

References

[1] M. Radestock and S. Eisenbach. Coordination in evolving systems. In *Trends in Distributed Systems. CORBA and Beyond*, pages 162–176. Springer-Verlag, 1996.