Notes on Architecture Description for Open Systems

Peter Henderson, September 2007

This paper is now considerably out-of-date and has been replaced by the papers at http://openpdq.com/OpenSystems.

Introduction

In describing an architecture for a large system, we use both structural and behavioural descriptions. We also arrange these descriptions hierarchically, describing the components individually and as a configuration that covers the whole system at a suitable level of abstraction.

When we want to discuss the openness of the solution, we need to do this in terms of procurable (and replaceable) components and their interfaces. For this purpose, the topmost description will be structural.

Openness, in this context, is the ability of a system to be built from independently procurable components. Thus, the interfaces to (and behaviour of) these components will need to be precisely specified and these specifications made available "openly" to the independent suppliers. The governance (or ownership) of the architecture and these specifications will need to be such that the openness is guaranteed. This enterprise issue, whilst crucial, is beyond the scope of this note.

To illustrate what we mean by a topmost structural description we will use the architecture of a (fictional) Roving Autonomous Vehicle (RAV see references). A simple structural description, which we will use throughout this report, is shown in Figure 1.

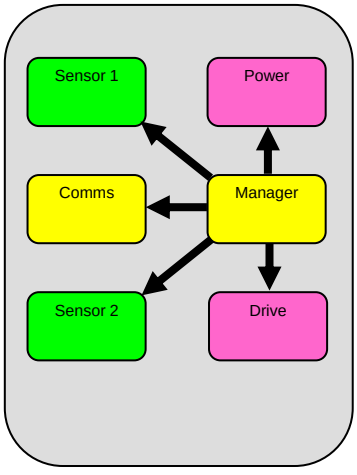

Figure 1 Structural Description of RAV

The RAV is for use on a distant planet. It will carry instruments and perform experiments under instructions from earth. Mostly it will operate autonomously, since the latency and bandwidth restrictions on communication mean that the only practical means of controlling the vehicle is to upload plans which it carries out independently.

The RAV comprises 7 components which are all independently procurable. The components, each of which comprises both hardware and software (i.e. each is a computer on a network), have the following behaviours

1. The Manager module is in charge. It runs continuously issuing commands and queries to the other modules
2. The Comms module communicates with earth
3. The Power module drives the wheels
4. The Drive module does the steering
5. The Sensor 1 module detects obstacles
6. The Sensor 2 module determines the geographic location of the vehicle
7. The Infrastructure module has all the mechanical components and the IT infrastructure embedded in it.

Sometimes we draw the structural model as in Figure 2, where we make it clear that the other modules communicate with each other through the infrastructure. This diagram is equally effective, but it loses the information about which modules communicate with each other. In both these diagrams the arrows are “uses” relationships or “client/server” relationships.

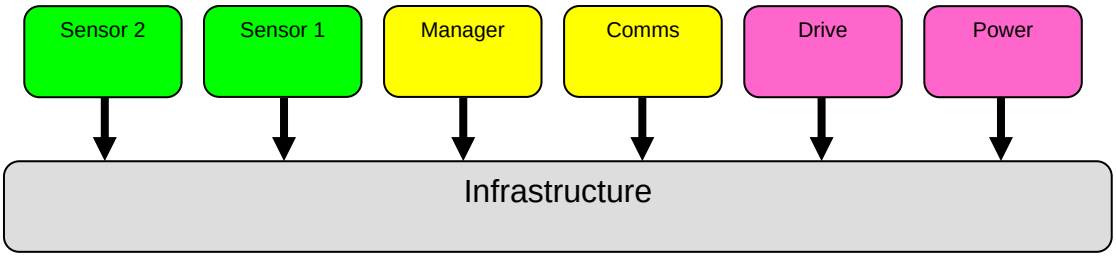

Figure 2 Alternative Structural Description

In order to understand the architecture and the sense in which this particular combination of components is indeed a solution to the given requirements, each component will need to have its interfaces and its behaviour defined. We will argue that, at this level of abstraction, the appropriate means for describing interfaces and behaviour will be to use narrative. However, lower down the hierarchy of descriptions, it will be necessary to be more precise about these descriptions, particularly the descriptions of the interfaces.

Observations on Architecture Description

Since our focus is on Openness, primarily we are interested in the extent to which modules can be removed and replaced with alternatives in order to effect the upgrading and eventual evolution of the solution. This leads us to the following observations on how the architecture of the solution should be described. I will illustrate these observations in the context of the RAV, later in the report.

1. The topmost architecture should be described as a collection of procurable modules

2. All the interfaces to these modules should be identified and specified
3. The behaviour of each module should be described
4. The way in which these component behaviours, taken together, combine to make the overall solution should be specified so that the behaviour *story* of the topmost component is clear in terms of the elementary stories of the embedded components
5. The requirements on the topmost component should be enumerated and stated
6. The induced requirements on the embedded components should be enumerated and stated.
7. The behavioural descriptions should match the functional requirements but do not replace them. The behavioural descriptions are the *stories* where the requirements are an enumeration of individual elements
8. Openness of the solution should be addressed directly by an argument that the details provided do indeed establish a base for independent procurement and upgradability of the product
9. A speculative family of solutions should be described showing that these components are indeed the basis for a long-lived evolving product
10. Particular attention should be paid to description of the (probably layered) infrastructure, which may or may not be a single component as suggested here. All the other components depend upon the infrastructure for their communication, meaning this may be the most difficult component to replace.

I have collected these observations above in order to have an easy reference for them. In the remainder of this document, I will illustrate their relevance in the context of the RAV example.

1. *The topmost architecture should be described as a collection of procurable modules*

For example, the Sensor 1 module will determine the proximity of an obstruction. How it does this may vary from supplier to supplier. As a pluggable component, it will need to be able to answer a query as to whether the path ahead is clear

2. *All the interfaces to these modules should be identified and specified*

The Comms module will have a single interface that allows the Manager module to enquire whether there are any messages from earth and to send a message to earth

3. *The behaviour of each module should be described*

The Comms module will always be listening for communications from earth and will need to store these messages until such time as the Manager module requests them. It will also need to store outgoing messages until some handshake from earth indicates

that they have been received. The integrity of the data in incoming and outgoing messages is an important responsibility of this module.

4. *The way in which these component behaviours, taken together, combine to make the overall solution should be specified so that the behaviour story of the topmost component is clear in terms of the elementary stories of the embedded components*

The story of the RAV is important, if the architecture as a whole is going to be understood by the various stakeholders. An attempt at this story follows.

The RAV is capable of moving around the planet autonomously and can avoid obstacles including craters and cliffs without damaging itself. Controllers on earth will periodically upload plans in the form of a set of tasks of the form – go to a specific location, carry out a specific experiment, and report your results when finished. The RAV is controlled by the Manager module, which in this first version controls everything. In carrying out a plan it can issue instructions to move and to steer the vehicle. Periodically it will ask the Sensor 1 module if there are obstructions, so it will be able to take appropriate action. Periodically it will ask the Sensor 2 for its current location so that it can steer appropriately and so that it can determine when it has arrived at its destination. Periodically it will ask the Comms module if there are any messages and periodically it will report its health to earth and send any outstanding messages.

5. *The requirements on the topmost component should be enumerated and stated*

The topmost requirements can be derived from the story but are not a good substitute for it. Rather, their enumeration forms a redundant and intentionally complete statement of function and performance.

Examples of topmost requirements on the RAV are

- Can receive and acknowledge messages of up to 1MB with guaranteed integrity
- Can move to a predetermined location without the need for instructions from earth
- Will send health report every hour

6. *The induced requirements on the embedded components should be enumerated and stated.*

In order that the embedded modules can be independently procured, the induced requirements on each of them needs to be clearly stated in terms of the communications across the interfaces they supply.

The requirement upon the RAV to receive and acknowledge messages induces a requirement upon the Comms module to be always listening, to detect incoming messages, to validate their integrity and to store them (until required). It must also acknowledge receipt in a way that reports its integrity metric. Only when end-to-end integrity has been established should the Comms module release the message (when asked) for action.

7. *The behavioural descriptions should match the functional requirements but do not replace them. The behavioural descriptions are the stories where the requirements are an enumeration of individual elements*

The story is a fabrication, a logical rendering of the use-cases that gives the stakeholders an opportunity to understand the whole system. Even suppliers of embedded components need this story, so they can see how their component contributes to the whole and see the potential for innovation.

For example, the supplier of the Power module may see that, in the context of the vehicle having to stop for an obstruction, some form of advanced movement (e.g. deceleration) may be more effective than simply stopping on instruction.

8. *Openness of the solution should be addressed directly by an argument that the details provided do indeed establish a base for independent procurement and upgradability of the product*

There can be no stronger argument than the fact that the procured component has been deployed in various independent systems. Failing this, where components are bespoke as many will be in the RAV, independent means of ensuring that the component does indeed conform to the requirements on its interfaces should be established. In the case of the RAV, a test-harness for the Manage module can include a simulated interface in which the module controls a simulated vehicle in a video-game-like scenario. The ownership of the conformance harness should be independent of the supplier of the Manage module.

9. *A speculative family of solutions should be described showing that these components are indeed the basis for a long-lived evolving product*

The RAV family includes RAV1.0 and RAV1.2 which are structured as in Figure 1, the second version having more elaborate behaviour. RAV2.0 and successors were restructured to separate off the collision-avoiding functionality, so that that became part of the basic vehicle allowing the revised Manage module to concentrate on carrying out plans. RAV3.0 will be developed to allow a group of RAVs to work collaboratively. For example they will share information so that valuable data will not be lost if one machine breaks or has to be rebooted. Some members of the RAV3.0

family will be specialised, for example being able to rove further by being lighter in weight.

10. *Particular attention should be paid to description of the (probably layered) infrastructure, which may or may not be a single component as suggested here. All the other components depend upon the infrastructure for their communication, meaning this may be the most difficult component to replace.*

[remember, this is fiction].The RAV was originally designed so that the IT infrastructure that supported the other modules provided a simple message-passing mechanism (a bespoke protocol over TCP/IP on the installed ethernet). Between version RAV1.0 and RAV1.2 this was replaced by an RMI solution, thus requiring every module to have a computer with a JVM installed. Other infrastructures had been considered, including Jini and JMS and in fact a bespoke messaging system was designed for use on top of RMI because it loosened the coupling. A messaging or data distribution solution is being contemplated for RAV3.0, since ideally onboard and offboard communication should be the same. Changes to the infrastructure are expensive because they require changes to other modules or the development of wrappers or adapters. Consequently an infrastructure decision is often one it is not economic to change. The RMI decision for RAV1.2 which persisted into RAV2.0 has become a cause for concern.

Other Issues

The following topics need further attention in the context of this report

Stakeholders – the principal stakeholder considered here is the supplier of the solution, whereas there are many stakeholders whose concerns need to be addressed, not least the customer and the eventual end-user. The extent to which they need to understand the architecture needs consideration

Openness – there are issues of openness, not least governance, which have not been addressed and which cannot be taken independently of the architecture. See my earlier note on this. There is the need to take that earlier note forward in the future, especially concerning standards bodies and lessons from open source.

Concurrent Engineering, Shared Models, Rich Pictures – there are lessons from concurrent engineering and soft systems which bear upon the development of a family of solutions and on the proper consideration of requirements (both functional and non-functional) which need to be brought forward. In particular I am thinking of the importance of shared models to any engineering solution where there are many disciplines involved.

References

RAV – http://openPDQ.com/RAV
Openness – http://openPDQ.com/openness