WICSA 5 Working Group Report

Architecture Description Languages in Practice

Eoin Woods
eoin@copse.org.uk

Rich Hilliard
r.hilliard@computer.org

This short document reports the content and results of the "*Architecture Description Languages in Practice*" working group held on 9th and 10th November 2005 at the WICSA 5 conference, in Pittsburgh, Pennsylvania, USA.

Introduction

The aim of the working group was to consider the relevance of contemporary architecture description languages (ADLs) and the future of ADL research. In order to focus discussion, the working group aimed to answer the following two questions:

- Are ADLs relevant to contemporary software architecture practice?
- What is the likely future of ADL research and how could it affect working software architects?

The meeting was 3 hours, 15 minutes long, structured into two sessions, each containing two short paper presentations (of 5-10 minutes each) followed by a group discussion, loosely structured around the two questions above.

The four papers presented at the working group were as follows:

- *Mapping ADLs into UML 2 Using a Meta ADL*, Adel Smeda, Mourad Oussalah and Tahar Khammaci.
- *Intelligent Instrument Design With ArchWare ADL*, Jerome Revillard, Sorana Cimpan, Eric Benoit and Flavio Oquendo.
- *DUALLY: Putting in Synergy UML 2.0 and ADLs*, Henry Muccini. Paola Inverardi and Patrizio Pelliccione.
- *An MDA Approach for a Multi-Layered Satellite On-Board Software Architecture*, Walter Dos Santos and Adilson Cunha.

Session 1

Walter dos Santos, from INPE's Space Systems Division in Brasil, opened the session by presenting his paper (*An MDA Approach for a Multi-Layered Satellite On-Board Software Architecture*). The paper described a project that applied real-time UML (within the context of an MDA project) to the problem of building highly reliable on-board satellite control software. They found that an approach using real-time UML, based on "4+1 views", was very effective for sharing knowledge between important stakeholders and the overall experience of using real-time UML for architectural description was positive. The project is still in progress and so final results are not yet available.

The second paper (*DUALLY: Putting in Synergy UML 2.0 and ADLs*) was presented by Patrizio Pelliccione from the University of Luxemburg. This paper described a new ADL, aimed at bridging the gap between current ADLs and UML. The motivation for the creation of the language was the experience of observing practitioners creating semi-formal models in UML, but then being unable to use the models for analysis due to their lack of precise semantics. DUALLY aims to resolve this problem by identifying a core set of UML concepts needed for software architecture modelling and providing an extensibility mechanism to allow the language to be extended while maintaining precise semantics.

The session then moved onto a discussion of the current state of ADL adoption. Some of the conclusions that resulted from the discussion included:

- A key problem is that there is a mismatch between the way architects work and the facilities provided by ADLs. For example most architects work with a number of views of their system and need to model many different structures (including functional, information, deployment and concurrency structure). However, most ADLs only support a single view and only provide the ability to model components and connectors (i.e. a functional structure).
- A related problem is that there is often a mismatch of expectations and assumptions between those who create ADLs and those who might use ADLs. An example of this is the assumption that architects want to analyse their models, when in reality it is far more common that architects create models primarily for communication and documentation purposes.
- Different people need different things from an ADL and perhaps ADLs should be specific to a particular quality property to allow them to be specialised and so more useful?

The group then stopped to ask the question, where are ADLs actually useful? The answers that were suggested included:

- For communication and understanding between stakeholders.
- In certain domains such as weapons systems and automotive systems (these being suggested because ADLs have been applied to large industrial projects in these areas).
- For describing small case studies for researchers. Of course, the point is that the case studies are often to illustrate the use of a technique, such as an ADL, and so the techniques should scale up. However, as of today, a lot of them don't and so their usefulness is limited to case-study sized problems.

Finally, the session turned its attention to the question as to whether UML is actually an ADL or not. The conclusion arrived at was that ADL experts don't consider it to be an ADL but in reality it is the most widely used notation for architectural description (apart from PowerPoint boxes and lines of course!) One of the reasons for this is probably that UML is primarily a visual language and if it was textual (like many ADLs) it probably wouldn't be used nearly as widely.

At this point, the working group reconvened in joint session with the "Documenting Software Architectures" group, which was meeting in the next room and discussing related subjects. After recapping the discussions from each group, the discussion moved (again) to UML and asked what would make UML a good ADL and why people would use an ADL.

When considering what would make UML a good ADL, the consensus was that it would need a truly extensible meta-model, textual and visual notations[1], first class views and domain specific representations or versions.

When considering why people would use an ADL, the consensus of the group was that people use architectural descriptions for analysis or for documentation (and a few may use them for both). If the main need is analysis, then UML probably isn't a very good base to build on. If the need if documentation then UML could be a perfectly good base to improve upon.

Session 2

The second session was opened by Tahar Kammaci presenting his paper (*Mapping ADLs into UML 2 Using a Meta ADL*) which presented a meta-ADL intended to allow other ADLs to be easily mapped to UML 2.0. The motivation for creating the language was the observation that ADLs tend to have well defined semantics but often aren't used due to poor tool support and usability problems. In contrast, UML is much more widely used as it has largely

[1] Although, to be fair, it already as XMI as a textual notation, although not a very readable one.

addressed these problems, but has quite weakly defined semantics for software architecture modelling. The meta-language can be easily mapped to MOF and so then to UML and should allow easier mapping, comparison and combining of ADLs.

Sorana Cimpan presented the second paper in the session (*Intelligent Instrument Design with ArchWare ADL*), which described the use of the ArchWare ADL in the instrument control software domain. This project was based at CERN in Switzerland and needed to address the problem of experts from the instrument control domain creating software without any direct input from software specialists. The domain is complex enough that few people have a high level of knowledge in both instrument control and software development and so there is a need to guide the domain specialists to produce well designed software that conforms to an overall architectural design. The solution was to use ArchWare ADL to create the definition of the desired architectural style and for the domain specialists to use these definitions to design their software. Java was then generated from the software models. This allowed the software specialists to change the definition of the style over time, without impacting the domain specific design. The project has been successful and much of the software has now been developed and used.

As in the previous session, the group then began a discussion, this time asking why ADLs aren't generally used by practicing architects and what could be done about this.

The conclusions drawn about the lack of ADL use were:

- Most ADLs are quite restrictive and impose a particular architectural model on the architect, which often isn't appropriate.
- All the ADLs that the group were aware of only support a single view, whereas one of the research ideas that practitioners have actually embraced is the use of multiple views!
- Most ADLs are very generic, with a lack of domain specialisation, which means that they do not serve the specific needs of any domain all that well.
- As with most research products, ADLs are developed in a research context and little thought is given to a "rollout plan" to allow technology transfer.
- Few ADLs are supported with good tools and so they are often awkward to use in the normal workplace.

After considering these factors, the group concluded that some of the ways in which the lack of ADL adoption could be addressed included:

- Specialising languages to particular domains (as AADL is specialised to embedded, real-time systems), in order to better serve their needs.
- Planning technology transfer as part of the project that creates the ADL.
- Aligning research work directly with technology trends (for example, by participating in the Java JSR process) to ensure that the ADLs produced are congruent with current industrial directions.
- Linking ADLs directly to technologies that practitioners want to use in order to make them directly useful.
- Recognising that tool support is important and developing good tools as part of the research programme. (In fact, the development of effective tools might itself be an interesting research project.)
- Encouraging researchers to spend time studying what architects actually do in order to understand where ADLs could be applied and they would be used.
- Once ready for early adoption, the results of ADL research should be published (or perhaps just publicised) in such a way that practitioners will be aware of them. Most practitioners do not attend specialised workshops or conferences like WICSA, so alternative communication channels need to be considered.

After this discussion, the group then again convened in a join session with the "Documenting Software Architectures" group and had a fairly wide-ranging discussion of the conclusions that

each group had drawn. Some of the consensus points that emerged from the discussion were:

- Decomposition often isn't important outside the module (i.e. code) view, as architects only want to think about the first level of detail.
- It's not clear that sophisticated tool support is really needed if in reality we need 4 diagrams and some supporting text.
- Adopting a development model like the open source community uses might focus effort in promising places. Open source projects don't develop very far until they know they are solving a real problem because people are using the software. In fact, could we interest the open source community in a practical ADL, and if so, what is the compelling reason for them to use it?
- The lack of case studies on ADL use is a concern and this suggests that more effort needs to go into understanding what architects actually do.
- Again, we need to be clear what ADLs are used for and many architects really want a "sketching" tool (rather than a blueprinting tool).
- Code generation is a potentially promising area, but it's not necessarily the answer to adoption because languages have a lot of momentum by themselves and often spawn their own specific solutions to problems.

At this point, we closed the two working groups, and reported these conclusions at the closing session of the conference.

Conclusions

This working group started out to answer questions as to the current relevance and likely future of ADL research. During the discussions, the focus changed slightly and as a result, we gained a consensus on answers to two specific questions about architecture description languages:

- Why aren't ADLs used in practice?
- How could ADL researchers address this current lack of use?

When considering why ADLs aren't used in practice, the group's opinion is that the main reasons are:

- ADLs make restrictive assumptions that limit their use and cause problems with adoption.
- Most ADLs are single view oriented (usually a components and connectors view) whereas most practitioners use multiple views to model different sorts of architectural structure (noted exceptions are Wright and Rapide).
- Few ADLs are supported by good tools. Where tool support is present, it is rarely robust or compatible with the architect's core toolset (MS Office, Eclipse, Visual Studio, Magic Draw, Rational Software Architect and so on).
- Most ADLs are general purpose and so lack domain specific constructs that are important to architects (such as client, application server, message listener, database, message queue and so on in the information systems domain).
- There is rarely a "rollout plan" or technology transfer programme to encourage the adoption of new ADLs.

When considering what could be changed in ADL research to address the current low level of adoption, the group consensus was:

- Practitioners seek ADLs linked to the technologies they are already using (such as UML and Java) and so ADLs should be linked to specific mainstream technologies to allow them to be immediately useful and help the technology to be transferred.

- Domain specific ADLs are likely to have more chance of success than very general ones and so this specialisation should be part of the research, not left as an “exercise for the reader”.
- It isn’t clear that ADL research groups really understand what architects do, or need from an ADL and so there is a need for user studies to identify the key usage scenarios for ADLs.
- An ADL should have a graphical representation. This is key for many stakeholders.
- ADLs should be supported by high quality, simple tools that integrate well with existing desktop environments. Examples would be Visio stencils, Magic Draw extensions and Visual Studio or Eclipse plug-ins.
- ADL researchers can improve technology transfer by:
    - Aligning ADLs with industry movements like the agile community or enterprise architecture.
    - Building a community of users and contributors in the “viral” open source style.
    - Working with relevant specification processes (like Java Specification Requests, OMG and W3C standards groups and so on).

In summary, the group feels that, while a lot of good research has been done to lay foundations for architecture description languages, the next phase of research needs to align ADLs more closely with industrial software architecture practice in order to aid adoption and ensure the relevance of the research.