**Critique\_4**

**By Anvesh Tummala (16173144)**

**Critique on “ArchJava: Connecting Software Architecture to Implementation”:**

I would like to analyze what extent ArchJava addresses the following

1. ***How ArchJava reduces the inconsistency between architecture and implementation?***

***Where does ADL’s stand in this process?***

ADL’s are only good at capturing the architectural properties. To make sure these properties are implemented we need a *consistent system* on top of it. ***ArchJava uses Darwin like ADL*** for architecture specifications. From this we can say ArchJava is limited to maintaining consistency among structural elements and interconnection.

***Where does MILL’s stand in this process?***

They are used for specifying the composition i.e. the relationship among components.

* It maintains communication integrity, no less no more connections in code, architecture. It also checks the consistency of connections by verifying each required method is uniquely bounded with a provided method. Allows communication only though existing connections.

1. ***How it preserves the architecture properties in code?***

* As it is an extension to Java that unifies architecture with implementation, we can assure that our code preserves architecture constraints.
* As said it uses ***Darwin like ADL*** that preserves architecture properties, these properties are directly mapped to code and it is make sure by ArchJava compilers.

1. ***How it support software evolution?***

* It supports **only structural evolutions even at dynamically changing distributed architectures** like run time component creation.
* Dynamic connections using “smart connector”, multiple connection using “*Port Interface*.”
* Dynamic component creation.
* Removing connections, components using garbage collection (based on reachability).
* Provided and required interfaces helps in better understanding the communications in system and helps in evolving them.

In the evaluation using Aphyds*, “The resulting architecture was finer grained than the developer’s conceptual architecture”* it seems more exaggerating because they look at the code for finding classes that can map to developers conceptual architecture, How can they get the more finer mapping? I understood there can be more subclass hierarchy, but they should only look to the level of mapping they have to look in the code. *“I don’t understand why they need to combine component classes into higher-level components.”* In those cases they should already have such high-level component before.

The case study they have done is only limited to the structural consistency in the implementation, it does not speak on consistency when behavioral changes occurs at architectural level. So, it is only limited to maintain structural consistency between architecture and code.

On the other side, *as we are fixing the programming language as Java, how architectural styles, patterns having constraints to use procedural and logical programming are implemented?* So it is not a solution for all, it is just a part of solution specific to software’s build using Java as programming language.

**Critique on “Enhancing Architecture-Implementation Conformance with Change Management and Support for Behavioral Mapping”:**

In the above ArchJava paper we have seen some limitations in mapping behavioral changes at architectural level. On the top, we have seen the architecture and code both are linked together as a single artifact, the changes in overwrites the other, which is really a big problem in the change management.

I would like to analyze how 1.x-way implementation solves the above issues, how best is the solution and is there a scope for better solution?

The change management by separating the architecture code is pretty handy. It is a simple solution for the above problem of overwriting the existing artifacts work. Here changes are only allowed on architecture (1) and part of the code(x- use defined code). In other words, we can say changes are allowed in both code and architecture, but **the way they are propagated to other artifact is completely safe**. Change in architecture effects the generated code and rarely makes some change notifications to user defined code. This is a situation where we need some developer’s active participation. It is not an issue, but I would think of making changes on to user defined code such a way that we won’t delete any part of code, instead we comment or add code when needed, which I think is better solution because it only needs passive participation from developers. We can still use notifications for alerting developers.

Introducing behavioral mapping at architecture level is really a significant idea in the mapping process. The incremental idea used in the mapping of sequence diagrams by just looking for the direct calls and finally mapping a sequence diagrams on to generated code is really a new start for behavioral mapping. **As this 1.x model uses xADL it is even possible for capturing dynamic properties using extentions. As** we are keeping the behavioral mapping in the architecture generated code, isn’t it a problem if programmer want to change (or add) the behavior in his user defined code? I mean we have some behaviors at generated code and some are at user defined code. **My main question is how well this 1.x-model maintain the conformance of changes in the behavioral mappings.**

**I think this paper does not look for “How changes in user defined code (that leads to splitting the components or changes in communication) are propagated to architecture prescribed code?” Why is this not important? If it does not exists, we always needs a reverse engineering after certain level of modifications in use defined code.** Even the “Registration” from code editor is only looking for subscribing the notifications from architecture changes. I understand the solution for this answer depends on way an architecture and code are mapped. **Like in ArchJava as both architecture and code unified to a single level, it is possible to remap the architecture changes when a code changes. But here we don’t have any such mappings between two artifacts.**

***I would like to suggest the better solution would be the one that can use both the ArchJava (code to architecture mapping) and 1.x mapping (generated and user defined code with limited write access) with change notifications. It might impart some limitations like in ArchJava, not supporting multiple styles. I understand that bringing code and architecture to a single level imparts a lot of issues. But we need a kind of programming language that abstracts all the existing programming paradigms and allows extension to any specific language (so allows all arch styles), which is highly hypothetical, combined with our 1.x model will be a better solution.***