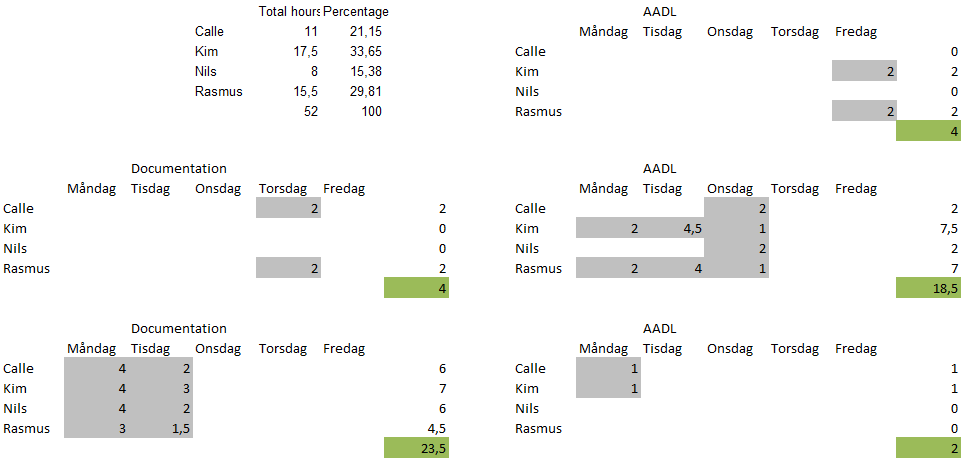
GA04 Deliverable

M.U.P

massively underdeveloped project



Rasmus Tilljander - rati10@student.bth.se  
Nils Forsman - nifo08@student.bth.se  
Calle Ketola - cake10@student.bth.se  
Kim Hansson - kiha10@student.bth.se

## Introduction

In this assignment we were to evaluate a premade architecture of a forest harvesting machine control system called Blunderjack using the formal architecture evaluation AADL with the program OSATE. We were to complete the AADL model after certain specifications and then evaluate it using two specific scenarios we were given.

The AADL model does fulfil the requirements under certain conditions. However it is quite possible that more scenarios would have to be tested to be certain that it does so in all circumstances.

For example missing sensors and overall robustness seems hard to test with AADL.

## Evaluation result

1. The changes made to the model

Each subsystem now has its own budget for MIPS and RAM resources, based upon the specification given in the assignment.

The predefined flow paths for Device Manager and Safety Manager have been connected successfully. Furthermore, we added a rotation\_inout\_flow to the Safety Manager for usage in scenario 2.

In the system configuration we created 3 different flow paths, the first one (F1) for scenario 1 and the last two (F2 & F3) for scenario 2.

We assigned the subsystems to processor and their memory banks based upon configuration 2 & 3 as can be seen in our AADL model. Based upon the specification we concluded that we could not assign the subsystems in configuration 1. We can find no difference in performance between configuration 2 and 3. Unless the memory requirements on the system changes we believe it would be more interesting to use configuration 2, since it uses all memory but has more MIPS capacity over.

2. The result of the evaluations

Our result for scenario 1 the highest latency was 265ms, this was well below the requirement given. Our conclusion is that given the current system architecture the system fulfils the requirements.

The result for scenario 2 had a latency of 435ms with the asynchronous test and a latency of 345ms with the synchronous test. Our conclusion is that for the system to fulfil the requirements, the system has to be run synchronous.

## Challenges:

### Understanding

In the beginning it was hard to understand the syntax, knowing which were part of the syntax and which parts were created by the architect.

Knowing the difference between different parts of the syntax and which variables one should use in which situations was far from clear. For example the difference between different types of flows and what they meant. Most of the time we relied on trial and error to find a solution we could be satisfied was correct.

A great example of this was the end flows for trying the latencies of the scenarios. We knew where to put the flows but we did not know what kind of flows they were or what kind we had access to. It was pure luck that we found a document describing a similar scenario with the flows implemented.

Overall the description was more than adequate and gave enough information to understand the system. Understanding scenario 2, however, was not very easy. The meaning of the flow described felt like it could be interpreted in many ways. Our interpretation was that it wasn't an end-to-end flow in reality but rather a flow with two sources and two sinks and a common connection point.

### Extending

Knowing whether what you had done was correct or not was very hard. Sure the errors and warnings sometimes went away, but that does not really have to mean anything.

A concrete example of how things were suppose to look would have been good. For example some kind of test scenario which would include expected output to validate whether you have done it correctly or not.

Also knowing how to test the latency and see the budget results was not obvious. Overall a better document for getting starting and understanding AADL would be better, something more compact than the several scattered tutorials we received, such as a single cheat-sheet for AADL.

## Reflection

The evaluation done in AADL is very different from the one done in assignment 3 in that it mostly tests latency and budget issues which would have been impossible with the evaluation done in assignment 3.

It cannot however be used to develop the architecture in the same way that the evaluation method from assignment 3 could. It will only tell you that your system does not fulfill the requirements in a certain area, not what the exact problem is.

It is certainly good to use a formal architecture evaluation if you have the budget to train your team in it. It can help with such issues that could arise from having development switch to a different team that would not otherwise understand the evaluation done in a non-formal way.

With the formal specification method we can validate the model, which is not possible with the evaluation method used in assignment 3.

To summarize we think that it depends a lot on what kind of system you are building and what kind of budget you have at your disposal. For larger development scenarios you might find it more efficient to use a combination of both to first help develop the model and then validate it instead of just focusing on one aspect.