AADL Display System Model Description

The AADL Display System model supplied by Rockwell Collins was part of a research effort that had two objectives. The first objective was to show that the AADL language was robust enough to model a slice of an IMA cockpit display system using Rockwell Collin's proprietary switched Ethernet LAN, call AFDX. The second objective, which required the successful achievement of the first, was on the analysis, both CPU and LAN, of this AADL model. This analysis was performed by Rockwell Scientific Company (RSC), a research company partly owned by Rockwell Collins, using a toolset that they had been developing to support our product groups using the AFDX. Both of these objectives were met.

As a member of the AADL working group I was aware of the architectural components of AADL. I began doing some conceptual modeling with Visio before a compiler was available, to endeavor to achieve the best mapping of the physical AADL components to actual hardware. This would enable me to attach project-specific properties and values to these hardware components that would be needed for the analysis toolset. This was the driver for modeling the nodes and switch as AADL device components. There was also the desire, on my part, to have the model reflect the actual entities in the system so that an AFDX system's architect could look at the model and immediately understand the physical and logical layout. In addition to naming these devices appropriate names as nodes or the switch, a project-specific property was created for AADL devices which was an enumeration of the device type: Node or Switch. This aided the analysis tool in parsing the XML model for absolute clarity of the device type. From a hardware standpoint there is an input and output port to/from each node to the switch. This was modeled using a pair of AADL bus components for each node. There was necessarily a node connected to each of the IMA processors, 5 in all, modeling only one side of a cockpit.

From the logical, or software, viewpoint the model was laid out from previous Display project's data. The necessary processes (virtual machines), their threads, and the thread data needs were fairly common across projects. This data was migrated to Excel, where Perl scripts were used to traverse the data to generate the AADL model. The physical, hardware, components were implied from the software, so that the entire model was generated from that data. One limitation of the beta version of the analysis toolset that was being used forced all communication to be point-to-point, unicast. These systems normally have an abundance of multicast data, particularly for redundant crosstalk communication. A significant amount of data traffic between threads was removed from the Excel file. As a counter-balance to that loss, traffic in another arena was increased. This beta version of the analysis tool did not support aperiodic thread scheduling and thus did not support aperiodic bus traffic. This was countered by a change in the Excel data of all aperiodic threads/data to periodic threads running at 5 Hz. This rate was chosen, because there were no other threads running at this rate and would facilitate a conversion of the Excel data back to aperiodic once the analysis tool could support that.

There are three major improvements that could be made to the model that was generated. The later two relate to the Perl scripting and could be easily rectified. These will be discussed below:

AADL Display System Model Description

1. The data flows that were generated for this system were purely “mechanical”. They reflected the flow of each data from its source thread, through its hardware-mapped node, through the switch and back through the appropriate node to reach its destination thread. There was no effort made related to domain analysis that would depict functional data flows through the system; e.g. a sensed button press by a thread which would communicate that with another thread. The second thread would compute a new flight mode for the aircraft and then pass this new mode to a destination display-related thread for annunciation.

2. With the Excel data specifying the mapping of threads to processes and processes to processors, it seemed practical and more appealing to the top level architecture, to aggregate all of the processes that would map to a specific processor into a system component that would then match one-for-one with the hardware system components which aggregated the processor, memory and node. What was missing from this decision was the fact that the analysis tool would perform more than a pass/fail for the given CPU and LAN configuration. If a configuration was not schedulable, for either or both reasons, it would provide the analyst with reconfigured systems which would be schedulable for both. Any “movement” of processes to other processors with an analysis-driven reconfiguration would necessarily require some major rework of these software systems, both from a process/thread standpoint and from a data communication standpoint. A better approach would have removed these software system aggregates and moved the processes as direct subcomponents of the top level system. There is one drawback to this approach. There are certain sets of processes that must execute on the same processor. To enforce this with the AADL mapping properties would require that these sets be grouped back into a software system, but in this case the aggregation is intentional, actually required, with mapping properties that would ensure this. The analysis tool had a provision for specifying this, but the interface from the AADL mapping restrictions and the toolset mapping provision was not developed for this particular research effort. (There is some debate as to whether there should be a new component added to AADL, Process Groups, which would be a concise means of modeling a group/set of processes that should be grouped together, from a functional sense, like having common mapping requirements).

3. The third area of improvement becomes immediately apparent if a reconfiguration of the system is required by the analysis toolset. The data port, port group, connection and flow names all used a “physical” naming convention instead of a “logical” one. The names specified the source and/or destination processor/node names instead of the source and/or destination thread and virtual machine names. Any processor reconfiguration from the analysis tool would result in major re-writes of the data through the model when using a “physical” naming convention. With a “logical” naming convention there would only be some remapping through the appropriate new nodes, and processors, of this new configuration. The impact to the model would be drastically reduced.