Architectural Modeling and Analysis for Safety Engineering

Danielle Stewart[1], Michael W. Whalen[1], Darren Cofer[2], and Mats P.E. Heimdahl[1]

[1] University of Minnesota
Department of Computer Science and Engineering
200 Union Street
Minneapolis, MN, 55455, USA
dkstewar, whalen@cs.umn.edu
[2] Rockwell Collins
Advanced Technology Center
400 Collins Rd. NE
Cedar Rapids, IA, 52498, USA
darren.cofer@rockwellcollins.com

Abstract. Architecture description languages such as AADL allow systems engineers to specify the structure of system architectures and perform several analyses over them, including schedulability, resource analysis, and information flow. In addition, they permit system-level requirements to be specified and analyzed early in the development process of airborne and ground-based systems. These tools can also be used to perform safety analysis based on the system architecture and initial functional decomposition.
Using AADL-based system architecture modeling and analysis tools as an exemplar, we extend existing analysis methods to support system safety objectives of ARP4754A and ARP4761. This includes extensions to existing modeling languages to better describe failure conditions, interactions, and mitigations, and improvements to compositional reasoning approaches focused on the specific needs of system safety analysis. We develop example systems based on the Wheel Braking System in SAE AIR6110 to evaluate the effectiveness and practicality of our approach.

Keywords: Model-based systems engineering, fault analysis, safety engineering

1 Introduction

System safety analysis techniques are well established and are a required activity in the development of commercial aircraft and safety-critical ground systems. However, these techniques are based on informal system descriptions that are separate from the actual system design artifacts, and are highly dependent on the skill and intuition of a safety analyst. The lack of precise models of the system architecture and its failure modes often forces safety analysts to devote significant effort to gathering architectural details about the system behavior from multiple sources and embedding this information in safety artifacts, such as fault trees.

While model-based development (MBD) methods are widely used in the aerospace industry, they are generally disconnected from the safety analysis process itself. Model-based systems engineering (MBSE) methods and tools Add citations: us, Trento folks, UPenn folks now permit system-level requirements to be specified and analyzed early in the development process. These tools can also be used to perform safety analysis based on the system architecture and initial functional decomposition. Design models from which aircraft systems are developed can be integrated into the safety analysis process to help guarantee accurate and consistent results. This integration is especially important as the amount of safety-critical hardware and software in domains such as aerospace, automotive, and medical devices has dramatically increased due to desire for greater autonomy, capability, and connectedness.

Architecture description languages, such as SysML [] and the Architecture Analysis and Design Language (AADL) [] are appropriate for capturing system safety information. There are several tools that currently support reasoning about faults in architecture description languages, such as the AADL error annex [] and HiP-HOPS for EAST-ADL []. However, these approaches primarily use *qualitative* reasoning, in which faults are enumerated and their propagations through system components must be explicitly described. Given many possible faults, these propagation relationships become complex and it is also difficult to describe temporal properties of faults that evolve over time (e.g., leaky valve or slow divergence of sensor values).

In earlier work, Rockwell Collins and the University of Minnesota developed and demonstrated an approach to model-based safety analysis (MBSA)Add citations using the Simulink notation Add Simulink citation. In this approach, a behavioral model of (sometimes simplified) system dynamics was used to reason about the effect of faults. We believe that this approach allows a natural and implicit notion of fault propagation through the changes in pressure, mode, etc. that describe the system's behavior. Unlike qualitative approaches, this approach allows uniform reasoning about system functionality and failure behavior, and can describe complex temporal fault behaviors. On the other hand, Simulink is not an architecture description language, and several system engineering aspects, such as hardware devices and non-functional aspects cannot be easily captured in models.

This paper describes our initial work towards a behavioral approach to MBSA using AADL. Using assume-guarantee compositional reasoning techniques, we hope to support system safety objectives of ARP4754A and ARP4761. To make these capabilities accessible to practicing safety engineers, it is necessary to extend modeling notations to better describe failure conditions, interactions, and mitigations, and provide improvements to compositional reasoning approaches focused on the specific needs of system safety analysis. These extensions involve creating models of fault effects and weaving them into the analysis process. To a large extent, our work has been an adaptation of the work of Joshi et. al to the AADL modeling language.

To benchmark our work, we have developed architectural models for the Wheel Braking System model in SAE AIR6110 to evaluate the effectiveness and practicality of our approach. Starting from a reference AADL model constructed by the SEI instrumented with qualitative safety analysis information [], we add behavioral contracts to the model. In so doing, we determine that there are errors related to (manually con-

structed) propagations across components, and also an architectural decomposition that allows for single points of failure. We use our analyses to find and fix these errors.

The rest of the paper is organized as follows...

2 Safety Assessment Process

COMPLETELY STOLEN FROM THE SAFECOMP-05 PAPER! Either note it or modify it

The overall safety assessment process that is followed in practice in the avionics industry is described in the SAE standard ARP 4761 [**?**]. Our summary in this section is largely adopted from ARP 4761.

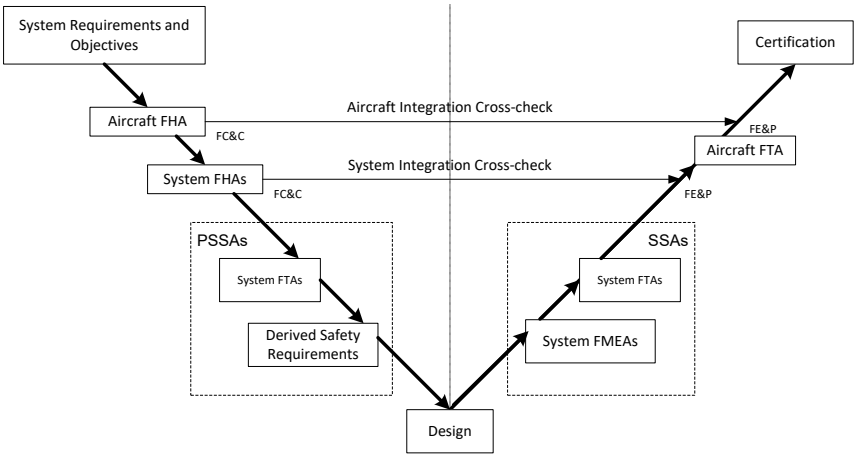

Fig. 1. Traditional "V" Safety Assessment Process

Figure 1 shows an overview of the safety assessment process as recommended in ARP 4761. The process includes safety requirements identification (the left side of the "V" diagram) and verification (the right side of the "V" diagram), that support the aircraft development activities. An aircraft level Functional Hazard Analysis (FHA) is conducted at the beginning of the aircraft development cycle, which is then followed by system level FHA for individual sub-systems. The FHA is followed by Preliminary System Safety Assessment (PSSA), which derives safety requirements for the subsystems, primarily using Fault Tree Analysis (FTA). The PSSA process iterates with the design evolution, with design changes necessitating changes to the derived system requirements (and also to the fault trees) and potential safety problems identified through the PSSA leading to design changes. Once design and implementation are completed, the System Safety Assessment (SSA) process verifies whether the safety requirements are met in the implemented design. The system Failure Modes and Effects Analysis

(FMEA) is performed to compute the actual failure probabilities on the items. The verification is then completed through quantitative and qualitative analysis of the fault trees created for the implemented design, first for the subsystems and then for the integrated aircraft.

We propose to modify this traditional "V" process so that the lower level PSSA and SSA activities are performed based on a formal model of the system under consideration. Figure 2 shows the modified "V" diagram for model-based safety analysis. The shaded blocks are those activities that will be modified or added.

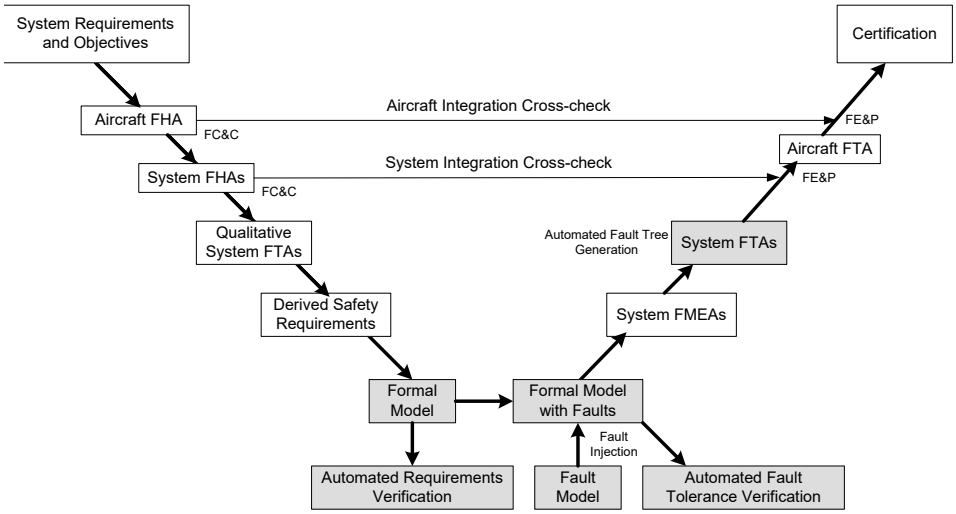

Fig. 2. Modified "V" Safety Assessment Process

As we can observe from Figure 2, the parts of the analysis that are primarily affected are at the bottom of the "V". The biggest difference is that the safety analysis activities at this level are now focused around a formal model of the system behavior, and that many of the artifacts of the safety analysis can be derived from this model. The idea is to try to pose the right verification questions to formal tools (such as model checkers and theorem provers) so that it is possible to derive the necessary safety analysis information. We then wish to turn the results of these analyses back into artifacts that can be easily understood and used by safety engineers.

3 Model-Based Safety Analysis

Audience should already know about model-based safety analysis. We want to discuss two "strands" of MBSA, one with explicit fault propagation of faults rather than system dynamics, and another based on faults propagating through their effects on dynamics.

A model-based approach for safety analysis was first proposed by Joshi et. al in [7–9]. In this approach, a safety analysis system model (SASM) is the central artifact in

the safety analysis process, and traditional safety analysis artifacts, such as fault trees, are automatically generated by tools that analyze the SASM. Figure include figure? provides an overview of this process applied to the wheel braking system example.

The contents and structure of the SASM differ significantly across different conceptions of MBSA. We can draw broad outlines between two different approaches: one in which *faults* are propagated between components explicitly and the analysis proceeds by determining the likelihood of faults to system boundaries, and another in which faults propagate by changing the system dynamics, which may cause the system behavior to visibly change. We call the first style the *explicit fault propagation* and the second style *implicit fault propagation* or *behavioral* propagation.

We contrast these two styles below...

3.1 Explicit Fault Propagation Approaches

What is being done in AltaRica, HiPHOPS, AADL Error Annex...more here about primacy of faults as the mechanism of propagation between components. Steal from proposal.

The explicit propagation approach focuses on faults rather than constructing a model of system dynamics. We illustrate this approach with the AADL error model annex [11] that can be used to describe system behaviors in the presence of faults. This annex has facilities for defining error types which can be used to describe error events that indicate errors in the system. The behavior of system components in the presence of errors is determined by state machines that are attached to system components; these state machines can determine error propagations and error composition for systems created from various subcomponents.

Error types in this framework are a set of enumeration values such as NoData, BadData, LateDelivery, EarlyDelivery, TimingError, and NoService. These errors can be arranged in a hierarchy. For example, LateDelivery and EarlyDelivery are subtypes of TimingError. The errors do not have any information (other than their type) associated with them. AADL includes information on the bindings of logical components (processes, threads, systems) and their communication mechanisms onto physical resources (memories, processors, busses), and the error annex uses this information to describe how physical failures can manifest in logical components.

An example is shown in Figure 2. Errors are described by the red rectangles labeled with error types: 1-BadData, 2-NoData, 3-NoSvc. Error events that can cause a component to fail are labeled with the corresponding error number. The error behavior of components is described by their state machines. Note that while all state machines in Figure 2 have two states, they can be much more complex. The red dashed arrows indicate propagations describing how failures in one component can cause other components to fail. For example, failures in the physical layer propagate to failures in the associated logical components.

Although the error model annex is very capable, it is not closely tied to the behavioral model of components or their requirements. For example, in the wheel braking system (WBS) example [2], it is possible that hydraulic system valves can fail open or fail closed. In fail closed, downstream components receive no flow and upstream pipes may become highly pressurized as a natural consequence of the failure. Physical

models of these behavioral relationships often exist that can propagate failures in terms of the behavioral relationships between components. However, with the AADL error model annex, the propagations must be (re)specified and defined for each component. The user must therefore model the system twice, specifying propagations in the models of the physical phenomena, and again using enumerations and propagation rules in state machines in the error model annex. This re-specification can lead to inconsistencies between physical models and error annex models. In addition, the physical relationships between failures can be complex and may not be describable using enumeration values, leading to additional inconsistencies between the behavior of the physical phenomenon and the behavior of the error model.

3.2 Implicit Fault Propagation Approaches (Behavioral MBSA)

Unfortunately, this is what is done, and done well in xSAP; they have a CAV paper on it; need to account for this.

The analysis starts from a *nominal* model of the system that describes the system behavior when no faults are present. To perform safety analysis, we then also formalize the fault model. The fault model, in addition to common failure modes like *nondeterministic*, *inverted*, *stuck_at* etc, could encode information regarding fault propagation, simultaneous dependent faults and fault hierarchies, etc. After specifying the fault model and composing it with the original system model, the safety analysis involves verifying whether the safety requirements hold in presence of the faults defined in the fault model.

In this work, a behavior fault modeling language was constructed as an extension to the Lustre language [6]. Lustre is the kernel language of the popular model-based development tool SCADE [4]. In this approach, a safety engineer can model different kinds of fault behavior: e.g., stuck-at, ramp-up, ramp-down, and nondeterministic, and then *weave* these fault models into the nominal model. The language for describing faults is extensible, allowing engineers to define a catalog of faults appropriate for their domain. In addition, the weaving process allows error propagation between unconnected components within a system model [8]. This allows consideration of physical aspects (e.g., proximity of components, shared resources such as power) that may not be present in a logical system model but can lead to dependent failures. In addition, it allows propagation of faults in the reverse direction of the model data flow. This can occur when physical components have coupling such as back-pressure in fluid systems or power surges in the opposite direction of communication through connected components. Finally, it is possible to create fault mediations to describe the output in the presence of multiple simultaneous faults.

A safety analysis system model can be used for a variety of simulations and analyses as shown in Figure include figure?. Modeling allows trivial exploration of *what-if* scenarios involving combinations of faults through simulations. For more rigorous analyses, static analysis tools, such as model checkers and theorem provers, can be used to automatically prove (or disprove) whether the system meets specific safety requirements. The primary approach used for analysis in previous work was model checking. After creating the system model, a model checker was used to verify that safety properties hold on the nominal system, an idealized model of the digital controller and the

mechanical system containing no faults. Once the nominal model is shown to satisfy the safety property, the behavior of the fault-extended model can be examined. In the approach described by Joshi et al. in [7–9], this analysis was performed by determining whether the property held for a given fault threshold: the maximum number of component faults to which the system is expected to be resilient.

This fault threshold is, in some sense, an approximation of the likelihood of component faults. It maps from the probabilistic *real world* potential for component failure into a non-probabilistic verification problem. Recent work [3] uses a more sophisticated approach involving minimum cut sets to describe the set of potential component failures that must be considered. Both approaches provide separation between the probabilistic aspects of the real world and the computational demands of formal analysis. This approach currently scales far better than direct use of probabilistic model checking tools such as PRISM [10], and will likely continue to do so in the future.

4 New MBSA Capabilities

REWRITE!

Previous research has been based on MBD tools (such as Simulink and SCADE) that were available at the time [9]. However, these tools are really targeted at the design and implementation of software components, rather than at the system architecture level where most safety concerns arise.

Within the past five years there have been great advances in the capabilities of tools for modeling and analysis of at the system level, based on languages such as SysML [5] and AADL [1]. We use these new MBSE capabilities and extend them to implement the safety analysis methods needed for the design and certification of commercial aircraft systems. The system modeling tools that we plan to use are based on AADL, but they can import and export models from SysML.

- We will improve the efficiency of MBSA methods by using MBSE tools that can perform compositional reasoning over complex system models. Using the assume-guarantee contract mechanism, these tools provide support for heterogeneous component models implemented in different languages (such as Simulink or C/C++).
- Our new analysis methods move away from traditional static safety analysis methods focused on probabilistic models (e.g., Fault Tree Analysis), to the direct modeling of potential failure mechanisms and the analysis of dynamic fault-mitigation strategies.
- Formal verification of system models provides increased assurance that these models are accurate and will produce correct results. The hierarchical structure of system architecture models supports analysis at varying levels of abstraction. Compositional analysis explicitly checks assumptions captured in component and subsystem contracts. Consistency and realizability checks [23] provide the ability to detect conflicting requirements between component and subsystem models.

We bridge the descriptions of errors in the error model annex with behavioral descriptions of components. We start from the error model notions of error types and state machines that describe transitions from nominal to error states. However, we then tie

these nominal and error states to behavioral models of the components in question that describe how the faults manifest themselves in terms of the signals or quantities produced by the components. Now the behavioral models can provide implicit propagation of the faulty behaviors and the natural consequences of failures on component behavior will be manifested in the propagation of other component faults through the behavioral model.

To accomplish this, we use AADL and the error model annex to describe faults, and to use the AGREE contract specification language to describe behavioral models. This requires extensions to AGREE to define fault models that describe how different faults manifest themselves in changes to output signals. It also requires changes to the error annex. The conditions under which faults occur will become richer such that they describe not just propagation of enumerations from other components, but also valuations of input signals.

5 ARP4761: Wheel Brake System

As a preliminary case study, we utilized the Wheel Brake System described in ARP 4761 - Appendix L [2]. The informal wheel brake system diagram is shown in Figure 3 and is taken from the ARP 4761 document. The Wheel Brake System is installed on the two main landing gears and is used during taxi, landing, and rejected take off. Braking is either commanded manually using brake pedals or automatically with no need for the pedals (autobrake). When the wheels have traction, the autobrake function will provide a constant smooth deceleration.

Each wheel has a brake assembly that is operated by two sets of hydraulic pistons which operate independently. One set is operated by the Green hydraulic line which is used in NORMAL braking mode. The second system (Alternate) is operated by the Blue hydraulic line which is used when the Normal system fails. The Alternate system is also supplied by an Accumulator which is a device with built up pressure that can be released if both of the two primary pumps (Blue and Green) fail. The accumulator supplies the Alternate system in case of EMERGENCY braking mode.

Switching between the hydraulic pistons and sources can be done automatically or manually. If the normal pressure (Green) is below a certain threshold, there is an automatic switchover to the Blue supply. If the Blue pump fails, then the Accumulator is used for hydraulic pressure.

In both NORMAL and ALTERNATE modes, an anti-skid capability is available. In the Normal mode, the brake pedal position is electronically fed to a braking computer called the Braking System Control Unit (BSCU). The BSCU monitors signals that denote critical aircraft and system states to provide correct braking function, detect anomalies, broadcast warnings, and sent maintenance information to other systems.

5.1 Nominal System Modeling

A formal specification of the nominal system model consists of mechanical and digital components and their interconnections.

The highest level component is the Wheel Braking System (WBS). It consists of a digital control unit, the BSCU, and Normal and Alternate hydraulic pressure lines (supplied by Green Pump and Blue Pump/Accumulator respectively). The system takes inputs from the environment including PedalPos1, AutoBrake, DecRate, AC_Speed, and Skid. All of these inputs are forwarded to the BSCU to compute the brake commands. The outputs of the WBS are Normal_Pressure, Alternate_Pressure, and System_Mode (NORMAL, ALTERNATE, EMERGENCY).

5.2 Braking System Control Unit (BSCU)

The BSCU is the digital component in the system that receives inputs from the WBS. It also receives feedback from the Normal and Alternate lines and two power inputs from two separate power sources.

References

1. AADL. Predictable model-based engineering. http://www.aadl.info.
2. AIR 6110. Contiguous aircraft/system development process example, Dec. 2011.
3. M. Bozzano, A. Cimatti, A. Griggio, and C. Mattarei. Efficient anytime techniques for model-based safety analysis. In *Computer Aided Verification (CAV '11*, 2015.
4. Esterel Technologies. Scade suite product description. http://www.estereltechnologies.com.
5. S. Friedenthal, A. Moore, and R. Steiner. *A practical Guide to SysML.* Morgan Kaufman Pub, 2008.
6. N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The Synchronous Dataflow Programming Language Lustre. In *In Proceedings of the IEEE*, volume 79(9), pages 1305–1320, 1991.
7. A. Joshi and M. P. Heimdahl. Model-Based Safety Analysis of Simulink Models Using SCADE Design Verifier. In *SAFECOMP*, volume 3688 of *LNCS*, page 122, 2005.
8. A. Joshi and M. P. Heimdahl. Behavioral Fault Modeling for Model-based Safety Analysis. In *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE)*, 2007.
9. A. Joshi, S. P. Miller, M. Whalen, and M. P. Heimdahl. A Proposal for Model-Based Safety Analysis. In *In Proceedings of 24th Digital Avionics Systems Conference (Awarded Best Paper of Track)*, 2005.
10. M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *In Proceedings of the 23rd International Conference on Computer Aided Verification (CAV '11)*, volume 6806 of LNCS, 2011.
11. SAE AS 5506B-3. Aadl annex volume 1, Sept. 2015.

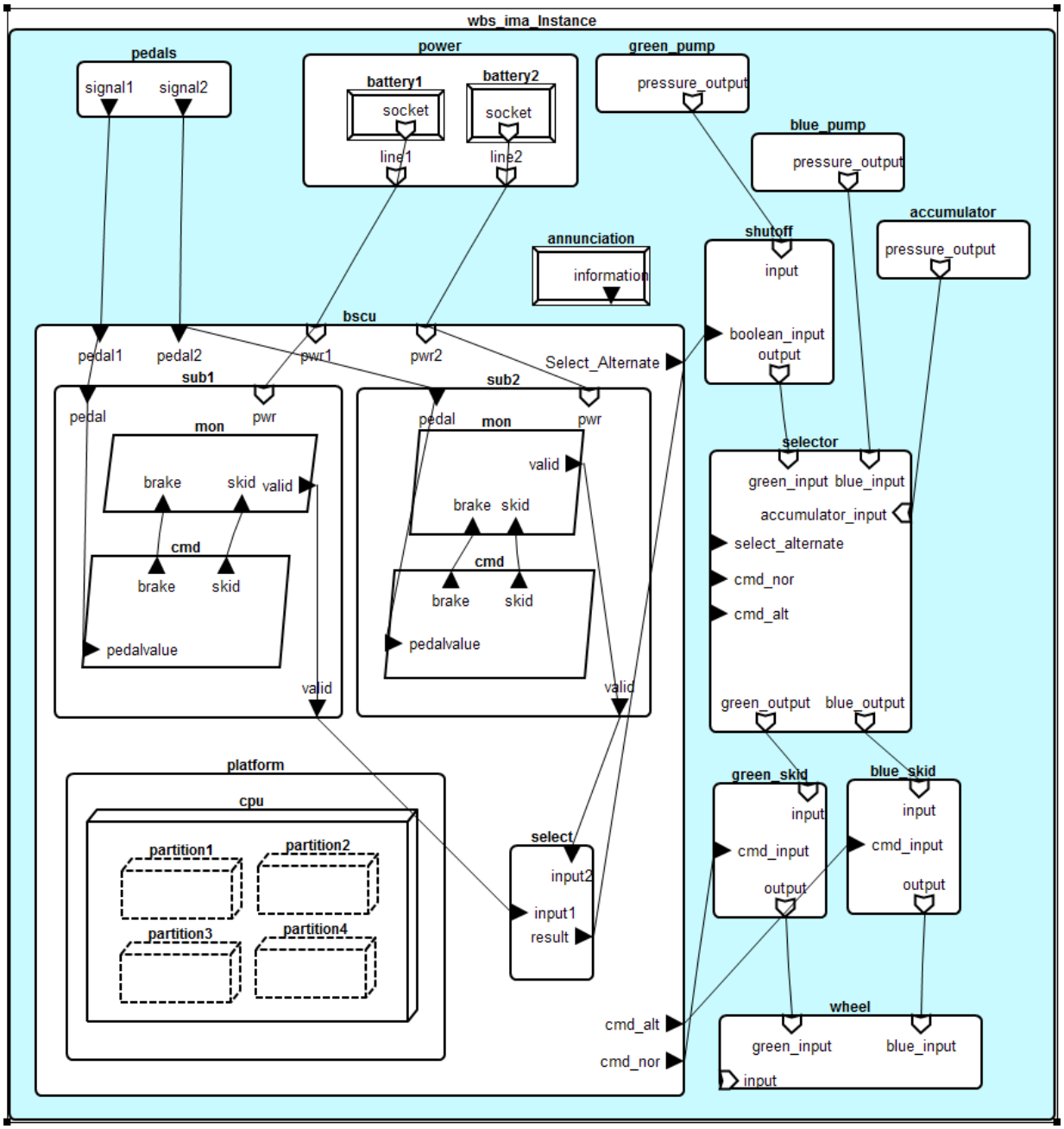

Fig. 3. WBS model diagram from Simple Version ARP4761