UTRC CODA Code Summary

UTRC Technical Point of Contact:
Brian Murray
Phone: 860.610.7696
Email: MurrayBT@utrc.utc.com

Design Flow Models & Tools

The design flow code includes two main artifacts: models and modeling tools. Models for components and topologies of typical electric power systems for aerospace applications are written in Matlab/Simulink. Models for the thermal management system are also written in Matlab. Models for a UAV aircraft engine are implemented in SysML. A System Dynamics Model for predicting Non-recurring Engineering (NRE) cost and schedule as a function of different META design process parameters is provided in Vensim. An extensive library of models have been developed in the PACELAB tool and include steady state models of all the sub-systems of a small UAV as well as mission models. All the models are considered non-ITAR.

UTRC CODA implements Platform-Based Design (PBD). Platform modeling tools are implemented as an Eclipse plug-in. The plug-in provides an editor to write models in the TinyCSL platform design language developed by UTRC. In addition, the plug-in is able to parse the model and perform verification.

In addition, in order to animate and demonstrate the design flow, the UTRC team developed a tool integration architecture and prototype implementation. The prototype framework employs a mini-SOA architecture to include different tools/modules. These modules were developed independently in different environments. The distribution includes the files necessary to demonstrate the design of an engine for the UAV case study. The design starts from a formal definition of a library of devices and design rules, from which the architecture enumeration tool generates all feasible engine architectures. These results were further refined by an optimization module that minimizes fuel consumption. The optimized results are stored in a repository and can be analyzed by design metrics tools, such as adaptability analysis. The results to carry forward to the more detailed design can be selected based upon analysis tools or handpicked. The META Vensim Design Flow Model allows prediction of speedup factor, NRE, number of design iterations, schedule and number of required design personnel. Outcomes can depend on META parameters such as the number of levels of abstractions, model library coverage and integrity and the ability to catch emergent behaviors during early design cycles.

Manufacturing Modeling

Engine Cost model:

Jet engine cost models at two levels of abstraction (Level 0 and Level 1) are included as an example for how to include manufacturing modeling in PBD. For Level 0, a parametric cost model was developed using historic data including development cost. The cost estimation relationship is a linear regression of historical cost data with engine weight, thrust, and fuel burn. The preliminary Level 1 cost model is based on first principal physics (thermal dynamic parameters). The cost is estimated considering component size, temperature and material strength requirements derived from thermal dynamic parameters. The constants in those equations are calibrated by available public domain engine data. The cost models are integrated in the engine design flow demo.

Feature-based Cost model:

A feature-based cost model was developed for estimating the cost of an integrally bladed rotor (IBR) using commercial DFM cost estimation software and is included as an example of the use of such models

in PBD. The design of the IBR is loaded from a CAD file. The first step is to specify the bulk material property and the manufacturing process parameters for the bulk material manufacturing such as powder metallurgy or casting. Machining operations to follow include multiple roughing and finishing steps. The machine equipment, machining process parameters and related costs such as material unit cost, labor rate, as well as machining time need to be specified. The equations to calculate these costs are coded in a language similar to C.

Design Space Exploration Tools

Architecture Enumeration and Evaluation (AEE) is a method for rapidly searching a combinatorically large design space to find feasible architectures (an element of Design Space Exploration: DSE). Rather than constructing every complete system architecture in the design space, and then using "evaluative" rules to check each architecture for feasibility, "generative rules" are used to build and test partial architectures progressively, checking them as each new technology option is added. The AEE tool applies the AEE algorithm for a single abstraction layer and can be called repeatedly with different input data at different abstraction layers.

In the CODA methodology, AEE is followed by optimization. Four different types of optimization were explored in CODA: off-the-shelf optimizers Matlab and AMPL, Diversity in the Objective Space, Maximum Diversity in Design Variable Space and Grey Box Modeling.

Diversity in the Objective Space (DMA) is a method for generating the efficient frontier for multiobjective problems. This approach is capable of solving mixed-integer and combinatorial problems. DMA finds Pareto optimal solutions by maximizing a proposed diversity measure and guarantees generating the complete set of efficient frontier points.

Maximum Diversity in Design Variable Space considers the problem of computing maximum diversity of the design variables in Multi-Objective optimization.

Grey-Box Modeling: Many researchers have addressed the optimization challenges for system models where explicit mathematical models are not available or accessible (e.g. CFD code). Such models are called black-box models. However, many times, mathematical equations are indeed available for a subset of the system, such as component connectivity information. Such models *Gray-box* models in CODA. A seamless optimization approach has been developed that can handle mixed types of systems models – White, Gray, and Black box. It is demonstrated that formulating gray-box structures that maintains dependency (new approach) is faster than treating an entire model as a black-box function (conventional approach). Code to wrap convert black box models to grey box is included.

The Emitter-Propagator-Absorber (EPA) code computes the EPA ratio for an architecture without resorting to an exhaustive quasi Monte Carlo (qMC) simulation of the whole architecture. The EPA ratios for individual subcomponents are computed using qMC. Depending upon the connectivity of the subcomponents in the architecture, the EPA ratio of the overall architecture is computed in a fraction of the time it would take to do the same computation using qMC. This is achieved using EPA composition rules for series and parallel subcomponent connectivity.

The clustering tool computes a complexity formulation for a directed graph and uses this to partition the graph into clusters that can be shown to have lower complexity than alternative partitions. This technique is intended to support the design of lower complexity design hierarchies.

Metrics

The distribution includes a toolbox for computing various complexity metrics related to large-scale engineered systems. The main functionalities of this toolbox are:

1. Structural Complexity - The structural complexity metric is computed based on the system architecture (represented by the binary DSM or the adjacency matrix). This metric expresses the

complexity inherent in the physical architecture of the system and is based on an analogy with quantum chemistry of complex molecules, using graph energy as a central concept.

2. Dynamic Complexity – multiplicative metric based on (a) connectivity of functional responses of the system and (b) their associated uncertainty which is quantified using a Shannon-entropy derived metric. It can be computed for Black-Box models where explicit mathematical models are unavailable.

3. Cycle Complexity - based on the underlying physical architecture of the system and captures complexity arising from cycles alone in the physical architecture and also computes a clustering of components into different layers (i.e., a layered decomposition of the architecture).

In addition, code to compute an adaptability metric is included in the engine demo - The adaptability and cost analysis tool produces adaptability metric and development cost estimations according to a definition of adaptability that indicates to what extent an architecture can adapt to different missions, based on the supported missions and associated switching costs. Switching costs are an aggregate measure that capture the component-level and interface-level changes that are required to transition from one engine architecture and operating mode to another.