Faster Exploration of High Level Design Alternatives using UML for Better Partitions

Waseem Ahmed
Curtin University of Technology
Sarawak Campus, Malaysia
waseem@curtin.edu.my

Doug Myers
Curtin University of Technology
Bentley Campus, Western Australia
myersd@mail.ece.curtin.edu.au

Abstract
Partitioning is a time consuming and computationally complex optimization problem in the codesign of hardware software systems. The stringent time-to-market requirements have resulted in truncating this step resulting in sub-optimal solutions being offered to consumers.

To obtain the true global minima, which translates to finding the best solution available, a new methodology is needed that can achieve this goal in a minimal time. An approach is presented that forms a basis for design space exploration from a partitioning perspective using UML 2.0.

1. Introduction
Partitioning algorithms operate at code level and have lengthy execution times. An added step of exploration of design alternatives in the development process, if not for an extremely good reason, will be seen as unnecessary and detrimental to designer productivity. Evaluation of design alternatives at the code level is undesired for the following two reasons. One, each design alternative will have to be translated to code before subjecting it to the partitioning algorithm, although only one will eventually be used. Two, partitioning at the code level is time consuming.

A better solution would be to evaluate design alternatives at a more abstract level. UML, a rapidly evolving standard traditionally used in the design of pure software, is slowly gaining a following in the real time and embedded system design world as well. Significant improvements, in the scale of 30% to 80%, have been reported in time-to-market, quality and reusability of corporate Intellectual Property(IP) with proper and intelligent application of UML in system design [1].

An approach for achieving faster design space exploration from a partitioning perspective is presented in the following sections.
previously rejected design from a set of design alternatives, may yield a more optimal solution to the cost function than obtained from chosen design if subjected to the same partitioning algorithms. Thus, the global minima obtained from a design cannot be claimed as the true global minima unless all the other design alternatives have been explored as shown in Figure 1.

Thus, in order to obtain better solutions it is very important that exploration of design alternatives in systems with codesign, be done from a partitioning standpoint.

3. Deriving appropriate estimation metrics

Closeness metrics [2] have been successfully used to evaluate partitions at the code level. Design evaluation at a higher abstract level has been also been done [3] but the basis of the comparison have been the program length (in KLOC) and man hours needed. These metrics although very valuable from a managerial perspective, do not serve as useful indicators from the design's partitioning point of view. Attempting to derive meaningful metrics indicative of a design's partitioning viability, at a higher abstraction level like UML, would be a much desired objective.

System requirements, translated into different UML diagram sets, are used as input to the design exploration process.

3.1 Metrics from UML diagrams

Structural and behavioural descriptions of the system are modelled differently in UML. The class and object diagrams give the structural representation of the design at a low level. The parameters for the cost function are obtained in the following steps:

1. The number of states in the system can be identified from the UML state chart. For each of these states, their duration can be obtained from the UML timing diagrams, which are a new feature in UML 2.0. These depict the different states, the state transition order and the state duration. Besides the state duration, the frequency of occurrence of a state is equally important as embedded systems are primarily event-driven reactive systems. This information is domain and system dependent and has to be obtained from the domain expert. Let's denote this information for each state as a tuple \( S_a = \{ S_a, S_b \} \)

2. Every state involves object activity. From UML sequence diagrams, all objects that help realise a state or multiple states, are associated with that state(s).

3. From the collection of sequence diagrams, object coupling and cohesion are obtained such that

\[
\begin{align*}
O_{cop} &= \sum O_a \rightarrow O_b, \forall a \neq b \\
O_{coh} &= \sum O_a \rightarrow O_a
\end{align*}
\]

where, \( O_a \rightarrow O_b \) represents a function call from an object \( x \) to an object \( y \). Object coupling and cohesion in a design reflects the inter-module communication that exists in the design and translates to the interconnect required for the design and the intermodule communication cost, depending on the part's final implementation media. Object properties have also been used in forming an IDM [4] before partitioning, albeit the purpose there has been primarily for design improvement.

3.2 Forming a cost function

The metrics obtained from the UML tools are combined together with the above obtained metrics to form a cost function, \( CF \) as given below

\[
CF = \alpha \left( f(O_a, S_c), f(O_b, S_c) = g(O_{cop}, O_{coh}) \right) \forall O_a \text{ in } S_c
\]

where \( \alpha \) is a domain dependent constant, \( f(O_a, S_c) \) is a function describing the objects properties in the design and \( S_c \) is a set of states on the domain dependent critical state transition path.

4. References