# Design Refinement for Efficient Clustering of Objects in Embedded Systems

Waseem Ahmed[1], Doug Myers[2]
[1]Curtin University of Technology, Sarawak Campus, Malaysia
[2]Curtin University of Technology, Bentley Campus, Western Australia

## Abstract

*Hardware software co-design seeks to meet performance objectives via a combination of hardware and software modules. One difficulty in reaching these objectives lies in lack of cohesion and increased coupling amongst the implemented modules that results in an increased inter module communication cost. While most of the traditional partitioning approaches are initiated in the post-coding phase, we suggest the design stage may be a better focus of attention in addressing this problem.*

*In this paper, we propose a novel approach that uses information from sequence diagrams in UML designs to help ease the partitioning problem.*

## 1. Introduction

A key phase in the design of an embedded system is hardware/software partitioning that refers to the partitioning of the application into separate hardware and software modules. Traditional approaches to this problem as highlighted in [1][2] have been to initiate the process after the system specifications have been translated into code. The input to such partitioning approaches is thus the source code of the application, a binary implementation, or an internal format generated from the source code during analysis as seen in Figure 1.

An exception to the above is a work based on UML design specification [3] that uses function point analysis and COCOMO to compare different design alternatives at an early stage of analysis.

A major assumption in most of these approaches is that the source code reflects the best possible design, which may not be always true, as the designer of the code might not have taken into consideration the mixed nature of the final implementation. The limitations of the design in terms of mixed implementation are thus carried unchanged into the implementation phase.

In this paper we propose to analyze the design of an application with a mixed hardware software implementation, prior to subjecting it to the partitioning process.

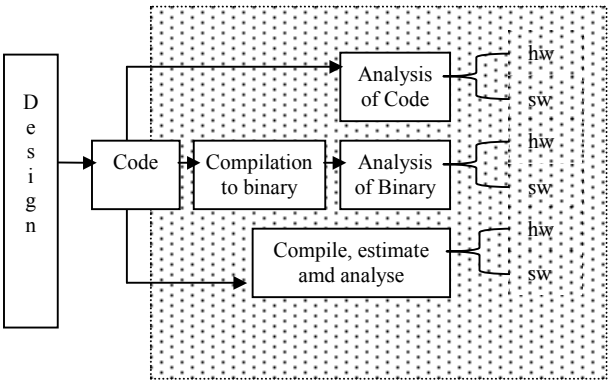


**Figure 1. Traditional Partitioning Approaches**

## 2. Cohesion, Coupling and Design Shuffle

An effective hardware software co-design is one that maximizes *cohesion (*the degree to which communication takes place among the module's elements) while minimizing *coupling* (the degree of inter-modular communication) amongst modules [4]. Modules that have been designed by not taking into account the mixed nature of implementation may have high coupling, resulting in a high inter module communication cost (IMCC), that cannot be wholly rectified by the current partitioning approaches.

Reducing coupling between components may involve either minimizing the interaction between them by shifting the onus of communication to another component or by shifting the entire function (and/or any interaction between them) to another object(s) if possible. We choose to refer to this heuristic as the *design shuffle* or just *shuffle* in this document.

## 3. Sequence diagram analysis

Sequence diagrams in UML are used for depicting the scenarios of typical interactions and message passing between objects that constitute the system. For a single

1530-1591/05 $20.00 © 2005 IEEE

use case there might be many such sequence diagrams.

Information from such sequence diagrams is used to form an inter object dependency matrix (IDM). The entry *f(x,y)* in the IDM gives the sum of all calls and/or instances of message passing between the two objects x and y such that $f(x,y) = (f(x \rightarrow y) + f(y \rightarrow x))$. The entries in the matrix will thus be symmetrical across the diagonal, i.e. $f(x,y) = f(y,x)$. By knowing the type of message passing the interconnect use can be minimized. For example, if the message passing between the objects is distinctly half-duplex, the corresponding IDM entry will be $max(f(x \rightarrow y), f(y \rightarrow x))$. Memory, data sources and data repositories are each depicted as symbolic objects on these sequence diagrams. Once the matrix is prepared, the analysis is a three step process

*I. Individual object analysis*

The entries for each object in the IDM are analyzed together. Based on the values of these entries the object may exhibit one of the three distributions based on the variance from the mean - low, medium or high variance. Objects that show low variance have uniform coupling with multiple objects and if implemented as is, will result in high IMCC. These objects are candidates for a *shuffle*.

Objects that have a high variance show strong coupling with a few objects and low coupling with other objects. Efficient partitioning in the implementation stage will mean that the objects with strong coupling will need to be implemented on the same architecture to minimize IMCC.

Objects with a medium variance lie in between the two extremes. The object pair that have coupling below the mean will have to be *shuffled* such that the coupling between them is further lowered, and the object pair that have coupling greater than the mean are shuffled for a higher coupling.

*II. Object grouping into modules*

After the initial shuffle, the IDM is updated and objects *x* and *y* are identified that satisfy the criteria $f(x,y) > (\mu + \sigma)$, where $\mu$ is the mean and $\sigma$ is the standard deviation. Such objects exhibit a relatively high coupling and are paired together. Let $m_1$ be the modules formed after the first grouping.

The steps are repeated for objects that satisfy the criteria $\mu < f(x,y) \leq (\mu + \sigma)$. Objects where one of its pair has already been grouped is put in the same module, and objects where neither of the pair has been grouped earlier are paired into separate new modules which we denote as $m_2$. Steps similar to the second grouping are repeated again for objects that satisfy the criteria $\sigma < f(x,y) \leq \mu$ and $f(x,y) \leq \sigma$ to create new groupings of modules $m_3$ and $m_4$ respectively.

The groups of modules in $m_1$ have high cohesion within them and need to be implemented on the same architecture to minimize the IMCC. The groups of modules in $m_4$ are basically stand alone modules with very little IMC with other objects. These objects may be eliminated by a good *shuffle*. If not eliminated, the objects in these modules can be subjected or one of the many existing partitioning algorithms for a mixed implementation. The objects in the modules of $m_2$ and $m_3$ are *reshuffled* and step II is repeated until no more *shuffles* are possible. IDM is again updated to reflect the changes.

At the end of this step we have modules that have a high degree of cohesion between them.

*III. Reduction of inter module coupling*

In this step we look to minimize the coupling between modules. All objects x and y where $f(x,y) > \mu$, and *x* and *y* do not belong to the same module, are subjected to a *shuffle* to reduce coupling among modules. This is repeated until no more *shuffling* is possible.

The modules with the redesigned objects are then subjected to one of the many existing coarse grained, hardware or software oriented approaches and considering the available hardware and software resources to get the most efficient implementation for the system.

## 4. Concluding Remarks

Most of the approaches over the past decade in solving the partitioning problem originate in the post-design phase and are based on the premise that the code has been designed with the mixed nature of implementation in mind. In this paper we proposed to initiate the partitioning process in the design phase by a thorough analysis of the interactions between the components for possible redesign.

## References


[1] J. Henkel, *An Approach to Automated Hardware/Software Partitioning using a Flexible Granularity that is Driven by High-level Estimation*, IEEE Transactions on VLSI, April 2002

[2] G. Stitt and F.Vahid, *Hardware/Software Partitioning of Software Binaries*, IEEE/ACM International Conference on Computer Aided Design (ICCAD 2002), Nov. 2002

[3] W.Fornaciari, P.Micheli, F. Salice, L.Zampella, *A First Step Towards Hw/Sw Partitioning of UML Specifications,* Proceedings of the Design, Automation and Test in Europe (DATE'03), 2003

[4] E. Braude, *Software Engineering- An Object Oriented Perspective*, John Wiley and sons, 1st Edition, 2001