Motivation for Providing an Entity Description Language in HASE.

Lawrence Williams
Department of Computer Science
University of Edinburgh

April 1996.

1 Introduction

This document aims to follow up the author's initial PhD project proposal [1] for exploring (semi)automatic model abstraction techniques within the HASE system. This document is primarily concerned with highlighting the current requirement for an entity description language (EDL) within the HASE environment and outlining a plan for the provision of such a language.

2 Short Term Strategy

As mentioned in [1] the first stage of development will be to:

> "Verify that simulations written (i.e. hand crafted) and run at two different abstract levels can produce the same results at the higher level as would be obtained by running the full simulation at the lower level."

3 Immediate Work

In order to start this development process a number of concerns regarding the current HASE system need to be addressed.

3.1 Current Simulation Development Paths

The development paths employed by HASE users when creating a new architectural simulation model currently fall into one of two categories[1] – either a high

[1] These two approaches are discussed in some detail in [2].

level GUI based design technique or a low-level C++ input to the HASE system.

Each of these approaches offers advantages and disadvantages with respect to the other. These can be summarised as follows:

1. **GUI based Approach:** By using the on-screen editor provided within HASE the user can manipulate icons representing simulation entities. Relationships between entities can be identified by connecting on-screen icons together with communication port links. These tasks of design layout are handled well by such a high level interactive interface as it removes the need for tedious trial and error programming when specifying a design's on-screen appearance and allows a design to be conceptualised by considering the components of a 'picture' rather than some detailed code fragment.

   However other tasks are not well served by the GUI approach. For instance when entering information regarding link or state parameters or to be used in a simulation, a labourious process of menu manipulation must be adhered to (e.g. five or six menu commands to complete the addition of a simple integer value into a link parameter's structure).

   Another *major* disadvantage of this GUI based approach is that as the experiment is created as a permanent entity only in terms of an ObjectStore database there is no means of re-creating the experiment should database integrity be breached (something which is a frequent occurrence given the constantly evolving nature of the HASE system).

2. **C++ file based approach:** This technique involves writing a detailed (low-level) C++ file which is linked into the HASE object code via a re-compilation of HASE itself (hardly elegant!).

   However even though this technique requires a detailed understanding of HASE's internal structures in order to describe an experiment it does offer various advantages over the GUI based approach. These include:

   - The ability to re-create a simulation should the experimental database be damaged in any way.
   - Allowing C++ constructs such as loops to be employed when creating multiple instances of entities.
   - Providing a terse, simple and text based input for specifying link, global and state parameters.

   Of course this technique has limitations when compared to the GUI based approach when we consider the ease with which a design layout is specified with the former technique.

It is clear to see that each of the currently employed development techniques have benefits to offer to the HASE design process. It is therefore logical to search for a compromise in which the best features of each approach can be employed. Ideally we would like a development path offering the following:

- Flexible design layout facilities.
- Robustness in the face of database failure (perhaps this questions the need for an experimental database at all).
- Simple entry of non-graphical elements of a design.
- High level construction facilities (e.g. entity creation *loops*)
- A specification not requiring a detailed understanding of HASE's internal data structures.

4 Proposed Solution

It is proposed that an Entity Description Language (EDL) be added to the current HASE architecture. This high level language will allow the user of HASE to describe in a few lines of code the defining attributes of experimental entities (ports, communication, internal data structures etc.).

The move towards a code based method of input to HASE also aims to eliminate much of the labourious work currently involved in adding/modifying experimental globals, parameters and link definitions via the HASE GUI.

It is envisaged that the flexibility currently offered by the GUI based approach in aiding experimental design (e.g. entitiy layout, direct manipulation/examination of experimental objects etc.) will be retained and that layout and on-screen manipulation of objects will not be a concern of the EDL (although obviously an interaction of the two is necassary at somepoint within the HASE architecture).

It is hoped that the EDL will bring together both of the currently adopted design techniques into some form of middle ground as well as offering an ObjectStore independant development path which will in turn mean that HASE can be offered without ObjectStore license restrictions to users at sites outside Edinburgh.

5 Summary of Immediate Action

In the short term it is envisaged that the current HASE release will undergo the following modifications:

- The removal of ObjectStore code dependance from HASE to allow the existance of a licence free version of HASE (Work already in progress by Paul Coe).
- The implementation (via use of `lex` and `yacc`) of an EDL to allow the rapid development of experimental descriptions (Work to be undertaken by Lawrence Williams).
- Conversion of some existing HASE experiements to EDL format descriptions for test purposes.

References

[1] L. Williams. Aiding model abstraction in a hierarchical architecture simulation environment. PhD. Proposal, February 1996.

[2] Lawrence Williams. Simulating dash in hase. Master's thesis, University of Edinburgh, Dept. Computer Science., 1995.