A Framework for Concurrent Simulation Engineering

H. Vangheluwe*, Bo Hu Li†, Y.V. Reddy‡ and G.C. Vansteenkiste*

*University of Gent
Belgium
{hv,ghivst}@biomath.rug.ac.be

† Beijing Institute of Computer Application
and Simulation Technology (BICAST)
China

‡ Concurrent Engineering Research Center (CERC)
West Virginia, USA
rar@cerc.wvu.edu

Abstract

This paper presents a generic framework for Concurrent Simulation Engineering (CSE). Based upon the general principles of Concurrent Engineering (CE), concurrent alternatives to the hitherto mainly linear, multiple feedback Modelling/Simulation/Validation process are proposed. Characteristics, merits and requirements of CE lead to the discussion of the actual framework, accentuating the need for Enabling Technology. The described work is based on the development of a Concurrent Simulation Engineering Environment (CSEE [1]) at the Beijing Institute of Computer Application and Simulation Technology, results of the Dynamic Expert Systems in Robotic Experimentation (DESiRE) project in Gent and experiences with the Darpa Initiative in Concurrent Engineering (DICE) at the Concurrent Engineering Research Center in Morgantown, USA.

1 Introduction

Recently, conception and practice of Concurrent Engineering have gained considerable interest in the (car) manufacturing industry (mainly in the US, thanks to the thrust given by DARPA).

Concurrent Engineering was defined as follows [2]: “Concurrent Engineering is the systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life-cycle from conception through disposal, including quality, cost, schedule, and user requirements.” From the above definition, the application to the development of simulations, rather than to more tangible physical products, is straightforward. In fact, successive simulations can be viewed as stepwise refinements in the development of a product (be it hardware or software). Such an approach interprets a simulation model of a product as a virtual product, on which operations such as quality testing, validation and optimisation can be performed. As such, using CE in all stages of the development process (from virtual to real product) is a powerful tool in meeting total quality requirements as imposed by the ISO 9000 norm [3].

Referring to the above description of Concurrent Engineering, we define Concurrent Simulation Engineering as follows: “Concurrent Simulation Engineering is a systematic approach to the integrated, concurrent design of virtual products, i.e., models of dynamic (time dependent) physical systems, their related processes, including specification, system identification, multifaceted system modelling, virtual experimentation by means of simulation, in/output processing, verification and validation. This approach is intended to cause the developers, from the outset, to consider all aspects of the virtual product life-cycle. In particular, the design of an appropriate process model, eliminating –or at least reducing– feedback loops is vital for improving the refinement process from specification through different abstraction levels to detailed virtual product (to actual product).”

Concurrent Engineering tries to upscale successes of small management “tiger teams”. The essence of such a team (and the seed of its success) is its tight coupling of experts in heterogeneous disciplines, allowing them to collaborate intensely and concurrently towards a common goal. Obviously, the upgrading of this process—in which each team member is replaced by an actual conglomerate of many experts (human and/or software), possibly geographically dispersed, using diverse information representations and technology—is not trivial, and has to be coped with through the deployment of appropriate Enabling Technology, as described later on.

The concurrent working mode transforms the former sequential (or time-delayed), staggered realisation of simulation engi-
neering into an organic, criss-cross and optimised realisation.

To implement simulation engineering successfully, three basic issues are important:

1. An underlying, formalised methodology and supporting techniques must be available.
2. Organisation and management of concurrent simulation engineering must become accepted and supported.
3. An environment (tool set) to support CSE, the “Concurrent Simulation Engineering Environment” (CSEE) must be provided.

In the sequel, a CSEE framework (issue 3) will be developed. In this development, both issues 1 and 2 will be implicitly and explicitly addressed.

2 From Sequential to Concurrent

In this section, the transition from a sequential to a concurrent life-cycle model for the “Simulation Process” is discussed.

Modelling the life-cycle can best be done using a process modelling tool [4]. Thanks to the process interaction nature of the simulation language GPSS [5] and its hierarchical extension HGPSS [6], these languages are particularly suited as a description tool for modelling the simulation life-cycle, as they allow for the visualisation of concurrency. Furthermore, actual simulation of the life-cycle process model can aid in evaluating that process [7]. Remark the meta-level of modelling: a particular modelling/simulation process (that of building a process interaction model of the simulation life-cycle) is used to describe/analyse the general simulation life-cycle. Two types of concurrency can be modelled using GPSS:

- Implicit Concurrency (Figure 1): The processing block implicitly processes (if capacity allows) subsequently generated transactions. Processing one transaction may still be going on while the next one arrives. Implicit synchronisation can occur if competition exists for a guarded resource (e.g., facility).

- Explicit Concurrency (Figure 2): Explicit parallelism is obviously visible in the two parallel processes. Interaction can be made explicit (e.g., by the competition for a shared resource or through a LINK block).

With this simple introduction to process modelling in GPSS, we now describe the transition from a sequential to a concurrent life-cycle model for the Simulation Process.

2.1 Sequential simulation life-cycle

As described in [8], the traditional simulation life-cycle is mainly linear, with multiple feedback loops. In particular, an error detected during the validation stage may imply a complete re-start of the process cycle. Components of the life-cycle are shown in Figure 3:
2.2 Concurrent simulation life-cycle

Concurrency can be introduced into the sequential process:

- Figure 3 contains implicit concurrency if the block diagram is interpreted in a “process interaction” sense. When the GENERATE block generates multiple (un)related simulation problems (that’s the meaning we choose to give to the transactions) in sequence, Figure 3 depicts the concurrent traversal of the sequential simulation life-cycle (and the tools supporting it) by multiple users. As several stages (MOD most notably) interact with a model database, such concurrency can give rise to consistency problems (different versions of the same entity for different users) and deadlock (if the version problem is solved by exclusive access locking). As long as no interaction occurs between different instances of the life-cycle process, this form of concurrency will enhance performance.

- During the modelling process, complex models may be composed from simpler building blocks: sub-models (e.g., with Zeigler’s multifaceted modelling methodology [9]). Hence, to build a particular lumped model, the whole simulation life-cycle may be spawned to construct/validate sub-models as shown in Figure 4 (i.e., sub-model goals, specs, ...). The spawning of the simulation life-cycle from the MOD block –visualised by a feedback loop in the process interaction representation– corresponds to a recursive, forked call to the main life-cycle in a functional representation. Again, concurrency will enhance performance if little or no interference occurs between spawned processes.

- As in Zeigler’s multifaceted modelling methodology, one may wish to investigate different aspects or alternatives of one model. This can again give rise to concurrent instances of the simulation life-cycle (Figure 5). Again, concurrency will enhance performance as most likely, little interference occurs between spawned processes.

- Upto now, concurrency was introduced through multiple invocations of the still mostly sequential life-cycle. Obviously, the most spectacular results will be obtained if the life-cycle components are invoked truly concurrently. Managing the interaction between the components (Figure 6) is not an easy task, which definitely calls for explicit process modelling. We will not venture into possible MAN meta process models, but two general classes can be discerned:
The functional diagram of a proposed Concurrent Simulation Engineering Environment is shown in Figure 7.

Due to our deeply rooted belief in the sequential process model, true concurrency may seem far fetched. Some hurdles may be crossed through suitable use of technology (e.g., supporting a virtual team), thereby shifting the management of concurrency to human rather than automated agents. Changes in modelling methodology (e.g., incorporation of validation tests in the design phase, better model re-use strategies –analogy based model building–, formal specification/verification) will undoubtedly reveal meaningful concurrency.

3 The CSEE

The functional diagram of a proposed Concurrent Simulation Engineering Environment is shown in Figure 7.

3.1 Framework

Due to the two dimensional nature of Figure 7, more information is embedded than is superficially visible:

- The top row of blocks depicts processes. These processes are run concurrently. Each of the blocks represents a generic process. Multiple (interacting) instances may be run concurrently. Typically, these processes will fully exploit hardware/software/human resources of the target they are allocated to. Local resources such as libraries will be used and communication with other processes will occur only when synchronisation or global information is needed. Users communicate with processes through a User Interface (UI). The transition and synchronisation between local and global information/activities is performed by the Local Concurrency Manager (LCM).

- The Communication Services bus depicts the set of services available for information transfer (i.e., product data, multimedia signals, ...).

- The Information Sharing Repository (ISR) borrows the blackboard idea from AI [10]. It provides a global means of communication: different processes can “post” relevant information on the board for others to read. Obviously, coordinating management of the ISR is crucial.

- The Process Coordinator is yet another process. Semantically, it is a meta-process as it contains and manages the process model and hence knows about the other processes.

The described components are generic. Actual mapping of the ideal, virtual concurrent architecture onto a physical concurrent environment is done in an attempt to optimise goal functions such as global machine load, given a set of constraints (e.g., availability).

3.2 Main Characteristics

1. A Framework of Concurrent Operations: As described in the previous section.

2. Information Sharing and Integration: The CSEE realises automatic capture and storage of information in the subprocesses, and information exchange between the subprocesses. Information is divided into local information for an individual sub-process, and global information for several processes. Information management can follow the multi-user sharing distributed pattern or centralised data and file pattern. In each sub-system, the Local Concurrency Manager is used to transmit/receive synchronous or asynchronous information to/from sub-processes, unify the information format (using “wrappers” [11]), coordinate Inter Process Communication and execute security control.

The distinction between local and global (coarse grain) information and processes is not only meaningful for implementation reasons. It also shows abstraction: the global process model need not be aware of local (fine grain) models as long as the local/global link is seamless. As such, low abstraction (from the process point of view, the actual data handled locally may be of highly abstract nature), sub-process specific problems are handled locally.

3. Team Work: The CSEE offers a basic framework for multi-layer, multi-personnel, multi-medium operation. When multiple experts/users are to collaborate (either online, or through common information repositories) on the development of a common virtual product, team work support is crucial. This team work support is realised through:

- A Team Work Interface: All simulation project specification/model/experiment information formats and interfaces should be user-friendly. Multi-media operation should include: symbolic information, text, patterns, lists, images, sound and speech, modelling schemes (object, equation, event, ...).

- Team Work Coordination: There must be a project management system for team personnel intercourse coordination and overall process monitoring/management. Process Modelling is seen as a tool for explicit team coordination (describing, prescribing and proscribing the interaction/operation process).
• Team co-location: To enable and stimulate collaboration, team components (both human and software agents) should be virtually co-located. Physical distances can be camouflaged through Enabling Technology such as integrated video conferencing and tool sharing [12].

4. Standardisation and Open Systems: There must be a unified model/experiment program/protocol and a standard regulation for information processing. An early attempt to such standardisation was made in 1967 by the Simulation Council. This resulted in the currently still used CSSL standard [13]. Without such a standardisation, automatic integration of information among the sub-processes and team-members would be impossible.

Concurrent Simulation Engineering is based upon a computer systems environment. Open Systems have been influential in the explosive growth of standardised, multi-platform computer environments in the 90’s. An Open Environment is a system realised in confirmation with the format and regulations of open interface services and support. The CSEE should also have software (operating system, languages, network, database and user interface) and hardware in confirmation with the open system regulation, thus enhancing portability and workability among different simulation systems.

4 Conclusions

The conceptual framework for Concurrent Simulation Engineering is deemed of growing importance. To realise CSE, new goals are shown for the organisation, management, methodology, techniques, and supporting environment. If these challenges can be met, however, a considerable improvement in both time to delivery and total quality of the “virtual” and real products is anticipated.

References


