Application of mobile devices within distributed simulation-based decision making

Josef Brozek* and Martin Jakes

Department of Software Technologies,
University of Pardubice, Czech Republic
Email: mail@jobro.cz
Email: jakesmar@gmail.com
*Corresponding author

Abstract: As a consequence of the development of the market with information technology, where users are increasingly inclined towards mobile devices at the expense of conventional stand-alone device; increasing user literacy in the use of smartphones and tablets; and the increasing computing performance of mobile devices; a study has been created that addresses the potential of using mobile devices in a distributed simulation. The study also focuses on the possibility of applying the various technologies and architectures in context of using mobile devices in simulation. This article provides overview information about the case study itself, but it strongly focuses on technologies and paradigms that were identified as highly perspective. The paper also explains fundamental themes so that the readers could also apply the information in their home environment. Part of the work itself is an extensive case study carried out in collaboration with a commercial entity.

Keywords: simulation; tablet; smartphone; mobile device; distributed simulation; heterogenic simulation; high level architecture; HLA; simulation-based decision making; decision making.

Reference to this paper should be made as follows: Brozek, J. and Jakes, M. (2017) 'Application of mobile devices within distributed simulation-based decision making', *Int. J. Simulation and Process Modelling*, Vol. 12, No. 1, pp.16–28.

Biographical notes: Josef Brozek is a postgraduate student at the University of Pardubice. Because he focuses on the topic of different computing complexities in his Masters, his research is now on the border of simulation and system engineering. Under Prof. Antonin Kavicka (University of Pardubice, CZ), he focuses on complex programing techniques and their applications, and due to his internship with Prof. Stephan Bhakti Onggo (LUMS, UK), he focuses on HLA and its application.

Martin Jakes focuses on the implementation in different kinds of segments. He specialised in mobile devices during his Masters studies. Now, during his postgraduate study at the University of Pardubice, he is focused on running simulation on non-standard devices. Since his Masters study, he has cooperated under supervision of Josef Brozek and Prof. Antonin Kavicka.

This paper is a revised and expanded version of a paper entitled 'Using tablets in distributed simulation' presented at the 26th European Modeling and Simulation Symposium, EMSS 2014, Bordeaux, France, 10–12 September 2014.

1 Introduction

Originality of our solution is based on cooperation with chemical company and on used technologies – mobile devices. But there are no publications which focus on similarly themes. Application of mobile devices to this kind of problems is a new topic, which has been researched at University of Pardubice for last three years.

The nearest, with their focus, are publications from Reda et al. (2014), Letizia et al. (2015) and Marina et al. (2013). Application of high level architecture (HLA) to simulators can be found in Letizia et al. (2014) or Bruzzone et al. (2010). Or it is possible to see research in healthcare in Cimler et al. (2014).

1.1 Motivation based on global market

When creating simulation systems of some classes, we need to take into account user preferences – this is especially true for simulation applications of the simulator (trainer) type, or decision-making simulators. This condition is particularly important for simulators, which are earmarked for the end users, who are not experts in the field of simulation. For this type of applications it is assumed that the user efficiently handles the control device which serves only as an input-output device for him.

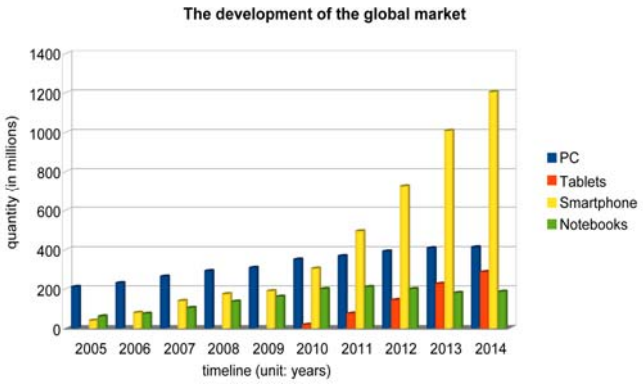
By generalising this principle, we achieve a thesis which says that for a simulation control/operation to be

effective, the user environment must be user friendly as much as possible. At present, however, in the field of user-friendliness, methods of software engineering and software design are ceasing to play the first fiddle. The same applies for the focus on the hardware device that is to serve for operation.

The current development in the information technology market, especially the one focused on the end user, shows that there is a dramatic increase in the sales of tablets at the expense of standard computers and laptops. At the same time, it should be noted that most modern mobile phones sold work on the same principle as tablets. Most of the information in the following text implicitly assumes that when a tablet is used, it can be generalised and applied also to other mobile devices (for example: prototype testing and verification of most of the principles mentioned in the text were, in addition to the tablets, tested at the same time on the mobile phone Samsung Galaxy.).

In general, however, we can draw a conclusion that the end users are increasingly becoming experienced in devices control by the touchscreen. To illustrate the issue, Figure 1 shows the development of the market in each period.

Figure 1 Development of the computer market (see online version for colours)



2 Verification of hardware possibilities

If we want to seriously consider using tablets as primary runtime components for distributed simulation, we need to realise the fundamental differences and define the problems that may arise. For more information you can look to Schön (2013), Kovac (2012), Ku (2012), Mocny (2009) or The Simulation Interoperability Standards Organization (2001).

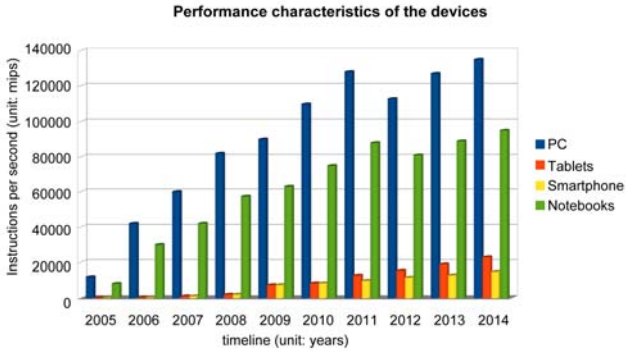
2.1 Performance comparison

The first characteristic that we naturally focus on is the performance aspect of individual devices. It is certainly a legitimate requirement, but it is important to realise that applies only for a certain class of tasks. A tablet, computer or IBM PC is not a device that we choose for simulations for their brute force effectiveness, but for the user friendliness. Therefore, this article takes into account that the class of simulation tasks requiring enormous computing performance will run directly on a cluster, or a cloud.

A bit more specifically to the performance, performance of a standard PC processor is about 80 GFLOPS[1], in contrast to tablets, where the standard performance is about 80 MFLOPS. At first glance, the absolute value says that the performance of the tablet should be approximately 1,000 times lower, compared to PC. At this point, it should be stressed that both architectures use completely different processors. A standard IBM PC uses the x86 architecture processors that may appear to be considerably ineffective [information on the very principle of CISC, internal RISC structure and the resulting information can be obtained from Schön (2013)]. On the contrary, the ARM processors (for more information, see Kovac, 2012) used in tablet PCs have better transfer of computing performance to instructions applicable in programs.

The possibility of an illustrative comparison is presented in Figure 2.

Figure 2 Various theoretical performances (see online version for colours)



The comparison clearly shows that tablets are not appropriate means for the class of tasks requiring brute force (on the other hand, below, the reader will be acquainted with the fact that even this problem is solvable with relatively easy approaches).

2.2 Comparison with regard to peripherals

Although this directly begs comparison of the physical peripherals and input-output devices, motivation for comparing peripherals is a bit different. Therefore, let's move from the intuitive comparison of user input-output devices to the network communication technologies.

A standard PC network connection is carried out through a cable line, which has a (relatively) high level of reliability. In contrast, a standard tablet will be most often connected during the calculation through a wireless connection, which has a high quality only under ideal conditions – otherwise, there is an increase in latency, or loss of parts of the data. Therefore, the creation of models for tablets is more challenging (on the other hand, it is about as challenging as creating models that are synchronised in, for example, the internet, via web services, etc.).

Wireless communication (either Wi-Fi or Bluetooth, at worst, GSM) has its own metrics, which is the portability of the device.

2.3 Physical comparison

Since the comparison is very intuitive, just a few words about a strength that we can advantageously use.

Easy portability of tablets allows using them outdoor in support of ad-hoc decision-making (which is the topic of the practical application presented in this article). The risk is the limited battery operating time, but this problem is gradually eliminated. In addition, certain classes of tasks do not require a long-lasting battery life.

The physical portability implies another major advantage and that is localisation – because if we have an active Wi-Fi or GSM receiver, using certain principles (detailed in Ku, 2012) we are able to perform accurate localisation of the device (this is possible even with GPS, but not in enclosed spaces).

A result of the portability and localisation is, for example, that if we create online simulation of factory operation, the tablet will automatically visualise only the part of the plant near to our current location, i.e., the one where a malfunction is being dealt with, for example. Likewise, these principles can be used to collect data from relevant sensors (as the tablet can also serve as a central point for the collection of values collected with the Bluetooth technology and external sensors that support this technology).

2.4 Practical programmer's perspective

This section is rather intended for software engineers, so it may not be interesting for other professionals engaged in the simulation. It can be skipped without losing the context of the entire thesis of the article.

For the purpose of implementation, we need to take into account several important criteria, such as the demands of development (particularly in terms of the programming language support), the existence of libraries, support, and size of the community engaged in the development.

In order to discuss these issues, it is important to realise that there are (essentially) two main representatives of tablets, namely tablet PC (with the most common operating system being Android, followed by Windows) and the Apple platform – i.e., iPad and iPhone (running on iOS).

2.4.1 Development for Android

First, let us focus on (at least in Europe) the most widespread variant, devices that run on Android. The programming language that is used for programming is, de facto, the programming language Java; we just need to use different libraries. The development itself is relatively trivial. Part of external libraries, written in the Java programming language and designed for standard computers, are also applicable in the solutions for Android.

The community of general programmers is relatively broad and focuses mainly on the graphical environment issue, which cannot be considered a problem as most architectural problems are the same as in the standard Java.

However, a major disadvantage and a great risk in creating applications for Android is the issue of operating system versions. It is because it happens very often that the things that worked with ten different versions in a standard and predictable way will fail to run with an eleventh version. Therefore, with every new update of the operating system we should re-verify the functionality of the solution. The actual verification is performed though verifying the functionality of each module. It does not happen that the application would not work and return invalid results. The need for testing a new version is still a major inconvenience.

2.4.2 Development for iOS

Even for the actual development for iOS it is necessary to have the relevant physical device – emulation is very difficult, practically impossible.

Another major disadvantage is the programming language. It is the Apple company's own language, which is a certain mutation and an extension of the programming language C, it is called Objective-C. For programmers accustomed to standard programming languages (Java, C#, C++), the actual syntax is relatively counterintuitive and we need some time to get accustomed to the language (as well as to the paradigms that the programming follows in that language).

The community is relatively large, but the collective problem solving takes place only to a limited extent. This is probably due to the fact that currently there is a transition to another programming language, so the community is slightly fragmented.

Another major problem is the non-existence of libraries that could be correctly linked to the program that will run on iOS devices, so it would be necessary to create the runtime environment for the simulation virtually from scratch.

2.4.3 Development for Windows

A brief mention should also be given to the development for tablets running on the Microsoft Windows operating system. The development is performed in a standard way that developers are used to, so it will not be discussed in the article, for more information you can go to Mocny (2009).

A moderate risk of this solution is the fact that it is not that easy to disseminate applications or to network owing to the intricate settings and security policies in this operating system. Thus, even though the programming is simpler, we can often encounter various types of runtime problems caused by the settings.

2.5 Recommendation

The computing efficiency and parietal possibilities support are enough for all devices. The primary differences are defined by programming performances.

The results of our work show that it is preferable for the simulation to use devices that run on Android OS or Windows. With the experience, we cannot recommend creating simulators for iOS.

3 Choice of software platform

If hardware is selected, it is necessary to specify optimal software layer – platform, architecture or at least paradigms. During development of the simulator we have tested concrete approaches:

3.1 Java RMI

> "Remote method invocation allows Java developers to invoke object methods, and have execute them on remote Java virtual machines (JVMs). Under RMI, entire objects can be passed and returned as parameters, unlike many remote procedure call-based mechanisms which require parameters to be either primitive data types, or structures composed of primitive data types. That means that any Java object can be passed as a parameter – even new objects whose class has never been encountered before by the remote virtual machine." Reilly (2006)

Although the Java RMI is very interesting technology, it has major drawbacks, in particular:

- it is relatively complicated to use, especially for large distributed simulations
- implementation of heterogeneous simulations is quite difficult (sometimes impossible).

Although it has these weaknesses, there are a few benefits that arise from the Java programming language, or directly from the principle of RMI, which overcome the disadvantages. RMI is possible to recommend for small projects, for experimental purposes and for teaching. But for a larger solution is desirable to use a different technology.

3.2 CORBA

> "Common object request broker architecture (CORBA) is a competing distributed systems technology that offers greater portability than remote method invocation. Unlike RMI, CORBA isn't tied to one language, and as such, can integrate with legacy systems of the past written in older languages, as well as future languages that include support for CORBA. CORBA isn't tied to a single platform (a property shared by RMI), and shows great potential for use in the future. That said, for Java developers, CORBA offers less flexibility, because it doesn't allow executable code to be sent to remote systems.
>
> CORBA services are described by an interface, written in the Interface Definition Language (IDL). IDL mappings to most popular languages are available, and mappings can be written for languages written in the future that require CORBA support. CORBA allows objects to make requests of remote objects (invoking methods), and allows data to be passed between two remote systems. Remote method invocation, on the other hand, allows Java objects to be passed and returned as parameters. This allows new classes to be passed across virtual machines for execution (mobile code). CORBA only allows primitive data types, and structures to be passed – not actual code." Reilly (2006)

Solutions using CORBA is very interesting. With it would be possible to create heterogeneous distributed simulator. The computational performances depend on implementation of the simulators themselves – here are not major losses caused by staging. Its major disadvantage is the complexity of the required of implementation works. In any case, if the HLA testing would failed, CORBA would be good choice for create own prototype. Simultaneously CORBA is good choice in situations where you cannot, or would not like to, use HLA.

3.3 Manual programming

Brief mention is also deserved for experiments with low-level programming through direct memory access (i.e., own bookings and initialisation of ports, the transfer data to binary streams and use communication via streams).

Despite the very low-level data handling the solutions did not show significantly better computing performance. This was most likely caused by programming in Java. The problem has caused overhead of own Java application (JVM, garbage collector, etc.).

This implementation was the most complex of all tested solutions. It is possible say that this approach is not ideal for use in distributed simulation (not only for decision-making, but generally).

3.4 DIS

The possibility of using DIS was rejected already at retrieval activities – so it is the only solution that was not implemented. But for completeness, it is mentioned.

DIS itself is defined as the standard (IEEE 1278). This works at a relatively low level and retains all control of the distributed simulation in the hands of an architect.

Although the DIS can be used for any type of distributed simulation, the original purpose is to join different types of interactive simulators together.

The complexity of standard binding with the method definition and high programming complexity are significant disadvantages. Because at the same time the retrieval operations were conducted with HLA testing, and it was described as very good solution, DIS tests were stopped.

However, for an overview, it is desirable to know the DIS, and the reader may learn more from Manling (2009).

3.5 HLA

One of the tested technologies is HLA. Because it has got best results, we used it in our solution. It is explain more in next section.

4 HLA basis

HLA is designed especially for the development of distributed simulation models. It is a comprehensive architecture. This may discourage users at first glance, but the basic principles are very simple. Because HLA was chosen for its properties as an ideal candidate for our own implementation, below are provided more details about it.

HLA is a standard for the creation and operation of distributed computer simulation systems. The standard does not limit the application domain and simulation modelling method. Hence, HLA has been applied to domains such as military, healthcare, supply chain and many more. HLA has also been used to link models developed using various simulation modelling methods such as discrete-event and agent-based. At a lower level, the standard specifies the method of communication between the simulators (called federates) of a distributed simulation. The standard sets the requirements for the actual transmission format using XML. This enables us to link simulation models written using different programming languages or simulation software. It also enables us to write a wrapper for legacy simulation models (non-HLA compliant) by converting the input-output data into an XML format.

The openness, flexibility and many functionalities of the standard may overwhelm novice users or users who are not familiar with distributed computer programming. One of contributions of this paper is to propose a framework that helps users to develop HLA-compliant models more easily so that they can use the full potential of HLA with minimal effort (at the cost of moderate restrictions which will be discussed later). This research is partly motivated by the lack of adoption of HLA in the Czech Republic. We hope this work may attract more people to develop HLA-compliant models.

The explanation is based on Fujimoto (2000), Kuhl et al. (2000), Rabelo et al. (2013) and standards IEEE1516:2010 (The Institute of Electrical and Electronics Engineers, Inc., 2010a, 2010b, 2010c, 2010d).

4.1 Terminology of the HLA

HLA standard, in order to maintain high level of abstraction, uses some very specific terms. For example, instead of the concept of a logical process, an agent, a decomposed part of the logical process, it uses common term a federate. With federate it is possible to work regardless of its internal structure or architecture.

4.2 Federation

A federation in HLA refers to an entire distributed simulation (or a complex distributed simulation model). There is one federation during one distributed simulation experiment.

4.3 Federate

A federate is an HLA-compliant application that can participate in a distributed simulation experiment. A federate may be in a form of a simulation model, software application, software agent of any type, an input sensor or panel, a display unit, a system for retrieving historical data, etc.

4.4 Objects

An object is the information to be exchanged during a distributed simulation experiment. In HLA, an object represents an entity with persistent states.

4.5 Attributes

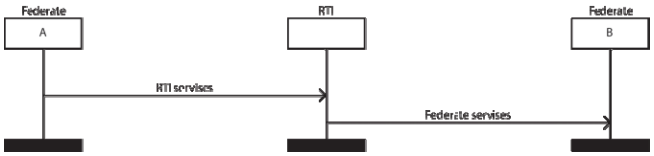
An object has a set of attributes that are relevant to an entity represented by the object. An attribute is basically a data field of a defined data type. Basic data types and attributes are defined in the object model template (OMT). We will explain OMT in Section 4.7.

4.6 Interactions

An interaction represents an event that may be of interest to two or more federates. Similar to objects, interactions must be pre-defined. Each interaction can have a set of parameters. Interaction can be performed directly through interaction classes, e.g., passing an object to another federate, etc.

Federates cannot send interactions to each other directly, but they must make a request to RTI (runtime infrastructure) as shown in Figure 3. We will explain RTI in Section 4.8.

Figure 3 Interactions in HLA



4.7 Object model template

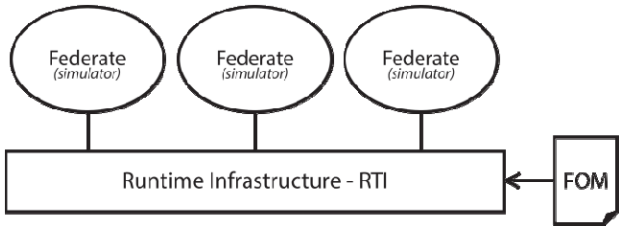
Objects and their attributes as well as interactions and their parameters need to be defined for the whole federation based on the standard set in the OMT (IEEE1516.2:2010). The definitions for objects and interactions must be provided in a federation object model (FOM) or simulation object model (SOM). Both are XML documents and are relatively easy to read. In practice, FOM is often built from one or more SOMs.

4.8 RTI

Run-time infrastructure (RTI) is a middleware that provides communication services to all federates in a federation. It is defined directly by the standard (IEEE1516.3:2010), independent of the platform and language. RTI provides APIs that can be called to perform certain functions such as

creating a federation, creating a federate, connecting a federate to the federation, etc. RTI needs FOM to understand the objects and interactions that will be exchanged between federates. Common terminology and the logic of RTI are shown in Figure 4.

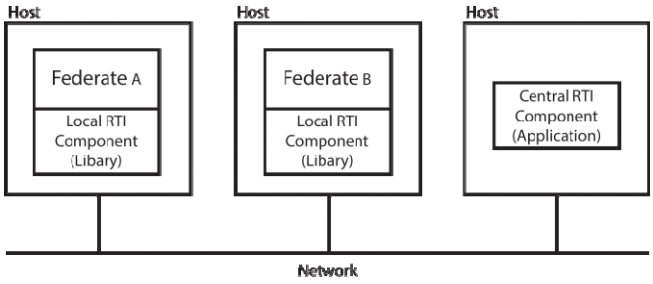
Figure 4 HLA RTI diagram



A number of vendors have implemented RTI products, for example Pitch RTI, MAK RTI, OpenHLA and many more. In principle, the implementation approach is almost the same as the one shown in Figure 5. A federate usually runs on one computing node (but one node can be shared by multiple federates). For each federate, we need to include the local RTI component. It is a library that needs to be linked to the actual software solution (federates) or to the application wrapper (for legacy or non-compliant software). This component actively manages communication between a federate and a central RTI component. The central RTI component is the busiest part in a federation[2]. It is responsible for managing the communication within a federation, for example, connecting a federate to a federation, managing objects, etc.

More information (especially about pRTI) and original pictures may be found at Pitch technologies websites.

Figure 5 RTI physical implementation



4.9 Major benefits of using the HLA

After this introduction to HLA, it is possible to say why this seems like the best solution for creating of distributed simulation in combination with mobile devices.

4.9.1 Platform independent

Because it is possible to create simulators regardless of platform and programming language, and because communication between federates runs through RTI (in XML), HLA is an appropriate solution. Hereby HLA easily allows the connection of tablets (which run programs, e.g., in Java) with other devices (without dependency on a programming language).

4.9.2 Centralisation

The existence of RTI as an active element for the connection of mobile devices is very convenient. It is unsuitable only when the entire solution and all distributed calculations are performed only on tablets – which is extremely unlikely, in the context of a decision-making support system.

Conversely, the combination of mobile devices and other computers (PC, cluster or cloud) provides the perfect environment for running RTI. If RTI is started on the strongest computational component, excellent results are achieved. Actual existence of RTI as separate software (and thus existence of a central component of distributed simulation) does not adversely affect a simulation run; it is not significantly degraded by high overhead.

4.9.3 Other benefits of HLA

Standard advantages resulting from the use of HLA are:

- there is no need to implement synchronisation, or low-level operations – just use the services of HLA
- the entire run is transparent and can be monitored visually on RTI – which greatly facilitates validation and verification
- reusability of individual simulators is very high
- there is a strong community with which it is possible to consult the issue
- nowadays it is possible to use some libraries and frameworks, particularly the simplistic framework for HLA (Brozek et al., 2014)
- the entire distributed simulator can easily meet the standards (IEEE 1516:2010) and its extension to other simulators or by other workers, or on another workplace is easy.

5 Methodologies for using mobile devices

The actual tablet can be used in simulations in several different ways, some of which are very intuitive, while some on the contrary deserve more attention (for example, owing to their considerable development potential).

In the following text are the mobile devices (tablets, smartphones) mainly seen as a small mobile computer. Expected operations relate mainly to the calculations, and the provision of user inputs or provision of outputs visualisation. It is necessary to say, that the potential of tablets is far higher. If we would like to focus on one example: through use of those devices in the simulation we can determine the position – with using GPS or Wi-Fi networks. It is ideal for simulations of movement (an army can use them in simulated combat).

5.1 Used to run the monolithic simulation

Currently, this method of use is purely a marginal issue. The performance is still insufficient and, even with calculations running on graphics accelerators, tablets experience memory issues. Therefore, the simulation thus triggered would be relatively undemanding and it would be rather a trivial demonstration without much practical use.

On the other hand, it can be expected that with the increasing computer performance of tablets and modifications in the architecture of processors (number of kernels, higher parallelisms on graphics accelerators), even this approach will be broadly applied in the future.

5.2 Distributed simulations on tablets

Distributed simulations solely on tablets can be run in two different modes, namely:

5.2.1 Based on the sharing of performance

If we have a certain amount of tablet users who have access to the network and enabled a specific application, it is possible to run a computationally oriented simulation with a part of distributed computing running in the background of each tablet (even more logical processes in one tablet).

This solution is especially useful for calculations with a high degree of parallelism and it is an alternative to the solution in the cloud. The advantage, however, is advanced debugging options of the application.

5.2.2 Standard distributed simulation

In the case of this mode, there is just one logical process running on each tablet, in the foreground of the simulation. Most commonly the method is used in an application of the simulator/trainer type. Each tablet thus serves not only as a computational node for the simulator, but also as an input-output device, which can be used to parameterise the simulation calculation directly during its run.

5.3 Heterogeneous simulation

It is the combined (according to some sources, hybrid) methods of the operation of distributed simulation that currently have the greatest potential for practical use. Specific usable solutions are listed below.

What all solutions have in common is that part of the distributed computation is performed on standard computers, servers, or even in the cloud (it is with the cloud solution that it achieves the best results). The combination of the computing performance of more robust computers with the advantages of tablets (portability, intuitiveness) is also used in the demonstrator presented in this paper.

Only four basic methodologies are listed, which can be combined.

5.3.1 Tablet used as a visualiser

Simple using the tablet as a visualiser is beneficial for those classes of problems where it is desirable to animate the simulation calculation. This has practical applications rather for presentations of the simulation process to a customer and for validation of processes, where a simulation specialist consults realism of the established processes with an expert from the modelled field.

5.3.2 Tablet used as a driver

If we want to use the tablet as a distributed simulation driver, we just need to create such a simulator that will be able to parameterise the conditions prior to the start of the simulation, and then also the conditions for termination of the simulation run.

The solution is useful in situations where it is necessary to parameterise the simulation according to the information that can be obtained only in the field, i.e., beyond the standard stationary device running the simulation.

The ideal is to use this principle in decision support systems, where we can relatively easily parameterise the simulator, without being limited by our physical location.

5.3.3 Tablet used as an output device

The next possibility is to use a tablet as an output device. Since it is not necessary to perform stable synchronisation as in the case of online animation (3.3.1), this solution is relatively trivial. Like the previous approach, this solution is advantageous for decision support systems. Users are often not interested in the simulation, but only in the result.

5.3.4 Tablet used as an input for trainer

The most demanding method of using the tablet occurs when there is a request to implement the distributed interactive simulation (DIS) methods (IEEE1278, more in The Institute of Electrical and Electronics Engineers, Inc., 2010a, 2010b, 2010c, 2010d).

It is especially challenging to secure such transmission problems in the network in order to make sure that all interactive interventions have been performed correctly and at the right time. Synchronisation methodology is equally challenging. It should be noted, however, that despite some complications in the development we can achieve excellent results due to the high user-friendliness of such simulators.

5.4 Tablet used as a preprocessor for data processing

Being a portable computer that has a number of wireless connections (Bluetooth and Wi-Fi), the tablet can be advantageously used as a preprocessor for data processing. It is thus possible to obtain relatively cheaply a complex measuring station. If we have a number of sensors or limit sensors that support the Bluetooth technology, it is possible to retrieve data from these sensors through the tablet and

process it into such a form that is required for distributed simulation. The actual distributed simulation is then connected to this node. Custom node behaviour with respect to distributed simulation purely depends on an architect (software engineer) of the simulation. It is possible to determine whether the mobile device will be part of a distributed simulation; or it will act only as an online sensor for simulation; or it will be completely transparent for simulation).

6 Mobile devices in decision making simulations

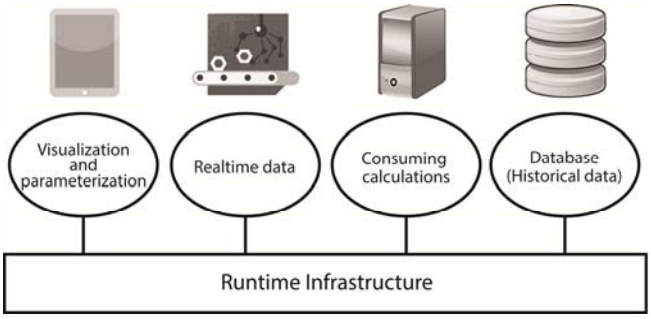
As mentioned above, the actual solution implementation is very suitable for uses of HLA. The following principles will be explained with the assumption that HLA is used. However, all of the principles are possible very easy to generalise and use in systems that do not use HLA.

6.1 Basic scheme and principle of specialisation

Connecting of a mobile device to decision making distributed simulation is the best, if it is realised by using the tablets only as input/output devices.

Pattern, which was proposed in the work, divides problem into four categories, each according to their specialisation. There may exist such decision-making simulation, which will not implement the solution of all four classes.

Figure 6 Architecture of simulation model



Our actual scheme of decision-making simulation is distributed and usually uses four types of nodes (Figure 6):

- the computing node is located on a powerful computer that is designed just for this single activity
- the node for parameterising, defining scenarios, launching the simulation and displaying the results is running on the tablet so that the operator can use any device in any production hall, without having to go back to the service centre once a malfunction has been detected
- the node for data collection is connected to the main automation computer and receives online data from the production
- the node for obtaining historical data is placed in the company's data warehouse and is used to collect historical data on production processes, repairs, etc.

Individual nodes can run in the simulation in a larger number (e.g., two independent nodes for parameterisation running in tablets are a natural part of the solution).

6.2 Computational core

The computing core is responsible for the implementation of all complex calculations. The actual core can be further distributed (split into more logical process/computers). However, it is preferable when it is used heavy-horse, cluster, or when the task runs in the cloud service.

For the SW implementation it is not necessary to follow special programmer's procedures, the standard software engineer's knowledge is enough. When you are choosing a programming language it is desirable to take into account the fact that this part of the solution is strongly computationally oriented.

To increase efficiency of whole solution it is desirable that the same physical machine runs the RTI.

6.3 Database connector

Database connector serves as a portion of a distributed simulation, which is mainly responsible for:

- acquiring historical data from the database at the request of the simulator
- if it is necessary it generates the data (usually on the basis of the any kind of regression from data obtained from the database) at the request of the simulator
- enter data into the database for later processing, or for future needs by self-learning algorithms.

DB connector principle is very simple. If simulation requires data, it can very easily request them. Because the database is accepted as very fast storage (especially if the data must be searched or sorted, etc.), it is the ideal solution for the case with dynamic setting of simulator, which is characteristic of large demands on data.

Although, DB connector principle is very simple, it is more complex during implementations.

The first step is to determine how to realise the entire solution. It is necessary to consider whether the database is available only for the purpose of simulation studies, or whether it is a database that is active and is achievable during normal operation. According to the destination database, must select the correct programmatic approaches. Thus, the maximum use of the database engine and database languages if the database is reserved only for simulation. In the latter case it is necessary to minimise work with databases, in which case it is necessary to implement own robust application. This will collect data from the database, process them, and transfer them into the simulation.

6.4 Real-time data miner

In the event that you would like to implement, in the online decision-making simulation, trips into the future, it is necessary that the whole solution contains a real-time data miner.

The actual form is difficult to describe, because it is very different according to the field to which the simulator is directed. Generally, however, it can be said that most of the system from which data will be required (to identify their real condition in real system), has this monitoring already implemented.

The implementation itself only involves linking the existing system for scanning into a distributed simulator.

Because not all real systems have a centralised collection of data on the computer, on which it would be possible to run a standard application that will work in the simulator, is here greatest variability of practically used programming languages.

6.5 Using of mobile devices

Mobile devices are in a system of this type used as input/output devices. A user can parameterise future simulation and calculation, and on the basis of visualised data gets the results. Implementation of input/output device also would be implemented on standard stand-alone computer, but for some problems it is not as effective as using mobile devices. The advantage of using a mobile device has the mobility rights – there are classes of problems (one of them is described in the case study), for which is the decision 'on the spot' invaluable.

The actual mobile device can use two different ways, namely:

- using in passive mode
- using in interactive mode.

It is true that the possibility of using a mobile device for parameterisation of simulator, its start and its shutdown, is common for both solutions.

The principles mentioned below are therefore usable also for stationary input output devices.

6.5.1 Using in passive mode

Using passive mode, thus only as a 'driver' for the simulation, it is suitable for simpler calculations and simulation processing.

However, the fact is that reading of all the required data (online and historical) and creating of scenario would be very time consuming, hence the second principle is more often used.

6.5.2 Using in interactive mode

Using the interactive mode allows the simulation to run almost continuously. State space of the simulator constantly corresponds to the real state of affairs (or is able to return to this state).

Now will be described technically simplest (though not ideal) solution, which is done within the distributed simulation.

In the event that the simulation is required to predict the future, the current fingerprint is created by simulation. This is followed by split into two independent simulation calculations (every run on a separate thread). The first calculation still illustrates the current real situation. The second calculation may increase speed of the virtual time. The computing core starts reckoning, and mobile device starts to illustrate the expected development of the future. The situation can further affect interactive interventions (by changes parameter of the simulator during its run). Using simulation is so easy to determine how the system state will change after a certain period of time. In the event that instead of one simulation several calculations with different parameters are performed, users are given high-quality data to support its decisions on what actions to perform in a real situation.

The actual method of control and simulation of forecasts in future has many forms and is usually adjusted according to the used hardware. On a cluster or in the cloud it is worthwhile to perform parallel calculations of several threads. On the contrary, a very powerful single-processor computer system is preferable to use for calculations and returns only a single thread.

SW implementation is thus dependent on the hardware used.

7 Demonstration solutions

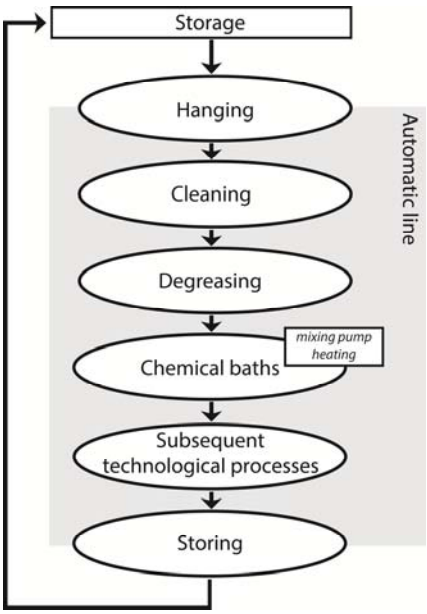
The demonstration example is a great example of how a distributed simulation using tablets can effectively support decision making and thus improve the efficiency of industrial production.

7.1 Description of the problem

The chemical metal coating plant is characterised by several facts. The whole plant is automated (human labour is only at the beginning of the process to place the products to be coated on hooks and then during the storage process).

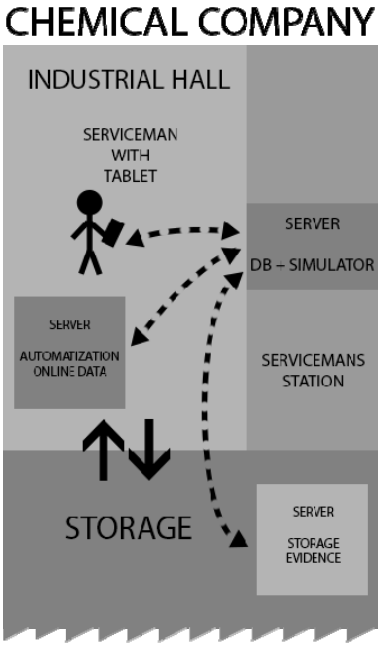
The high degree of automation allows the factory to process a large number of products (you can see example of one process at Figure 7). The factory itself contains several lines for different types of chemical metal coating (bluing, electrolytic and galvanic lines), while the actual chemical bath is subject to change. It is also true that one chemical bath can be used for a number of metal coating lines (i.e., if the chemical bath has a size of $20 \times 4 \times 2$ metres, it can be used for four metal coating lines and each line can process different types of products with the same surface finish).

Figure 7 Chemical plant line diagram



The problem of the plant, however, is that due to the chemicals and enormous currents (thousands of amps at 3–5 V) there are frequent malfunctions on the lines. The chance that a fault occurs in one line, in any shift is up to 7% (up to 14% with lines fully loaded). If the factory has 40 lines, it is a relatively high number. But there is no chance to get more reliable devices due extreme chemical conditions in the company. Hall structure and data flow can be seen at Figure 8.

Figure 8 Physical situation in company



For the increase of security of application the principle defined in Cimler et al. (2015) can be used, but in manufactory it is not needed.

7.2 Motivation for solving simulations

If there is a malfunction, is it better to stop the operation and remove the error, is it advisable to have the production completed, or is it ideal to switch the production to another line? And what happens with the products? Will there not be unreasonable queues? Will there not be a plethora of rejects? Alternatively, will the products degrade that have gone through the preparatory process and are still waiting for a galvanic bath?

For such complex problems analytical methods are stretched to their limits, and better way is to use simulation, as is noted by Manling (2009).

At present, all the problems that do not require acute repair are postponed until the production is completed and only then they are dealt with. However, the problem is that some of these errors reduce the efficiency of the process (such as restrictions of the functionality of some pumps for mixing the baths) when it is necessary to slow the production by about 20% and there is an increased risk of poorly metallised parts.

By applying online simulation (i.e., simulation that uses actual operating data – statuses of the input queues, warehouse and individual lines – but from the start, the simulation runs with a time base much faster than the real time) it is possible to evaluate the best solution method immediately after detecting a problem.

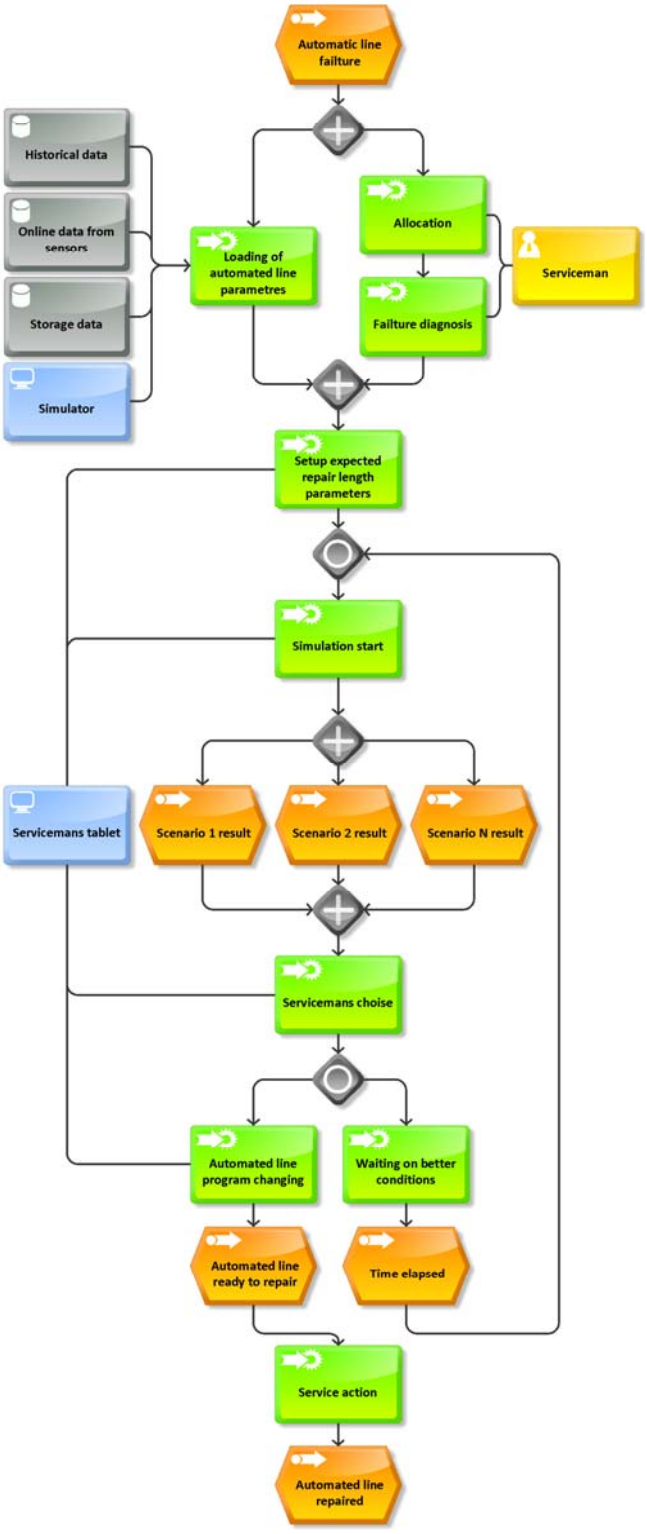
If we know the malfunction, owing to the frequent repetition of identical malfunctions, we know the time needed for the repair. Based on the parameters of the line (temperature, type, degree of metal coating, input queue, availability of other lines), the simulation can then be used to create an ideal scenario for the repair (so far, service engineers have usually not been able to evaluate all parameters only by expert judgment, as there are many variable parameters).

The simulation then can recommend redirecting the production to another line for the time needed for repair (this solution minimises the loss of unprocessed elements that have already undergone preparatory processes), suspending the line production and repairing (there will be an increase in the line capacity, if there are large input queues), or recommend continuing the production without any repair (with small input queues, or expected change in the production).

By including the simulation model to support decision making in the work of service engineers, we can increase efficiency of the total production under standard conditions by up to 2% to 5% (in excess of orders and with busy input warehouses and all lines, it is possible to increase efficiency and production throughput 7% to 10%).

Business process model, which illustrates repair sequences, can be found at Figure 9.

Figure 9 BPM designed for this solution (see online version for colours)



7.3 Real example A

One of four automatisation lines from the chemical bath has shut down. The serviceman comes to it, and finds the problem – oxidised conductor belt at the main panel. He knows that he has three solutions. He can ignore it and transfer production to other three serial lines. Also he can shut-down all serial lines passing towards the bath and take one hour to repair it. And on the last, he can call his colleague from home, then they can repair it in ten minutes, if they stop the line.

After he starts the decision-making simulation (after he set parameters of problem), he is informed that there are last 1,000 pieces which need to proceed to chemical bath. Then simulation tells him, that if he will wait for two hours, he will be able to repair problem without any losses.

7.4 Real example B

Sometimes there are minor accidents, such as problem, which decrease average temperature of a chemical bath. Serviceman identifies the problem as in thermostat, but to change it he needs to stop all line moving through chemical bath. If this problem is repaired, the time for plating increases about 10%.

After serviceman runs the decision-making simulation, he is informed that there are full input storages and if he does not repair it, then in about one hour there will be big congestion on the automatic lines. He tried to run simulator with second scenario – what if he repairs it. The simulation shows a bigger problem than before, because the congestion starts in 30 minutes. After serviceman starts the simulation with third scenario – impregnation of neighbouring chemical bath and redirection of production, the simulator tells him, that is the best solution.

Both of the examples are little bit simplified, but are enough illustrative to explain the benefits of our solution.

7.5 Physical simulation

Simulation is solved with use of all principles introduced in this article. Simulation is built on HLA, using conservative synchronisation provided by HLA.

Simulation has eight federates. Three of them are mobile simulation stations (tablets) with I/O. There is also one database connector, which uses data from warehouse. There is one compute federate, which runs on computer with RTI. The other federates are connected with sensors and other online systems of company. There is very interesting fact, that the bridge between online system of company and the simulation is solved by tablets. But these tablets are stationary and their only task is to adapt signal which comes to their USB ports, store it and use it, if it is necessary.

7.6 Software solution

Since it is a distributed solution that is to run on heterogeneous hardware platforms and it was assumed that it would be necessary to program various parts of the system in various programming languages (e.g., the tablet requires the Java language, the data acquisition system in the automation computer requires the C programming language coding), the use of HLA seemed to be ideal as it has its internal processes adapted to all these contingencies.

8 Potential of future development

Currently, the prototype works only for a limited class of malfunctions – the article seeks to present an idea that the systems can deal with using of mobile devices. It is the first thing which needs improvement.

It is desirable to extend the system itself with the capability of automatic detection and conjecture of malfunctions. Ideally also with neural networks to deliver the learning ability so that the system is able to better work with various malfunctions, their diagnostics and solutions, and be able to create scenarios for malfunctions of a new type.

On the request of servicemen, we are preparing improvements, which will be able autonomously to predict the errors. There are a few signals which indicate that a device will crash. For example, if consumption of electrical energy at chemical bath slowly, but constantly rises, there is a strong chance that in two hours occurs to overheating or rupture of anodes.

In general it may be said that further development is rather a low-level issue, and it does not change anything significant on the very ideas and principles of using combined simulations and tablets for decision support.

9 Conclusions

The article is focused on illustrating the benefits of using tablets in simulation-based decision making. Studies have made for three years, and finish in a practical application for chemical industry. Major problem has been created by non-existing community or methodology for using mobile devices in distributed simulation. As a result, we need to test a large number of alternatives.

The article presents many possibilities how to use tablets in monolithic or distributed simulation. A few alternatives are not efficient. But, as our our own software demonstrates, there are solutions that should be used to make distributed simulation-based decision very efficient. Case study shows that facilitating control of simulator can lead to its application. Consequently it leads to dramatic economies at corporate owner side.

Authors are expecting great potential in connecting of mobile devices to simulations, because it is one of the simplest ways to access simulators to its users.

References

Brozek, J., Onggo, B.S. and Kavicka, A. (2014) 'High level architecture virtual assistant framework', *EMSS Proceedings*, University of Bordeaux, Pardubice, ISBN 978-889799932-4.

Bruzzone, A.G., Fadda, P., Fancello, G., D'Errico, G., Bocca, E. and Massei, M. (2010) 'Virtual world and biometrics as strongholds for the development of innovative port interoperable simulators for supporting both training and R&D', *International Journal of Simulation and Process Modeling*, Vol. 6, No. 1, pp.89–102.

Cimler, R., Matyska, J., Balik, L., Horalek, J. and Sobeslav, V. (2015) 'Security issues of mobile application using cloud computing', *Advances in Intelligent Systems and Computing*, Vol. 334, No. 334, pp.347–357, DOI: 10.1007/978-3-319-13572-4_29.

Cimler, R., Matyska, J., Balik, L., Horalek, J. and Sobeslav, V. (2014) 'Security aspects of cloud based mobile health care application', *Lecture Notes of the Institute for Computer Sciences Social Informatics and Telecommunications Engineering*, Vol. 144, No. 144, pp.202–211, DOI: 10.1007/978-3-319-13572-4_29.

Fujimoto, R.M. (2000) *Parallel and Distributed Simulation Systems*, John Wiley & Sons, New York, ISBN 04-711-8383-0.

Kovac, P. (2012) *Reflection: Where are We Going Tablets?* [online] http://www.svethardware.cz/uvaha-kam-smeruji-tablety/34506-3 (accessed 15 July 2014).

Ku, A. (2012) *Lenovo's ThinkPad X230T Tablet PC, Tested and Reviewed* [online] http://www.tomshardware.com/reviews/thinkpad-x230t-review-benchmark,3229-2.html (accessed 15 July 2014).

Kuhl, F., Dahmann, J. and Weatherly, R. (2000) *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, Upper Saddle River, NJ; Prentice Hall PTR, ISBN 01-302-2511-8.

Letizia, N., Alessandro, C. and Francisco, S. (2015) 'Advanced interoperable simulators for training in car terminals', *International Journal of Simulation and Process Modelling*, Fall, Vol. 10, No. 2, pp.132–143, DOI: 10.1504/ijspm.2015.070482.

Letizia, N., Alessandro, C., Charlos, A. and Alessando, D. (2014) 'Hybrid approach for container terminals performances evaluation and analysis', *International Journal of Simulation and Process Modelling*, Fall, Vol. 9, Nos. 1/2, pp.104–112, DOI: 10.1504/ijspm.2014.

Manlig, F. (1999) *Computer Simulation of Discrete Events* [online] http://www2.humusoft.cz/www/archived/pub/witness/9910/manlig.htm (accessed 15 July 2014).

Marina, M., Alberto, T., Simonluca, P. and Letizia, N. (2013) 'HLA-based real time distributed simulation of a marine port for training purposes', *International Journal of Simulation and Process Modelling*, Fall, Vol. 8, No. 1, pp.42–51, DOI: 10.1504/ijspm.2013.055206.

Mocny, O. (2009) *Real-time Physics Simulation for Mobile Device*, Bachelor thesis, Charles University.

Rabelo, L., Sala-Diakanda, S., Pastrana, J., Marin, M., Bhide, S., Joledo, O. and Bardina, J. (2013) *Simulation Modeling of Space Missions using the High Level Architecture* [online] http://www.hindawi.com/journals/mse/2013/967483/ (accessed 15 July 2014).

Reda, T., Elhaq, S.L. and Ahmed, R. (2014) 'Modelling methodology for the simulation of the manufacturing systems', *International Journal of Simulation and Process Modelling*, Fall, DOI: 10.1504/ijspm.2014.066372.

Reilly, D. (2006) 'Java RMI & CORBA – a comparison of competing technologies', in *Java Coffee Break* (cit. 2015-01-20) [online] http://www.javacoffeebreak.com/articles/rmi_corba/.

Schön, O. (2013) *How Powerful is Your PC? New 3DMark Test Measures the Ability of PC and Tablets and Phones* [online] http://tech.ihned.cz/hry/c1-59252020-3dmark-pro-pc-i-iphone-a-android (accessed 15 July 2014).

The Institute of Electrical and Electronics Engineers, Inc. (2010a) *IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules*, IEEE, New York, ISBN 978-0-7381-6251-5.

The Institute of Electrical and Electronics Engineers, Inc. (2010b) *IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specifications*, IEEE, New York, ISBN 978-0-7381-6249-2.

The Institute of Electrical and Electronics Engineers, Inc. (2010c) *IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification*, IEEE, New York, ISBN 978-0-7381-6247-8.

The Institute of Electrical and Electronics Engineers, Inc. (2010d) *IEEE Standard for Distributed Interactive Simulation Application Protocols*, IEEE, New York, ISBN 07-381-0992-4.

The Simulation Interoperability Standards Organization (2001) *Independent Throughput and Latency Benchmarking for the Evaluation of RTI Implementations*, Fall, Orlando, Florida, USA, DOI: SISO-01F-SIW-03.

Notes

1 Flops are a measure of computer performance, indicating a count of basic floating-point operations per second.

2 However, it is necessary to note that it is possible to create other than simply linear non-hierarchical simulation models, which makes it possible to use a number of central RTI components in a single simulation.