Imprecise Computation as an Enabler for Complex and Time Critical HLA Simulation Networks

Cesar Tadeu Pozzer, Dr.
University of Skövde - HiS
Box 408, 541 28
Skövde, Sweden
pozzer@inf.ufsm.br

Joni A. Amorim, Dr.
Universidade Estadual de Campinas - UNICAMP
Cidade Universitária "Zeferino Vaz", 13.083-970
Campinas, Brazil
joni.amorim@gmail.com

Per M. Gustavsson, Dr.
Combitech/Saab
Box 46, 541 21
Skövde - Sweden
per.m.gustavsson@saabgroup.com

Jonas Mellin, Dr. Ilona Heldal, Dr.
University of Skövde - HiS
Box 408, 541 28
Skövde, Sweden
jonas.mellin@his.se, ilona.heldal@his.se

Anibal T. Azevedo, Dr.
Universidade Estadual de Campinas - UNICAMP
Cidade Universitária "Zeferino Vaz" - 13.083-970
Campinas, Brazil
anibal.azevedo@fca.unicamp.br

Keywords: scheduling for imprecise computation, HLA, simulation, joint fires, real-time systems, integration.

ABSTRACT: A trend over the past years is that simulation systems for training are being connected in simulation networks, allowing the interaction of teams spread in distributed sites. By combining interconnected simulation systems the simulation complexity increases and may affect time-critical simulation tasks in a negative way. As a consequence, the training simulation objectives may not be met. The same problem may occur when performing, for example, mission rehearsal on site, since available computation resources are usually very limited in this scenario, or for a joint fires scenario, where the large and complex functional chain (including intelligence, C2, forward observer, pilots, etc.) may overload existing resources. In this work, the technique of imprecise computation in real-time systems (ICRS) to preserve time-critical simulation tasks is presented. The ICRS technique allows time-critical tasks to produce quicker solutions for approximate results and saves computational resources. This paper discusses the main advantages of the ICRS technique by a review of the commonly used optimization concepts built upon imprecise computation field. The paper ends with presenting a work-in-progress: an architectural solution for aligning ICRS with the High Level Architecture (HLA), standardized as the IEEE 1516-series.

1. Introduction

With the current performance in both computers and networks, distributed systems enable execution of high-fidelity simulation models, and thus entail realism in training. The use of distributed system technologies allows the inclusion of more nodes (processing entities), ideally loosely coupled, in order to increase the processing power. These technologies bring team members from various sites together and allow new
forms of communication, cooperation, and interactions. Training can also be connected into networks so that multiplayer environments can be created.

However, distributed simulations often tend to expand, since the users may add more models and nodes into the environment, thus increasing the need for required processing and networking resources.

Three immediate bottlenecks problems can be identified in distributed simulations. The first refers to network scalability, due to limited bandwidth or the large amount of data exchanged among the nodes in the simulation. The second refers to processing scalability, due to the implementation methods. Models may be choked in intensive processing in nodes. Finally, the third refers to simulation scalability, due to the simulation complexity. The needed data for the simulation itself cannot be shared between the large number of computers, causing not all computation be made on time.

In a controlled environment these challenges can be managed better to some extent. It is possible to: replace or add more bandwidth to the network; add new nodes and/or processors to the network; and conduct a rigorous setup and testing before the simulation is executed. However, when performing simulations on the field, i.e. outside the training facilities, those problems may become even more evident. As an example, consider a mission rehearsal performed on the field just before a real mission begins. Due to that the boundary of the scenario is not known at the simulation system development, the simulation systems may need more processing power than are available in order to meet the execution deadlines. In a mission rehearsal scenario, it is difficult to build up a robust infrastructure to run the simulation.

Another scenario that adds to complexity is ‘Joint fires training’. Here the challenge is to interconnect several systems, with various fidelity levels, that not necessarily were designed to be interconnected in a simulation system. The training requirements can also range over basic training, group training, functional training, and mission rehearsal. Further, it is necessary to provide scalability so that the training systems can be executed on everyday computers often with limited memory and processing capabilities.

Different countries may adopt different regulations for using simulators as well. For example, according to the 2013 regulatory ordinance PN-1.814-MD from the Ministry of Defense in Brazil, all simulators need to be purchased or to be developed by the Brazilian Armed Forces need to be compliant to the IEEE 1516 HLA in order to facilitate interoperability.

Until now, the authors could not find any specific work that uses scheduling for imprecise computation on integrated military simulation in conformity with the HLA standard. However, there are many studies focusing on using effectively the level of details or the level of fidelity in simulation [1]. For example, by allowing different sketching techniques in urban planning, objects used to be visualized and simulated depending to their relevance in accordance with the underlying concept [2]. The most similar work found in the literature is the one presented by Schricker and Lippe [3]. They use the concept of Selective-Fidelity to allow the simulation, based on criteria such as computing resources or user interest as a way to adjust the level of fidelity. They proposed a system implementation based on HLA considering the existence of a set of similar HLA federates that perform the same task, but at different levels of fidelity. Through the HLA ownership management service, appropriate federates can be assigned to perform the computation of specific simulation objects at different fidelities. However, this approach requires that models should be executed in parallel, which also means additional requirements for computational power.

This paper presents the technique of Imprecise Computation in a Complex and Time Critical HLA Simulation Network that attempts to overcome the identified bottlenecks described above. The proposed solution is to use imprecise computation in real-time systems (ICRS) in order to degrade the simulation fidelity [4] in a controlled way. The trade-offs are scheduled in real-time, so that fidelity, resource usage and timeliness are balanced. In other words, if a simulation system is high processing and time-critical, one approximate result delivered in time should be better than waiting for the correct result.

The main contribution of this work is the presentation of concepts and architecture that explores imprecise computation in the context of HLA.

This paper is organized as follows. In Section 2 we present definitions and concepts about scheduling for imprecise computation and some applications for this technique. Concepts about HLA are presented in Section 3. Section 4 presents an architecture to integrate scheduling for imprecise computation into HLA, and concluding remarks are presented in Section 5.

2. Scheduling for Imprecise Computation

In simulations were fast response time is more important than high fidelity the technique of Imprecise Computation [5] [6] is a promising approach since it enhances the importance of processing tasks and also
meeting quality requirements and deadlines at the same time. It allows dynamically balancing quality (e.g., fidelity and functionality) requirements, resource usage, and timeliness. The aim is to use alternative algorithms to solve the tasks with different quality, resource usage requirements, time constraints, as well as considering new abstraction levels (e.g. high-level – for example, for a single aircraft behavior, mid-level – for example for a group of aircraft behavior, or low-level – for simulating complex air-traffic flows). If the simulation requires fast response, less resource demanding tasks can be used for simulating phenomena with lower fidelity.

2.1 Basic Definitions

Scheduling for imprecise computation is a mechanism that uses policies to control processing tasks to meet goals and deadlines. Tasks specification and policies are heavily dependent on the problem that needs to be modeled. This paradigm depends on the concept of tasks. Task is the basic element of any scheduling algorithm, the smallest schedulable unit of processing. They are also responses to requests, and often have temporal scope. Essentially, scheduling for imprecise computation is used to say how tasks (with timing constraints) should be allocated to resources (processor, memory, disk, etc.), frequently in a real-time environment, so that they can meet their deadlines.

The scheduler takes as input the tasks and criteria. It has to sort and dispatch task for execution according to the selected criteria. The algorithm for evaluating the criteria should be as simple as possible. The more complex they are, the more costly they are to be evaluated. In some situations one may not be able to process all tasks in the highest resolution. When such situation occurs, according to the importance of the tasks and its associate deadline, the scheduler, based on predefined criteria, choose what needs to be done. The various algorithms have different computational resource usage and data sets for processing the same job. High priority tasks (these have to have high weight) need be performed with more accurate algorithms, whereas low prioritized tasks can be performed by simplified algorithms.

Basically, there are three ways of scheduling tasks for imprecise computation.

1) Mandatory/optional, each task has an optional and a mandatory part. The mandatory part must be processed, whereas the optional only if there is available time.

2) Anytime, tasks are modeled in such a way that the results continuously improve as the task runs. The scheduler does not have to look into the future to decide when to stop the task. It just has to consider specific deadlines. The processing stops when the deadlines are reached.

3) Alternative, in this approach, the scheduler needs to know in advance, already before the execution starts, how to process each task. Later is not possible to correct a wrong decision. Depending on the resolution level tasks are scheduled, different quality results may be produced, leading to eventual and specific errors in the simulation.

2.2 Related Works on Scheduling for Imprecise Computation

Scheduling for imprecise computation is used in many different research fields and for many different purposes.

Gao et al. [7] describe a real-time scheduling algorithm to handle media stream delivery over a network. They use the Mandatory/Optional approach (number one above). Imprecise computation aspects are used for the definition of tasks consisting of mandatory and optional parts (subtasks). Considering the fact that a video can be reconstructed by a combination of layers, the mandatory part is associated with base layer streams, responsible for the basic visual quality, whereas the optional part deals with sub-streams of enhancement layers, responsible for continuously improving the quality of the reconstructed video. Trade-offs are basically related to bandwidth limitations and transmission errors. Based on those two factors described above, the scheduler can decide how to select and transmit data packages. Different bit rates have to meet different qualities based on the available bandwidth and playback deadlines that can selected on the client side.

Breuer [8] extends the concept of impreciseness for VLSI chip manufacturing, introducing the concept of error tolerance, by considering acceptable minor hardware defects, since the difficulty of producing defect-free chips. Breuer does not suggest any specific imprecise computation approach to handle the problem. He states that in a multimedia application, for example, for MPEG and for JPEG hardware encoders, computational results do not need to be correct all the time. By allowing erroneous, but acceptable results, the manufacturing yield, which in turn is continuously decreasing as a result of the increasing complexity of the chips, can be increased.

Millan-Lopez [9] proposed a congestion-control scheme to avoid failures in package transmission when traffic peaks occur. The approach is similar to that of Gao et al. [7] and uses the Mandatory/Optional approach (number one above). A message is considered to be composed as a set of packets. Some of those packets contains basic information, and are associated to mandatory tasks,
whereas other packets contain complementary information, being those ones associated to the optional tasks. The latter can be discarded in case of network congestion so that messages can meet their deadlines. In opposition to the multiple levels of quality [8], this approach considers only two levels of quality: basic or extra. A similar approach for compressed image and video transmission is presented by Chen et al. [10].

Tung et al. [11] use imprecise computation concepts to model the control power consumption for a flash memory storage in real-time. Their approach uses the Alternative approach (number three above). The solution consists of defining tasks that can operate at different voltages to attend requests for memory access and at the same time satisfying time constraints. Data can be accessed faster by using higher voltage, at the cost of higher power consumption. The proposed solution considers tasks as having mandatory and optional parts. The mandatory part refers to the execution of the whole task in high voltage, whereas the optional to the execution in low voltage. The scheduler was designed to minimize the power consumption of tasks and, at the same time, meeting deadlines.

Nakanishi et al. [12] use imprecise computation to model the traffic flow in order to increase throughput and avoid collisions of automobiles, regardless how congested the road is. They use the Alternative approach (number three above). Based on information collected along the expressway, the system must compute distances to determine a) how congested the zone is and b) which actions regarding speed, distance and lane-change decisions each driver must perform to avoid collisions. Collisions may happen in case of deadline violations. Since the computation required is dependent upon the number of vehicles, the scheduler uses the Alternative approach to process the tasks: when the zone is congested, a simple version of the simulation is performed and the opposite when the zone is not congested.

3. HLA Fundamentals

HLA is a service-oriented, flexible, and scalable architecture for distributed simulation. It uses a FOM (Federation Object Model) to describe objects (persistent simulated entities over time) and information that can be exchanged within the whole federation. Subsystems, when integrated form a Federation. One federate can represent a single object (e.g. a tank) with its attributes, an aggregation of a large number of objects, or even more abstract and complex simulation, such as the complete Army of a country. Since information is not domain specific, HLA allows wider applicability. Additionally to the FOM, a Simulation Object Model (SOM) contains a description of produced and consumed information a single federate is supposed to manage [13].

A Run-time Infrastructure (RTI) is used to provide interoperability services for data exchange and coordination by using standard APIs through which participating members can get access. RTI must meet HLA requirements; however, implementation details are often depending on the developers’ decisions. Since the HLA specification does not impose restrictions on the RTI implementation, it is possible to add new functionalities. To be HLA compliant, RTI must provide six management services: Federation, Declaration, Object, Ownership, Time, and Data Distribution. It must also provide one additional Support Services as well [13] [14].

Figure 3.1 depicts a generic scenario with four federates and eight objects running in a set of three nodes connected over a network.

![Figure 3.1. Generic HLA scenario.](image)

Bottlenecks can easily arise depending on bandwidth and processing limitations and on the number of federates, of objects and of exchanged messages. Bottlenecks may refer to the amount of required processing to run the local simulation, to the amount of received messages to deal with, or to the amount of messages to transmit over the network.

Based on scheduling for imprecise computation, the next section presents possible solutions for these problems.

4. Proposed Architecture

Modeling scheduling for imprecise computation is very dependent on the problem being considered. In Section 2.2 several examples of Imprecise computing for various domains (e.g. media streaming over networks and transportation systems) are presented. The proposed solution uses imprecise computation and is focused on
the definition of a generic architecture, but at the same time trying to explore the maximum number of possibilities over HLA.

At a high level there are three tasks that need to be conducted.

1) **Object Behavior Analysis**: models the tasks association with objects (e.g. representation of behaviors), and what models that are available or what optimizations that can be made in order to save processing and bandwidth resources. The basic idea is to have different implementation for different levels of representation. The basis for the analysis is the Training Need Objectives (TNO) or simulation purpose.

2) **Scheduler Policy**: The definition of a scheduler (local coordination mechanism) for each federate to coordinate task execution, and consequently to coordinate local resource usage. In our proposal, these schedulers should communicate with the Resource Policy Manager to either send requests or receive orders of how to behave.

3) **Resource Policy Manager**: It is an additional coordination mechanism to the existing one present in the RTI, and has the role of coordinating schedulers in the appropriate way. It acts also as a centralized point for coordination.

Figure 4.1 depicts these three elements. They are detailed in the next sections. In this figure, each dotted square represents a federate, which in turn can have many objects. Usually each object is processed by a single task. The Resource Policy Manager communicates with schedulers present in each federate.

4.1 Object Behavior Analysis

Each federate may have an arbitrary number of objects. Each object may represent, for example, a soldier, an aircraft, a missile, a tank, a ship, among others. All computations a single object is allowed to perform must be put into a task, or into a set of tasks, implemented in such a way that they can run at different resolutions. Most computations are related to the processing of the behaviors, which may include movements, decision, perception and message exchange.

The main challenge sits on how to implement tasks for each kind of possible object in such a way the scheduler can select tasks for execution according some priority, thus allowing trade-offs in real time according to available resources and required fidelity. This is dependent on the simulation particularities and, due to this, it will not be treated here.

For a joint fires scenario, our experience in imprecise computation tells us that the best strategy is to use the Alternative approach (See Section 2.1). For each simulation step, we must select in advance the computation each tasks will perform. We must have in mind that both processing and network should be optimized, so the solution must meet these requirements.

Objects must be sorted in some kind of priority, so that low priority objects (or tasks) can be selected by the scheduler to run in lower resolution, while high priorities objects keep running in the highest possible resolution. From a perspective on training on joint fires, the trainee’s position can be used as the main reference point for selecting the priority of the objects.

Objects can be seen as being in “foreground”, “background” and “backstage” resolutions in relation to the avatars (entities under user/trainee control), as presented in Figure 4.2.

![Figure 4.1. Overall proposed architecture.](image)

![Figure 4.2. Foreground, background and backstage configurations.](image)
In foreground, we run the algorithms in the highest resolution. Objects in foreground are more likely to interact and influence decisions taken by trainees. The thresholds for defining what is in foreground, background, and backstage can be adjusted in real-time according to regulatory instructions sent by the Resource Policy Manager. Objects in backstage, for example, can use low fidelity algorithms and they can be update less often than background, and even less often than foreground objects. Less updates implies also less update messages, thus reducing the network usage. Only the background/backstage simulation will be compromised with low fidelity simulation. But in a strategic simulation, for example, small details will not impact the training value.

The importance of objects may vary among federates. For a given federate $f_i$, all its objects $o_1, o_2, \ldots, o_n$ can be seen as foreground. For federate $f_j$, the same objects can be seen as backstage, for example. This means that we can combine computation levels and network updates accordingly. One object can be processed in the highest resolution for local interaction while it is updated less often in other federates.

Many implementation optimizations can be used to have objects running in different resolutions. Navigation meshes, perception mechanism, decision-making processes, and movements performed at different resolutions or at different time intervals can be adjusted for this purpose. All those decisions are under responsibility of the scheduler. Reducing the update rate causes a reduction in the number of communication messages (packets) and required processing in other nodes to process incoming messages and update local simulations. For example, dead-reckoning approach [15] can be used to estimate the position of remote moving vehicles if updates are not frequent and the local simulation requires high fidelity.

4.2 Scheduler Policy

The scheduler has the responsibility to define the way tasks should perform. It has a list of all objects and tasks that are assigned to the federate. Each federate must have one scheduler, as presented in Figure 4.1. The scheduler must be simpler; if not it may impact negatively on the processing load of the federate. The proposed scheduler receives instructions from the Resource Policy Manager to select how tasks should be executed.

The Resource Policy Manager assumes control of all imprecise aspects of the simulation; the scheduler should put decisions in practice. The schedulers do not take decisions by themselves. When a problem is detected, it is reported to the Resource Policy Manager that sends back general instruction of what to do, being either mandatory or advisory (see Section 4.3).

There are two basic ways to trade off fidelity, resource usage and real-time aspects in real time: adjusting thresholds and adjusting the frequency.

By adjusting thresholds, we define which objects are in foreground, background and backstage, as presented in the previous section. Objects in the foreground consume more processing than backstage ones. Distance from avatars or importance of the object can be used as input parameters for the scheduler. This process do not need to be performed every simulation step.

The other way is by adjusting the frequency in which objects are updated. By using time stamps the scheduler can keep track of the executions from previous tasks in the local simulation and also the last time local attributes were updated in the whole simulation. This approach can be used to ensure that even backstage objects maintain a minimum level of fidelity. This strategy also allows control of the execution of individual objects in a more precise and effective way. If a node has a lot of objects, it may be sufficient to relax one or a set of them, or even alternate relaxed objects throughout simulation steps.

4.3 Resource Policy Manager

This module controls the execution of all objects, federates and network usage. It must evaluate the simulation, identify bottlenecks, and take high-level decisions in order to keep the simulation running with the highest possible fidelity given restrictions on processing, network and real-time aspects.

It continuously receives messages from federates related to processing usage (e.g. federates can report excessive number of incoming messages or processing overload), and sends regulatory instructions to control the resolution that the simulation must take (e.g. to reduce the number of sent messages or to set the desired level of fidelity). Regulatory instructions may assume two different roles: mandatory or advisory. Mandatory regulations force the schedulers to increase or decrease fidelity, irrespectively of the local processing load and the importance of the objects being simulated. Advisory regulations only notify the schedulers about the possibility to adjust the local computation whenever necessary. In the last case, each federate can evaluate if it is necessary to reduce or increase the fidelity of the simulation. For example, when the simulation begins it can be set that whenever a federate cannot maintain real time execution, it can reduce simulation fidelity by itself without any further instructions or requests.
Another design aspect refers to the way the Resource Policy Manager is put into the HLA standard. We foresee three possibilities, as depicted in Figure 4.3. Regardless of the adopted solution, there must be an agreement among all parts. The possibilities are as follows:

a) As an external module to HLA: in this scenario, the Resource Policy Manager acts as an external component. The communication may be implemented apart from the HLA standard.
b) As a federate: no modifications in the RTI are required, which just have to manage regulatory instructions between the Resource Policy Manager and federates. The Declaration management service can be used to manage publish and subscribe intentions for this purpose. The core implementation remains on the dedicated Federate;c) Into the RTI (HLA standard): this implies in strong modifications into the RTI in order to implement these new functionalities;

For case a), an additional communication mechanism must obligatory be implemented to allow exchange of the “imprecise” messages. Since this process runs as a separate service apart from HLA, coordination mechanism may not become completely integrated. The main disadvantage of this approach would be the possible existence of a number of different implementation and design solutions, hampering the integration of simulators from different vendors. Independent of the adopted solution, guiding principles of how to build federates to support imprecise services should be designed.

For cases b) and c), all federates to be HLA compliant, should know this new “Imprecise Service” in order to be allowed to join the HLA federation or to have access to all functionalities. In this setup, when federates register to the HLA, they can inform the maximum capability of processing or maximum data they can handle, the imprecise computation capabilities, and the number of objects they have. If the federate is expensive or cannot handle impreciseness, it may be removed out of the simulation. We can use existing HLA services or add an additional Imprecise Computation service to the HLA to control the imprecise aspects. The difference between b) and c) is that whereas b) is just using the HLA/RTI as a message channel, c) is implemented in the HLA/RTI as a service.

In both solutions the Resource Policy Manager should keep track of relevant information about individual federates in the simulation such as: processing power, imprecise computation capabilities, the number of object, and a short term history of the processing load. This module can also have a monitor that displays information about all federates in the simulation, including processing demands and detected bottlenecks. In the same way as RTI operates, the Resource Policy Manager does not keep a track of objects attributes either.

5. Conclusions

In this work we introduced the concept of scheduling for imprecise computation in the field of simulation over HLA. Imprecise computation has proved to be very effective in many different fields. Also, it presents many features that apply directly to the field training since it characterizes a real time simulation that may require large amounts of computations for simulating objects that assumes different roles and importance. It also applies to this area since in many real training situations is not possible to keep real-time by just adding more computational resources, thus degrading the simulation fidelity is a reasonable solution. The task concept
matches the need to represent individual objects into the HLA simulation.

Handling trade-off in real time is very important, due to the existence of bottlenecks on both available processors and networks. Imprecise computation showed to be effective in handling tasks in order to manage different situations and problems that arise when the size of the simulations becomes too large, not only regarding the number of objects, but also the number of interconnected nodes and available bandwidth as well. The proposed architecture attempts to explore a large diversity of configurations, optimizations for different forms of implementations.

Practical experiments need be performed in the future, in order to systematically evaluate the effectiveness of the proposed solutions and architectures. We also need to evaluate in a structured way the details related to integrating imprecise concepts into the HLA; it is also a need to foresee the necessity of creating more accurate recommendations for better integration among different implementations and vendors. Imprecise computation promises to be useful, and HLA can be extended in this direction.

Another future research direction is the investigation of the use the Resource Policy Manager to negotiate ownership transfer over HLA. Consider, as an example an aircraft shooting a facility on the ground. The bomb, the aircraft and the facility are objects. The ownership of the bomb belongs to the aircraft. But, when the bomb comes closed to the facility, since the facility has a better model to calculate the damage, it should assume the ownership of the bomb. The ownership is handed over from one simulation and goes to another simulation. However, if due to overloads the more appropriate module cannot calculate, it may ask for the shooter to calculate, and then the simulation will be imprecise.

The authors anticipate that the Imprecise Computation Service will be added into HLA at some point in time.

Acknowledgements

Cesar Tadeu Pozzer has been granted scholarship from CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, in Brazil, from CISB - Centro de Pesquisa e Inovação Sueco-Brasileiro, in Brazil, and from Saab AB, in Sweden.

References


Author Biographies

CESAR TADEU POZZER is a postdoc researcher at the University of Skövde, in Sweden, in collaboration with SAAB Training and Simulation. He got his PhD from Pontifical Catholic University of Rio de Janeiro, and Master from Technological Institute of Aeronautics, both in Brazil. His main interests are virtual environments, simulation, and real-time simulations for training purposes. Dr. Pozzer is a professor at Federal University of Santa Maria - UFSM, Brazil.

JONI A. AMORIM is PhD in Engineering. Postdoctoral Fellow at the University of Skövde - HiS, in Sweden, in collaboration with SAAB Training and Simulation. Dr. Amorim previously worked as a researcher and as a teacher at Universidade Estadual de Campinas – UNICAMP, in Brazil. His research has an emphasis on multimedia production management, project portfolio management, distance education and training based on serious games and simulations.

PER M. GUSTAVSSON is a Principal Research Scientist at Combitech a Saab group company. Dr. Gustavsson research is in the domain of Command and Control and supporting fields and technologies ranging from operational art, cyber, processes, methods and interoperability to technological infrastructures and models. He was co-charing the MSDL and C-BML PDGs within SISO 2006-2014. Dr. Gustavsson is also affiliated with the department of Command and Control Science at the Swedish National Defence College, and the Center of Excellence in C4I at George Mason University.

JONAS MELLIN is a professor in Computer Science and member of the Information Fusion Program at the University of Skövde - HiS, in Sweden. His main research interests are in software architecture (in particular, decentralized processing), complex event stream processing, databases and real-time systems.

ILONA HELDAL is a professor in Informatics at the University of Skövde - HiS, in Sweden, with a focus on Interactive Systems. She is the Program Director for the Industrial PhD Program in Applied Informatics, ApplyIT. Her main research interest is on collaboration and interaction in virtual environments and on how visualizations support collaboration.

ANIBAL TAVARES DE AZEVEDO is a professor at Universidade Estadual de Campinas – UNICAMP, in Brazil. Dr. Azevedo graduated in Applied and Computational Mathematics from UNICAMP (1999), holds a Master’s degree in Electrical Engineering from UNICAMP (2002) and received his Ph.D. degree in Electrical Engineering from UNICAMP (2006). He has experience in software development and in mathematical modeling for Production Engineering and Planning, for Scheduling of Power System Operation and for Education. His research has an emphasis on Linear Programming, Nonlinear Programming and Mixed Dynamics.