A Framework of Software Process for Interactive Training Simulators

Anelia Ivanova

Abstract: *The paper describes a generic framework of software process for interactive training simulators (ITS) development. The framework outlines an operational concept to describe how the developer team will accomplish work and includes specifications of team roles, artifacts and core workflow of life cycle activities. A basic architecture of an ITS is also presented.*

Key words: *Software life cycle, Software process model, Computer based training, Simulation, Interactive training simulator, UML, Activity diagram, Use case model, Class diagram, Component diagram.*

INTRODUCTION

As the move towards on-line, computer-based instruction continues, the use of simulations as a training tool has increased significantly, thus providing an important enrichment to the learning environment. The linkage of simulations to on-line instruction enables the student not only to demonstrate his or her understanding of the instructional material presented, but also the application of that instructional material for solving problems in a realistic environment. In [1] are discussed three instructional uses of simulation:

1. Pre-assessment of knowledge, where the learner is asked specific questions about the running simulation he/she is observing, where the questions can be modified independently from the simulation.
2. Practice of skills, where the learner can use knowledge information he/she learned through didactic or interactive instruction. The learner receives immediate feedback in the form of specific results shown in the simulation.
3. Final assessment, where the learner's relevant knowledge and skills can be assessed within a single, cohesive training and assessment environment, providing immediate feedback and remediation as necessary.

The importance of simulations will rise with the increasing penetration of computer-based training therefore the requirements to their quality will become higher. Here arises the question: How to ensure the development of effective, usable, highly interactive and realistic simulations, closely adequate to learner's needs?

The answer of this question requires at first an understanding about the specifics of the simulation as a training tool. Some well-grounded definitions and classifications are provided in [2]. According to them, a simulation is a computer program that models some aspect of the real world (object, system, phenomena, etc.). Simpler simulations have a fixed set of parameters and can only simulate one situation. The student can only watch the simulated object, not make any changes. Most simulators, however, allow the user to change the parameters in order to see their effects. These *are* interactive and allow learning by experimentation. Pure simulations (with no tutorial or rule-base) can only provide feedback on the users *actions*. If a rule-base is added then some degree of feedback on the quality or correctness of actions can also be provided, but not feedback on the user's *conceptions*. Tutorial-simulators offer feedback on a student's concepts as well as their actions. They usually remember the student's previous actions which gives them better access to their conceptions.

Current research in this area is oriented in two directions – in the first one, researchers focus their efforts to creation of generalized or specialized environments for development of various scientific or technical simulations [3, 4, 5, 6]. The fundamental problems with these environments are the instructional design neutrality and the lack of capabilities to build effective instructional interactions. Normally the instructional designer

is not a capable programmer, and the proficient programmer is not an instructional designer, therefore development of instructional effective training simulator requires a team effort. Approaches like discussed in [7, 8, 9, 10, 11] focus researchers' efforts in creation of specialized training simulators, conforming both to pedagogical and software design principles and result in construction of pedagogically considered training products. However, there is a lack of research, directed to systematization and unification of activities and artifacts, related to the life cycle of an interactive training simulator (ITS).

It is well known that a well judged planning and management of software life cycle results in a qualitative software product, therefore a generic framework of software process for ITS development will benefit instructors and developers in their attempt to create learner-centred training simulations. ITS is a relatively small application and use of an existing software process model (e.g. USDP, RUP) is not quite reasonable, because of its complexity (these models are oriented to development of large complex systems). By this reason a generic framework for ITS development is proposed and discussed in this paper.

LAYOUT

The proposed framework outlines an operational concept to describe how the developer team will accomplish work. The framework includes specifications of team roles, basic artifacts and core workflow of life cycle activities. An analysis of the existing solutions leads to the conclusion that an ITS basic architecture should include the modules, presented in Fig.1. Taking into account this architecture model, we can define the team roles and their responsibilities that will ensure its implementation and bringing to practice.

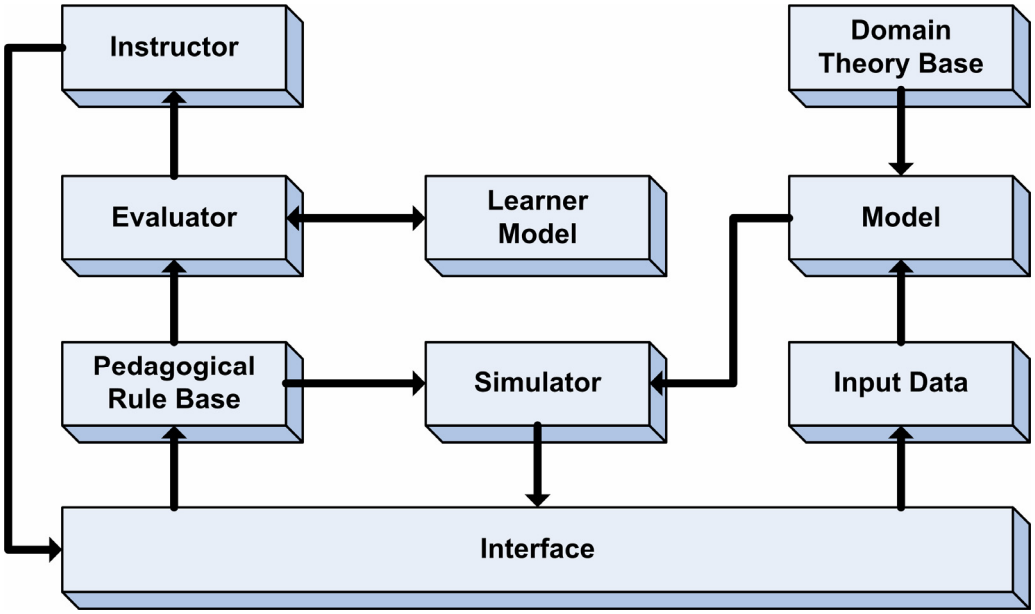

Fig.1. Basic ITS architecture

Four main roles are distinguished:

- **Project manager** – coordinates the team work and integrates the results, produced by other team members;
- **Domain expert** – has the responsibility for precise description of knowledge on the specific technical domain that will be simulated and transmission of that knowledge to the ITS;
- **Pedagogical expert** – has the main role in selection of the best instructional strategy, adaptable to the simulated domain and definition of the interactive pedagogical scenarios that will be associated with the ITS;
- **Application developer** – this role embraces four sub-roles:
  - ***Analyst*** – leads and coordinates requirements elicitation and use-case modeling by outlining the application's functionality and delimiting the application.

- ***Software architect*** – leads and coordinates technical activities and artifacts throughout the project. Establishes the overall structure for each architectural view: the decomposition of the view, the grouping of elements, and the interfaces between the major groupings.
- ***Designer*** – defines the responsibilities, operations, attributes, and relationships of one or several classes and determines how they should be adjusted to the implementation environment. In addition, the designer have responsibility for design packages or design subsystems, including any classes owned by the packages or subsystems.
- ***Implementer*** – develops the components and related artifacts and performs unit testing, constructs a build and inspects the code for quality and conformance to the project.

The core workflow of proposed framework (fig.2) is represented as an activity diagram, using UML concepts and notations. Life cycle activity states are denoted as rounded rectangles, completion transitions are shown as arrows, branches – as diamonds and artifacts – as gray rectangles. The key roles and artifacts are presented on fig.3.

During requirements and analysis phase three flows are simultaneously carried out. The first one is leaded by the ***Analyst*** and is concerned with contextual analysis and gathering, analysis and elicitation of the requirements. Contextual analysis specifies the intended users of ITS and defines the scope of requirements through answering the following questions: What do the users know? What do they need to learn? How do they learn and work? Separation of requirements into groups will benefit further development therefore two groups of them are defined: ***Software requirements***, oriented to all the aspects, concerning the general functionality and performance of the developed ITS and ***Didactic requirements***, focused to the aspects, related to didactical side of ITS (didactic concept, information presentation, cognitive load, knowledge space compatibility etc.). This flow completes with ***Requirements Specification***.

The second flow is managed by the ***Domain Expert*** and is directed to analysis of the real object that is to be simulated. The analysis is determined by two issues: "What is the structure of the simulated object?" and "How does the object work?" The outcome of this flow is a ***Simulation model*** – a set of models, describing the real object from structural and functional perspective. These models will be interpreted by simulation engine that controls visualization and representation of the simulation to the learner.

The third flow deals with pedagogical aspects of ITS and is conducted by the ***Pedagogical Expert***, who selects the most appropriate instructional strategy, a relevant learner modeling technique and basing on them defines a pedagogical rule base in a form of conditions, constraints and feedback actions. The outcome of this process is a ***Pedagogical scenario*** that will be used by ***"Pedagogical rule base"*** module to provide control of learner-simulation interaction.

The next state of the workflow is targeted to defining learner's role and building a detailed specification of intended learner's actions, taking into account all the produced to this point artifacts. To this stage the workflow has followed the sequential model, but further it becomes iterative as each iteration flow depends on test results, obtained in previous iteration. The specification serves as a base for writing a scenario, necessary for use case modeling of ITS. Scenario writing is an activity, conducted by the ***Analyst***.

The following stage is directed to modeling of ITS and desired outcome of this activity is a domain model, describing ITS from both functional and structural perspective. The functional perspective is represented by ***Use Case Model***, built up by the ***Analyst*** and structural perspective is represented by ***Class Diagram*** and further elaborated to ***Component Diagram***, both of them composed by the ***Designer*** and coordinated by the ***Software architect.***

The ***Implementer*** has the responsibility for the next stage of the workflow that deals

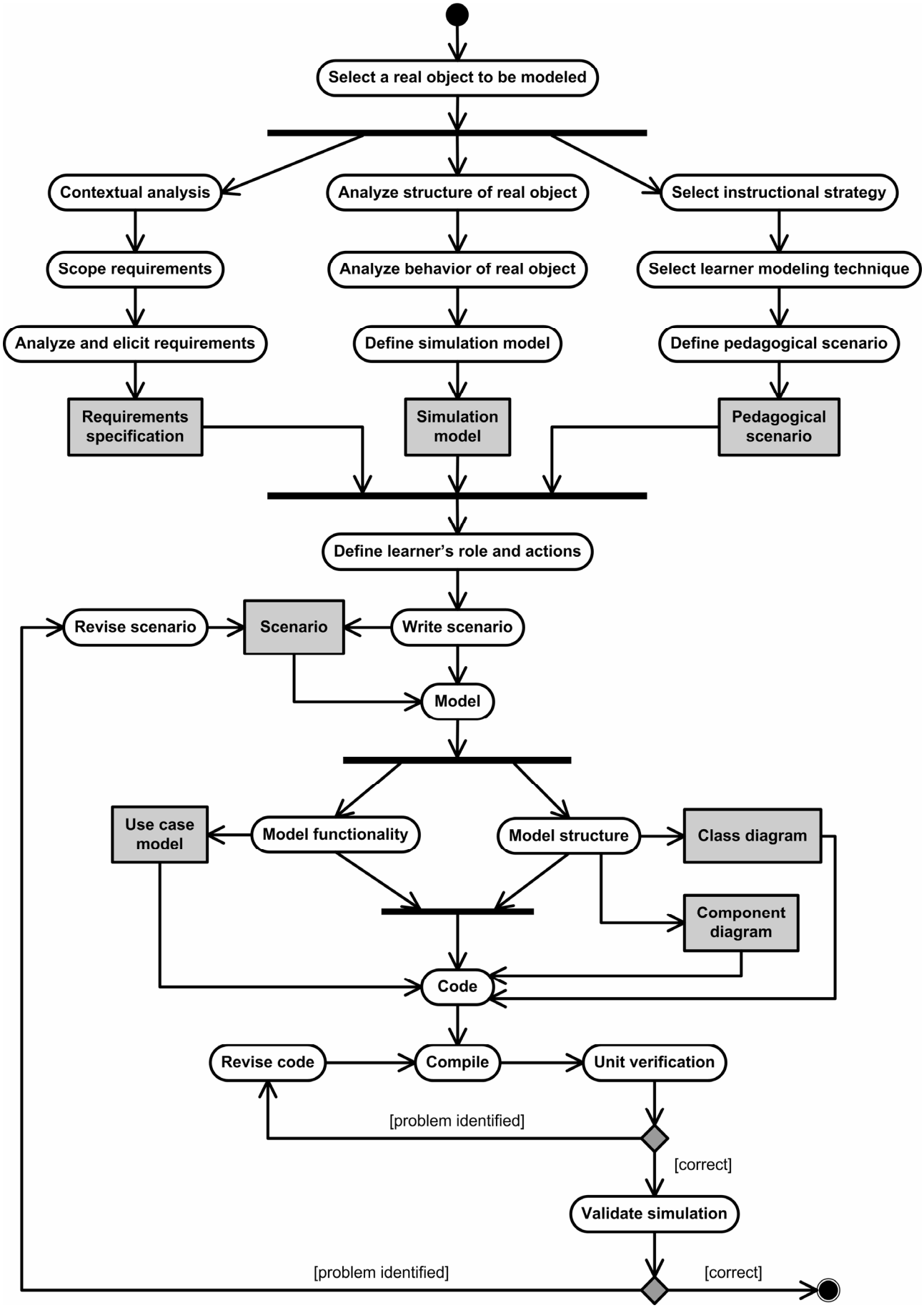

Fig.2. Core workflow of ITS Process Framework

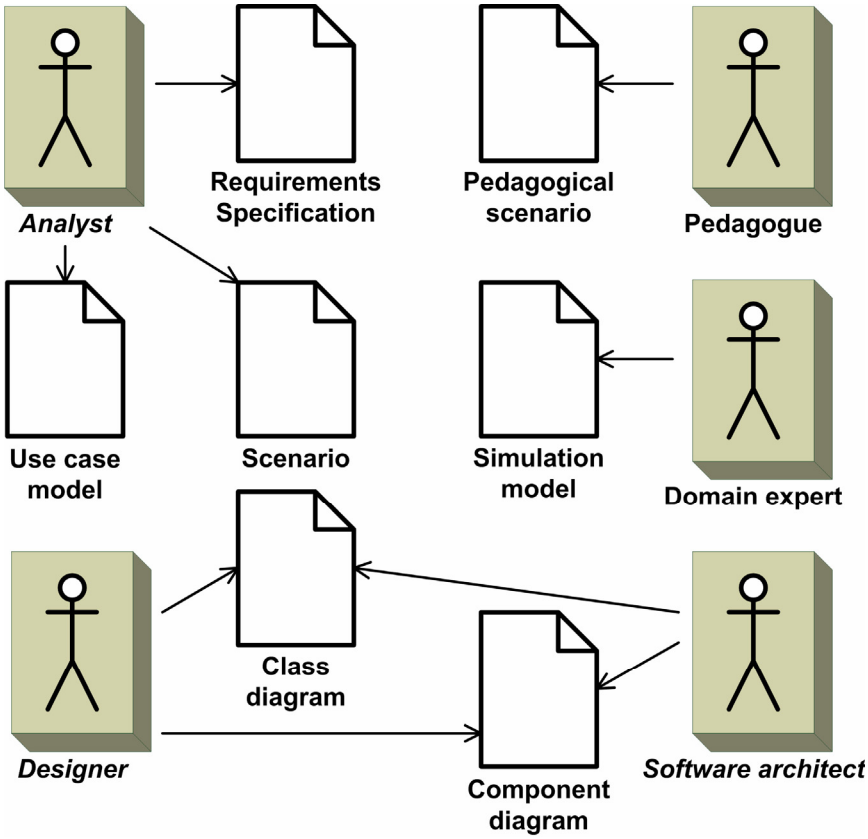

Fig.3. Roles and artifacts

with all the programming activities – code producing, compilation, unit verification and integration. Unit verification aims to test basic functionality of ITS, without considering pedagogical aspects and simulation fidelity. If a problem is identified, the code is revised, compiled and tested again. Iterations continue until unit verification completes successfully.

The next activity is conducted through the joint efforts of ***Implementer***, ***Domain Expert*** and ***Pedagogical Expert*** and its purpose is to validate the simulation via examination of feedback adequacy, correct representation of concepts, conformity between simulation and behavior of simulated object etc. Identifying a problem invokes a revision of the scenario and directs the iteration flow to modeling over again and subsequent steps of the workflow. Iterations continue until the examination group validates the simulation. After validation the implemented ITS is brought to use. If conditions of use are changed and a modification of ITS is needed, then existing artifacts should be revised which requires an overall passing through the workflow.

CONCLUSIONS AND FUTURE WORK

After analysis of existing research, dedicated to development of interactive training simulators (ITS), a basic architecture of an ITS is outlined. This architecture could be used as a pattern for ITS design.

In order to be ensured the realization of the architecture, a generic framework of software process for ITS development is proposed. A rationale of its foundation is provided.

The ITS Process framework defines the developer team members and their responsibilities, the activities they are engaged in and the artifacts they work out. A core workflow of ITS life cycle activities is proposed and visually represented via UML concepts. The relations between key team roles and artifacts are discussed and presented, too.

The achieved results will benefit each interdisciplinary team, involved in development of interactive training simulators.

REFERENCES

[1] Haynes, J., S. Marshall, V. Manikonda, P. Maloor, Enriching ADL: Integrating HLA Simulation and SCORM Instruction using SITA (Simulation-based Intelligent Training and Assessment), Proceedings of IITSEC'2004, Orlando, FL, USA, 2004.

[2] Stimson,G., B. Tompsett, The potential contribution of virtual and remote laboratories to the development of a shared virtual learning environment, The JISC Technology Applications Programme – Report 013, October, 1997.

[3] Cubert, R. M., P. A. Fishwick, A Framework for Distributed Object-Oriented Multimodeling and Simulation, Proceedings of the 1997 Winter Simulation Conference, pp. 1315-1322.

[4] Fishwick, P., J. Lee M. Park H. Shim, RUBE: A CUSTOMIZED 2D AND 3D MODELING FRAMEWORK FOR SIMULATION, Proceedings of the 2003 Winter Simulation Conference.

[5] Gueraud V., Pernin J-P., Developing pedagogical simulations: generic and specific authoring approaches, Artificial Intelligence and Education (AIED'99), Le Mans, France, July 1999, pp. 699-701.

[6] Çavuşoǵlu, M. C., T. G. Göktekin, F. Tendick, S. Sastry, GiPSi: An Open Source/Open Architecture Software Development Framework for Surgical Simulation, Proceedings of Medicine Meets Virtual Reality XII (MMVR 2004), Newport Beach, CA, January 14-17, 2004, pp.46-48.

[7] Sekar, B., C. A. Chung, Design and Development of a Training Simulator for Pre-Operational Setup Procedures on Computer Numerically Controlled Turning Centers, Journal of Engineering Systems Simulators, Vol. 1 No. 4 Winter 2004, pp. 11-18.

[8] Billard, R., A. Patterson, A. S. Ré, B. Veitch, Development of a Small Vessel Training Simulator, International Marine Simulator Forum, September 21st, 2005.

[9] Bensalem, H., T. Bensebaa, Towards a Distributed Pedagogical Simulator, CAiSE Workshops 2003, pp. 244-254.

[10] Izuha, T., T. Sato, H. Gomi, Y. Sono, T. Hida, K. Washizu, Operator's Training Simulator for Blast Furnace Plant, Nippon Steel Technical Report No. 89 January 2004, pp. 85-90.

[11] Þórólfsson, G., Simulator for operator training in the Sudurnes Regional Heating Corporation 30 MWel combined heat and power (CHP) plant, International Geothermal Conference, Reykjavík, Sept. 2003.

ABOUT THE AUTHOR

Anelia Ivanova, Assistant, McS, Department of Computing, University of Rousse, Phone: +359 82 888 827, E-mail: aivanova@ecs.ru.acad.bg.