The Role of Modeling in Clinical Information System Development Life-Cycle

Mor Peleg, Department of Information Systems, University of Haifa, Haifa, Israel

Correspondence to:
Mor Peleg, PhD
Department of Information Systems
University of Haifa
Haifa, 31905, Israel
Email: morpeleg@mis.hevra.haifa.ac.il

A model is an abstraction of some "thing" (e.g., object, system, process, phenomenon) in our world that we create in order to understand it better. Models can be physical (e.g., an architectural model of a building, a prototype of a user interface), mathematical (e.g., a model for predicting the weather, for estimating population growth), or conceptual (e.g., a clinical algorithm, the logical relationships between data items in an EMR). Models can be used to describe an existing complex real-world object or phenomenon, or they can be used as a vehicle to design a man-made object or system. While conceptual models can also be specified in narrative, in this discussion we address conceptual models that have a symbolic representation with a diagrammatic notation. Thus, conceptual models specify objects or processes, their properties, and their relationships.

A conceptual model of a proposed process or system has several important benefits, two of which are of great importance. First, the process of creating a conceptual model of a system helps its designer to study the problem domain better, to understand the system's components and their relationships, the system's desired functionality and behavior, and its interaction with users and other systems. Second, a conceptual model can facilitate the communication between different stakeholders of the process (or system), including for example, customers, end-users, (medical) domain experts, system analysts, and software developers; all stakeholders have different expectations from the system. Using a conceptual model is one of the ways by which we can narrow the design-reality gaps [1] between the conceptions of the system by its different stake holders.

Conceptual modeling can play important roles in the development life-cycle of health information systems (HIS, e.g., electronic medical record (EMR) systems, computerized physician order-entry systems (CPOE), and clinical decision support

systems (CDSS)). Organizations find it useful to use a *systems development methodology* to support the process of developing and maintaining their information systems [2]. Most system development methodologies identify several stages in the development of an information system: system conception and planning, identification and analysis of requirements, system design, implementation, and finally, use and maintenance. During the entire system development process, feedback from users (and other stake holders) is received, generating new or revising existing requirements. If these requirements are identified early in the development life-cycle then it is easier and more cost-effective to support them than if they are identified in late stages, especially after the system is already in use. The system can be maintained and updated for some time until so many new requirements are collected that a decision is reached to start a new cycle of system development.

According to Boehm [3], around 80% of all errors found in the final software system can be traced back to the requirements analysis and design phases. In other words, many of the errors are in fact due to requirements that were not elicited, were not thoroughly developed, or were misunderstood. To address this issue, several approaches have been developed to support the system development cycle, while addressing the need for getting the requirements right. Here, we list just these approaches that relate to the papers in this collection. The reader is referred to [2] for a review of other approaches. One of the approaches is the traditional waterfall approach, which advocates going through the system's life-cycle stages in sequence, spending much effort on the requirement analysis and system design phases by using conceptual modeling. This approach is similar to the one proposed by Osheroff and colleagues [4], where they developed a workbook that implementers of a CDSS can use to work through the process of identifying stakeholders, determining the goals and objectives of the CDSS, cataloging the host information system’s capabilities, and selecting, deploying, and monitoring specific CDS interventions. A second approach, Rapid Application Development (RAD) decreases the time needed to design and implement an information system by extensive user involvement, integrated computer-aided software engineering (CASE) tools that assist in updating the different conceptual models and migrate the design specifications into code, and the use of prototyping. A prototype is developed after a shorter analysis and design phase. Users can then try the prototype and provide feedback on the evolving system. A third approach is adopted by agile methodologies, which follow an iterative development cycle of system versions that have only a subset of features. The

system is released to users who provide feedback. This process embraces change in the requirements during system development. A fourth approach is that of the Unified Process (UP) development methodology, whose most well-known refinement is Rational UP (RUP). UP uses object-oriented modeling methods in an iterative and incremental (i.e., done in portions) agile design cycle. The cycle includes inception (identification of use cases and risks), elaboration (analysis and design), construction (coding and possible revision of analysis and design), and transition into the next phase, which includes correcting problems and system testing with users.

Conceptual models used for system analysis and design are used in all of the above-mentioned systems development methodologies. Different models have different focuses. Some focus on data modeling (e.g., Entity-Relationship Diagrams (ERD) [5], used in [6]) others on process modeling(e.g., Business Process Modeling Notation (BPMN) [7], used in [8]), or object-oriented models, which combine data and process together into objects. The most famous of the object-oriented models is the Unified Modeling Language (UML [9], used in [10]), which has a collection of many different models, each focusing on a different perspective of the system: use-case scenarios, static system elements, system interaction with users and external components, states and activities, and implementation.

The benefits of conceptual modeling outlined above make conceptual models natural choices for supporting the requirements analysis and system design phases, but, they could also be used in validation of the implementation of the system and in evaluation of its usage, as demonstrated by one of the papers in this collection [8].

The three articles in this collection, while focusing on different stages of the HIS development cycle, all use conceptual modeling methods. In the paper *A Business Rule Design Framework for a Pharmaceutical Validation and Alert System* [10], Boussadi and colleagues suggest an agile and business-oriented design methodology for the implementation and maintenance of business rule-based decision support systems. They describe their experience in the adaptation of the UP systems development methodology for the creation and maintenance of business rules for the validation of drug prescriptions and the generation of alerts. This adaptation, called business rule design framework (BRDF), introduces two new activities into UP: business rule specification (e.g., medication-associated laboratory testing decision rules) and business rule instantiation. Business rule specification involves generating semantic templates, domain vocabularies (for the pharmaceutical

domain), and business rules via the Semantics of Business Vocabulary and Business Rules (SBVR) formalism, which was developed by the Object Management Group (OMG), who also developed UML, which is used for system modeling in UP. SBVR business rules are written with a business object model – a conceptual model which is based on the UML class diagram, but at the same time they are expressed in a form very close to natural language, which makes it easier for domain experts (pharmacists in this case) to understand them and be involved in specification and instantiation of the business rules. The instantiation of rules corresponding to the business object model requires the identification and naming of relevant relationships between classes.

While the paper summarized above addressed the entire system development life-cycle, the two other papers in the collection address its beginning and end phases. The paper entitle Options for Diabetes Management in Sub-Shahran Africa (SSA) with an EMR System [6] by Kouematchoua-Tchuitcheu and Rienhoff, focuses on the early phases of system conception and analysis. The authors used a systematic process for performing an analysis of the requirements for an EMR system for diabetes management in Sub-Shahran Africa, where resources are poor, and evaluating the appropriateness of a potential EMR system. The methodology began with a literature analysis about information and communication options for diabetes care in SSA, followed by a need assessment field survey, which helped them identify critical issues and needs for improvement of diabetes management. These issues were used to conceive scenarios involving patient continuity-of care issues. Process-oriented analysis of these scenarios led to the specification of functional requirements for the EMR system. A conceptual model was developed to address a solution for different cases of patient continuity of care among diabetes care providers. The conceptual model was then used to analyze the potential impact on the requirements elicited for diabetes management. A potential EMR system (an open-source EMR system that was used for AIDS and multi-drug resistant tubercolsis management in Latin America) was analyzed to see whether it could support the needed functionalities inferred from the conceptual model. The needed enhancements to the ERM System were designed using an Entity-Relationship diagram that considers the conceptual model of continuity of care. A prototype of the EMR system was created. Validation of the prototype by experts and users obtained favorable results, demonstrating that it is possible to find IT solutions for diabetes care in SSA.

The paper Objectifying User Critique: A Means of Continuous Quality Assurance for Physician Discharge Letter Composition [8] by Oschem, Mahler, and Prokosch, considers the last phase of the system development life-cycle: system use and maintenance. The setting for this paper concerns a new system for composing discharge letters that was implemented at the University Hospital Enlargen in Germany. Users complained that the new system was too slow but these complaints were too vague and did not allow enough direction into what needs to be changed in the system. The authors suggest a process-based approach to objectify user critique. The process starts by interviewing users to identify a research question for in-depth evaluation. Then, a workflow model of the system to be evaluated is created using the BPMN process modeling notation. A formal hypothesis and indicators are defined, which map the user critique to the workflow steps. Indicators are measured and the results are analyzed and hypotheses are tested. Based on the results, the system is then improved (optimized).

The three papers demonstrate how quality assurance (QA) of safe, effective, and efficient HISs can be achieved and maintained using systematic processes that are tightly tied to conceptual models of the system's static elements and/or its processes. In [10], different UML diagrams were used to formulate requirements and design the pharmaceutical CDSS such that it meets the requirements of users and other stake holders. In the activities that the authors added to UP, class diagrams that specify the business object models of the pharmaceutical CDSS were used to formulate the semantic templates of medication-associated laboratory testing rules and instantiate them; In [6], functional requirements for the EMR system were derived based on a process-oriented analysis and conceptual modeling. The conceptual model of continuity of care processes helped analyze the different use-cases for the system and determine their relationship to patient data and the required data transfer between different healthcare institutions. After checking which requirements were met by the candidate EMR solution, ERD was used to design extensions to the EMR that would support the missing requirements; lastly, in [8], the process model (workflow) of the discharge summary system was instrumental in defining indices for evaluating the performance of different activities supported by the system. Evaluating these indices values helped in identifying objectively which activities of the system should be optimized. Hence, the process models were part of a method for continuously assessing the effectiveness and efficiency of the HIS.

The conceptual models used in the three papers exhibited the two benefits described at the beginning of this editorial; they supported understanding, analysis, and design

of the problem (requirements) and solution domains and facilitated communication between stakeholders. The conceptual models helped in organizing the cognitive thinking processes involved in structuring a process in terms of its components activities, the data and resources needed for them, the organizational roles taking part in them, and the interaction among system components and users. Such modeling enabled identifying differences between the processes supported by different HISs [8] as well as defining needed extensions to systems to supported needed requirements [6].

It is interesting to note that conceptual modeling was beneficial even when the conceptual models used were not standard models that have gained experience and have been shown to be effective for many system analysis and design projects; while UML models and the BPMN model are well-established models, the conceptual model in [6], which was helpful in analyzing and describing the data needs of different cases of patient continuity of care among diabetes care providers, was not expressed in a known formalism. Interestingly, this non-standard model bears some resemblance to several UML models [9]. Similar to Use-case Diagrams, it shows how actors (patient, researcher, health ministry) are related to different use-cases of the EMR system. Similar to Collaboration Diagrams, it shows the relationships and some of the messages flowing between objects collaborating to perform a particular task (e.g., care providers from the district hospital can use a secured connection to access patients' data stored in the EMR via a server). The use of standard models with their rich constructs and available user manuals and guides, could help in conceptualizing and understanding the structure and communication of the system to a greater detail. Use of CASE tools could establish consistency of the specification and, if desired, could help migrate the specification into implementation code.

While developing the pharmaceutical validation and alert CDSS [10], Boussadi and colleagues realized that the standard UP system development method that uses standard UML diagrams was not enough to support all modeling activities need to develop decision rules. Instead of using ad-hoc methods for that task, or focusing on the decision rule modeling method without its relationship to the other steps involved in the system development life-cycle, the authors decided to adapt and extend the standard UP method to their needs. In this way, the modeling of the clinical logic is done via a model that is integrated with other system modeling methods within the development methodology.

Using standard (or augmented) systems development methodologies and modeling methods has its benefits, but the drawback is that they are not particularly tailored to the domain of healthcare. Several research groups have developed modeling languages and development process for clinical-guideline based DSS. Examples of such modeling languages include Asbru, EON, GLIF3, Guide, and PRO*forma* [11]. These modeling languages use conceptual models that allow modelers to specify clinical guidelines as task networks. Using such models helps in conceptualizing clinical guidelines as networks of clinical actions and decisions that unfold over time, can express clinical concepts, abstractions, and relationships, and include patient information models that aid in linking the specified guideline to EMR data. In addition, the specifications of clinical guidelines in those modeling languages are formal enough to enable them to be computer-interpretable, allowing their execution using guideline execution engines that in some cases can also link to EMRs to retrieve patient data. Based on experience in modeling using guideline modeling languages, several groups have suggested methodologies for developing the computer-interpretable guideline specifications [12, 13]. However, these methodologies, while focusing on the guideline logic, do not cover the process of eliciting the requirements and designing the front-end of a CDSS, which interacts with users and delivers advice based on the computer interpretable guideline specification; these steps that are essential for developing CDDS are best supported by existing standard system development methodologies and modeling methods [14].

References

[1] Heeks R. Health information systems: Failure, success and improvisation. Intl J Med Inform 2006;75:125-37.

[2] Hoffer JA, George JF, Valacich JS. Modern Systems Analysis and Design, 4th Edition: Addison-Wesley; 2005.

[3] Boehm B. Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall; 1981.

[4] Osheroff JA, Pifer EA, Sittig DF, Jenders RA, Teich JM. Clinical Decision Support Implementers' Workbook. Chicago: Healthcare Information and Management Systems Society; 2004.

[5] Chen P. The Entity Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems 1976;1:9-36.

[6] Kouematchoua-Tchuitcheu G, Rienhoff O. Options for Diabetes Management in Sub-Shahran Africa with an Electronic Medical Record System. Methods Inf Med 2011.

[7] Initiative BPM. Business Process Modeling Notation

(BPMN) Version 1.0. In; 2004.http://www.bpmi.org/downloads/BPMN-V1.0.pdf
[8] Oschem M, Mahler V, Prokosch HU. Objectifying User Critique: A Means of Continuous Quality Assurance for Physician Discharge Letter Composition. Methods Inf Med 2011.
[9] Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide: Addison-Wesley Longman, Inc.; 1998.
[10] Boussadi A, Bousquet C, Sabatier B, Caruba T, Korb V, Durieux P, et al. A Business rule design framework for a pharmaceutical validation and alert system. Methods Inf Med 2011.
[11] Peleg M, Tu SW, Bury J, Ciccarese P, Fox J, Greenes RA, et al. Comparing Computer-Interpretable Guideline Models: A Case-Study Approach. J Am Med Inform Assoc 2003;10(1):52-68.
[12] Shalom E, Shahar Y, Lunenfeld E, Taieb-Maimon M, Young O, Goren-Bar D, et al. The Importance of Creating an Ontology-Specific Consensus Before a Markup-Based Specification of Clinical Guidelines. Proceeding of the biennial European Conference on Artificial Intelligence (ECAI); Riva del Garda, Italy; 2006.
[13] Peleg M, Wang D, Fodor A, Keren S, Karnieli E. Lessons learned from adapting a generic narrative diabetic-foot guideline to an institutional decision-support system. In: A. Ten Teije SMaPL, editor. Computer-based Medical Guidelines and Protocols: A Primer and Current Trends, Studies in Health Technology and Informatics; 2008. p. 243-52.
[14] Peleg M, Shachak A, Wang D, Karnieli E. Using multi-perspective methodologies to study user interactions with the front-end of a guideline-based decision-support system for diabetic-foot care. International Journal of Medical Informatics 2009;78(7):482-493.