Author's response to reviews

Title: Requirements for Guidelines Systems: Implementation Challenges and Lessons from Existing Software-Engineering Efforts

Authors:

Hemant Shah (proteme@gmail.com)
Raymod D Allard (dallard2@hfhs.org)
Robert Enberg (enbergs@gmail.com)
Ganesh Krishnan (gkrishn1@hfhs.org)
Prakash M Nadkarni (prakash.nadkarni@yale.edu)
Patricia Williams (pwilliam@hfhs.org)

Version: 3 Date: 5 December 2011

Author's response to reviews: see over
Response to Review

We thank Reviewer 1 for a knowledgeable review. We do not dispute any of the suggestions, except one (flagged with two asterisks, **). Below, we reproduce the reviewer's comments in bold, and our response in plain text.

Major Compulsory Revisions

I recommend the authors to summarize those new requirements in a table grouped by main topics (knowledge representation, ambiguity, etc.).

Excellent suggestion: We have added a Table 1.

For instance, on pages 6-7, requirements about knowledge representation (KR) are analyzed. The authors identify several sub-requirements related with KR such as modularity, reusability, use of design patterns, etc. but some of these requirements are also identified in the past, such as the latter identified in works of Peleg and Tu.

We cite these references at this point (we have already cited them elsewhere). The contribution of this paper is in addressing them in greater depth: for example, the fact that Arden does not support these has not been emphasized in the literature previously: informaticians who decide to adopt Arden based on the fact that it is an ANSI /HL7 standard often get a rude shock after discovering this fact.

On page 7, the subsection named ‘Permitting Ambiguity: Parameterization’ is not well justified. I agree with the authors that the parameterization of CGs is interesting, but the section talks about two different types of parameterization that are not well distinguished. In one hand, the adaptation of a CG to local circumstances, and on the other hand, allowing the use of ‘fuzzy’ states inside CGs in order to represent a non-evidence-based state. I think that these two perspectives should be clearly identified.

Concerning the ‘parameterization’ and ‘requirements -issues 4 and 5-’ sections, in (Batet et al, 2011) we addressed the modularization and personalization of CGs to local/patient circumstances. At first, the paper distinguishes different layers of abstraction, then, it presents the concept of individual intervention plan, which are general guidelines adopted to the particular circumstances of the patient, and finally, it presents a distributed architecture to enact those sequences of medical actions over a patient during a mid or long term period.

We agree. We had originally combined them because, from the programming perspective, the same underlying mechanism (parameterization) is used, but from the perspective of the user, they are two different problems. We have now broken them up into separate sub-topics indicated by sub-headings within bullets: adaptation to local/patient circumstances, and representing non-evidence-based states within guidelines.

Thank you for the Batet et al reference: we have included and summarized it here.

Section ‘Complex Coordination: Relation with Business Workflow Systems’ is quite interesting and requires more details. The section begins with the introduction of several concepts coming from the business side. … integration, robustness and
scalability are not well explained. The Authors can use examples of using existing medical systems such as CPOEs or EHRs in order to explain how CGs can use these systems as facilities during its enactment.

We have now provided one-sentence definitions of each concept, preceding an explanation, as follows. (We have broken up "robustness" into two aspects – data persistence and error recovery.)

- **Extensibility**: the ability, for developers, to add to the capabilities of the base software through custom code written in one's programming language of choice.
- **Integration capability**: the ability to inter-operate with external software packages.
- **Scalability**: the ability to execute a large number of instances (guidelines, workflows) concurrently without noticeable performance degradation.
- **Error Recovery**: the ability to recover from hardware- or network-related error conditions.
- **Data Persistence**: The ability to create and retrieve data that outlives the software process that created it.
- **Human participation**: the ability to devise algorithms that rely on human input in a decision-making capacity.

We have added (under the Integration and Data Persistence concepts) that inter-operation with CPOEs and EHRs is essential for CGSs, and that CGSs must store patient-related data in an EHR where possible.

** Modularization and reusing existing systems have pros and cons that also should be analyzed.**

We do not believe that either of these two features have any cons, because their value has long been accepted in software engineering: the system of Batet et al uses modularization, for example. (The only possible con, the modest technical difficulty of implementation, should not be an excuse for CGS implementers: those who lack the skills to implement these features in CGSs need to hire computer scientists or IT professionals who have the skills.)

All modern programming languages support reuse of compiled object-code modules: Sun's Java team implemented the Java Native Interface (JNI) knowing that nobody was prepared to rewrite thousands (or millions) of lines of code that already worked just to be able to use Java.

All modern object-oriented languages (Java, C#, Ruby, Common LISP) support modularity. The need for modularity (in particular, for information-hiding support) was articulated as far back as 1972 by David Parnas, [http://en.wikipedia.org/wiki/David_Parnas](http://en.wikipedia.org/wiki/David_Parnas) in his seminal paper "On the Criteria to Be Used in Decomposing Systems into Modules". Fred Brooks, in the second edition of his software classic "The mythical man-month", points out one of his mistakes in the first edition--"David Parnas Was Right, and I Was Wrong About Information Hiding."

Changes made to the text: None.
Moreover, one of the items identified is particularly critical: the participation of humans during the process. In this sense, several papers talk about a general reticence of practitioners to use guideline-based systems in daily care.

Thank you. We agree that there are numerous papers expressing this concern. We have cited a reference by Brian Hurwitz, British Medical Journal 1999 "Legal and political considerations of clinical practice guidelines", which describes the reluctance of clinicians to use guidelines because of a fear that the clinician is not in charge. Business-workflows must similarly be often designed to accommodate highly fluid circumstances and allow an executive to make a decision and override the standard workflow.

On page 15, section ‘Implications of Longer Duration of Execution’ should be improved. The authors introduce the problem but the solutions proposed by business-workflow engines are not well detailed. Persistence is a technique to maintain the correct state of data. The authors should include more examples explaining how persistence could solve the described requirement. In addition, the second situation (‘A sophisticated version-aware runtime engine’) should be explained.

We agree with the reviewer, but are conflicted regarding how much detail we must provide to a readership that may not have computer-science expertise. For example, while it is easy to "persist" data per se, the representation of the stored data structures, in circumstances where version control must be supported, is necessarily much more complex than for one that does not require version control. Similarly, the stored representation for an instance that may run for a year (during which time the machine may have been rebooted several times as part of operating system upgrades, for example) is more complex than that for one which is guaranteed to complete within a single interactive user/session: the latter might only require a simple audit trail, along with storage of altered patient data in the EHR.

Explaining the details of the knowledge representation (which, in any case, is proprietary to individual workflow engines) might require describing the schema of a system where this is publicly documented (e.g., Windows Workflow Foundation).

In an attempt to balance the need to provide explanation versus the requirement not to overwhelm the reader with detail, we have re-titled this section as "Support for Multiple Guideline Versions", and modified the text as reproduced below. The first two paragraphs are the same as in the previous version of the paper, the rest has changed: we have also provided two new references for the reader who wishes to learn more. We invite feedback from the reviewer in case this text is still confusing.

"Unlike most decision-support systems, a patient going through a guideline workflow may stay within that workflow for long periods. For example, a workflow related to management of infertility would last more than a year, because pregnancy takes time to establish. The clinical aspects of the workflow may no longer apply (e.g., because of the availability of a new and better treatment, recommendations may have changed), and so a new clinical sub-workflow may supersede the old one.

"However, workflows also have administrative aspects (inventory adjustment, service itemization, reimbursement from the insurer/payer), whose management can be more complex: for different patients, multiple versions of the same administrative sub-
workflow may run simultaneously. For example, even if a reimbursement agreement has changed since the patient has begun treatment, the contract with the payer may require the old agreement to stay in force.

"Therefore, depending on the nature of the guideline or workflow, the execution engine may need to enforce one of two strategies:

- Replace the old version of a guideline/workflow with the current version for all instances/patients (This is technically challenging because many patients are in the middle of a guideline or flow, and there must be rules that specify what is to be done if, for example, a patient is in a guideline branch that is now obsolete.)
- Allow multiple versions of the same broad guideline/workflow to coexist and run at the same time, the choice of version depending on the patient.

"The knowledge representation employed by advanced business-workflow engines supports hierarchical versioning, with an individual version represented as a “child” of a base version, and differences between the base and child modeled as atomic change units analogous to the “deltas” used in software version control [40], along with metadata at the base level that specifies which of the above strategies is to be followed if changes occur. Every active instance is associated with its current version, and its current position/branch within the version, and the path taken to get to this position. The runtime execution engine will implement the desired strategy for a given instance if changes occur to the guideline. While most engines use proprietary approaches, there are attempts, e.g., Juric et al [41], to apply such techniques to standard representation languages such as WS-BPEL."

**Conclusions should be improved drastically summarizing the main (new) requirements identified by the authors. It is important to note the contributions of this paper towards the widely adoption of these type of systems in daily care, assuming the advantages provided to practitioners and patients.**

Rather than recapitulating the new table 1, we refer to it, and note our contribution. The first sentence of the original paper has been expanded as follows:

"The contribution of this paper is two-fold:

- It discusses in detail the sub-requirements that become critical during implementation of production clinical guideline systems, in accordance with the framework outlined in Table 1. These have not been previously discussed in the literature to the requisite depth needed by production-CGS implementers.
- It cites practical issues from a closely related, but far more mature domain, that of business-workflow engines, whose implementers have encountered, and often solved, many of the problems that CGS implementers have yet to encounter because the latter have not attempted to create industrial-strength systems that are used in daily patient care. Production-capable CGSs must demonstrate the same robustness as the CPOE and EHR software with which they are required to inter-operate."

**Minor Essential Revisions**
In (Ongenae et al, 2010) a rule-based guideline-based execution engine using Drools is introduced. It could be added to the ‘Requirements’ section because it is a good example that combines medical ontologies, standard representation languages, and a complete case study.

Sweidan et al, (2011) deals with evaluation measures (safety and quality) associated with clinical software that could be adopted by guideline-based systems.

Discretionary Revisions

I have a formal doubt about how to add references to tools or URLs. The paper combines two ways: including the URL in the text (inline), and adding a reference at the end of the document (outline). Please, revise the ‘guide for authors’ of the journal to harmonize them.

Thank you: we have cited Ongenae et al at the end of section 3.6, and Sweidan et al in the Introduction.

We have placed all references in the bibliography and removed inline URLs.