Abstract. Governments, insurance organizations, hospital boards, physicians and patient organizations support the relevance of rigorous engineering methods for the development and certification of information systems in health care. This paper leverages model-driven engineering techniques to improve (the use of) health information systems that are process-oriented (e.g., clinical decision support systems and workflow components in electronic health record systems). We focus further on model-driven techniques since they enable the separation of medical and organizational concepts from system implementation details. This is important since information system architectures vary significantly within and between care institutions while at the conceptual level patients cross the institutional and system boundaries. To set a baseline, we derive from literature a simple classification framework for process-oriented models in healthcare. We then demonstrate how that theoretical baseline has triggered the development of a practical tool-chain for making the development of decision support apps more evidence-based. We discuss where current MDE techniques have supported us most such that practitioners can focus on these techniques but also suggest to MDE researchers where additional support is needed in practice.

1 Introduction

Model-Driven Engineering (MDE) is a software engineering method in which explicit modeling language and model transformation definitions are used to generate or configure powerful tools. As in most other engineering disciplines, models in MDE are simplified representations that enable one to reason more easily about complex issues. At the 2002 Integrated Formal Methods conference, Kent underlined that more sophisticated MDE tools were needed than the commonly available “model editors” [1]. Among others, he called for tool support for heterogeneous modeling (e.g., editors that can handle multiple languages), he called for model transformation tools, conformance checking tools and version management tools.
At the time of writing, that is ten years later, the FHIES symposium calls for solution-domain papers that present the potential of particular engineering methods for healthcare. Moreover, FHIES 2012 calls for problem-domain papers to educate the methods community about the unique challenges and characteristics of the health care domain. In this paper, we show the potential of MDE technology for improving workflow management and decision support in healthcare. To understand better the nature of models in this context, we have systematically analyzed the English and Dutch literature on care planning and execution. Our analysis results are presented succinctly via a layered, two-dimensional classification that identifies clinical guidelines, clinical protocols, (reference) care pathways, individual care pathways and assigned pathways as key candidates for the model-driven development and analysis of health information systems (HISs). Our discussion of the classification clarifies among others which stakeholders work with these models and discusses among others “instantiation” and “specialization” relations between these models. We use the classification as a theory to structure our research roadmap for MDE in healthcare. To illustrate the applicability of this theory, we discuss an initial MDE prototype that we have developed at one specific level of the classification. The prototype shows on the one hand the potential of MDE for healthcare, inviting health IT professionals to engage on similar projects. On the other hand, it gives direction to MDE researchers that wish to validate and perhaps specialize generic model management techniques.

The remainder of this paper is structured as follows: Section 2 briefly describes related work, Section 3 provides the theory of this paper: it presents the classification that we use to put in perspective our practical MDE contribution from Section 4. Section 5 makes the applicability of that contribution more tangible to healthcare professionals with an example. Finally, Section 6 concludes.

2 Related Work

Terminology for process-oriented healthcare is used rather confusingly both in medical and in information systems literature: different terms are used interchangeably, and the same term may have different meanings. Therefore, the authors have first surveyed and classified the related literature in order to understand similarities, differences and overlap between terms. The literature study was conducted conducted by the student co-authors of this paper and was validated and refined by the other co-authors (two information systems researchers and a neurosurgeon). This paper will only present the synthesis of our findings; the complete literature study results will be published in a separate paper. In the meanwhile, the interested reader is advised to consult the survey of pathway concepts by Vanhaecht et al. [2] as well as the process-oriented information systems survey by Gooch et al. [3]. Vanhaecht et al. have inspired our new classification scheme while Gooch et al. suggest that model-driven tools are urgently needed for bridging the gap between theory in practice for process-oriented, patient-centered, HISs.
In addition to this overview and classification of terminology, we have also looked into available literature for model driven decision support tools in process-oriented healthcare. For this we could only find a few papers. Most of the available studies focus on capturing the textual descriptions in a formal model \[4,5,6,7\], but do not elaborate on the tool support based on this formal model. In this paper we specifically focus on the tool supported derivation of the formal model and on the generation of a light-weight Clinical Decision Support System (an “app” CDSS) based on the formal model of a clinical guideline. Evidence is emerging that handheld clinical decision support tools improve adherence to guidelines for diagnosis \[8\]. Studies also suggest that they lead to more appropriate drug prescriptions \[9\]. Our study supports the systematic derivation of guideline models from medical literature, the potential discussion, refinement and linkage of guideline models and the co-evolution of guidelines and derived CDSSs. To the best of our knowledge, the use of MDE techniques has not yet been explored. Nonetheless, some isolated engineering efforts have been published. For example, Lobach et al. \[10\] describe a systematic process for translating informal guidelines into computer-interpretable representations.

3 A Classification of Models for Process-Oriented HISs

Our ultimate goal is to improve (the use of) process-oriented HISs by means of MDE techniques. In this paper, we demonstrate the result of an initial step in this direction: we show that through a structured approach an annotated guideline can be transformed into a decision support application based on a flow chart. In this section, we share the result of the literature study that has driven our first steps and that is guiding our ongoing efforts. We have classified all process-oriented models described in literature along two dimensions:

D1 (Patient Scope) The first dimension involves the scope of the description from the patient perspective: the most coarse grained descriptions aim at any type of patient, regardless of care groups. Other descriptions aim at multiple patients but within one specific care group. Finally, some descriptions are specific to an individual patient.

D2 (Provider Scope) In the provider aggregation dimension, some descriptions aim at any organization that delivers a particular type of care. Such descriptions tend to be subject to (inter-)national review processes. Example review organizations are the US Agency for Healthcare Research and Quality (AHRQ) and the Dutch Centraal BegeleidingsOrgan (CBO)\(^1\). Other descriptions aim only at those organizations that are part of a cooperation network. A Dutch cooperation network that among others aims at improving the efficiency of care delivery is the Eye Care Network\(^2\). Even other descriptions are oriented at one specific care organization. At the finest level of aggregation, some descriptions are oriented at just one caregiver.

---


2 See [http://www.oogzorgnetwerk.nl/](http://www.oogzorgnetwerk.nl/)
Descriptions can also be classified by analyzing whether they have a medical versus a logistic/administrative focus but especially since descriptions become increasingly hybrid along this dimension, we consider only the above two dimensions. Our literature survey has resulted in the following list of descriptions related to process-oriented health information systems:

**Clinical Guidelines** (CGs) are used by medical professionals for grounded, evidence-based decision making in the diagnosis and treatment of patients. Our literature survey has resulted in the following definition: “A guideline is a recommended way of working that describes what tasks can be done by clinicians in different organizations for specific clinical circumstances in order to support the decision making process.” Various terms with the same meaning exist: medical guideline, clinical practice guideline, evidence-based guideline, etc. We stick to **clinical guidelines** since is the most commonly used term [11].

**Clinical Protocols** (CPRs) describe what and how should be done for a specific issue within an organization by clinicians, based on (clinical) guidelines designed to standardize the patient care [11]. Since protocols are developed and certified by various cooperating specialists with authority in a care institution, individual caregivers need to carefully motivate protocol deviations. In contrast, clinical guideline adherence control (if any) is typically much less formalized.

**Care Pathways** (CPAs) are less general than clinical guidelines since they focus on a certain group of patients and contain organization-specific information on capacity and possibilities in the organization (e.g., equipment to be used for the treatment, who should do what, etc.) Our literature survey has resulted in the following definition: “A care pathway is a description of a care process from an organization point of view for a specific disease and for a specific group of patients. It is based on evidence and on (clinical) guidelines and it is designed to improve efficiency and patient outcomes.” Various terms with a similar meaning are in widespread use: nursing care pathways, integrated care pathways, critical pathways, clinical pathways, etc. but we agree with Vanhaecht et al.’s arguments that **care pathway** is a preferable term [2].

**Individual Care Pathways** (ICPs) are patient-oriented specializations of care pathways [11]. The specialization can be based upon pruning out activities that relate to patients of a different age. Unlike what is common in a programming context, this specialization of pathways may also involve unstructured additions or removals of activities from the pathway.

**Assigned Pathways** (APs) are instantiations individual care pathways. The instantiation involves (non exclusively) setting dates and locations. While such information may be automatically initialized at instantiation time, it may be changed arbitrarily over time (to keep the assigned pathway in sync with unanticipated events in the real world). For uncompleted activities, an assigned pathway is a prospective model whereas for completed events it is a retrospective model.
Reference Pathways (RPAs) are generalizations of care pathways. Vanhaecht et al. suggested the use of such a concept to refer to those pathway descriptions that characterize the common aspects of care pathways that are used at different organizations. Unlike the previous concepts, reference pathways are largely a theoretical concept today, since in practice there is very little formal reuse in the area of care pathway modeling.

Most descriptions of the aforementioned types have besides their native, caregiver-oriented representation also patient-oriented views. ICP and AP views are often communicated via patient brochures and care portals. Also, patient-oriented views upon CGs are used (among others) reduce societal costs by providing boilerplate advice online. It is obviously challenging to keep patient-oriented advice (e.g., [http://www.patient.co.uk/guidelines.asp](http://www.patient.co.uk/guidelines.asp)) consistent with the underlying guidelines. Similar challenges have already been discussed in MDE literature (e.g., [12]).

Fig. 1. 2D Classification of Process Oriented Care Descriptions

Fig. 1 shows that these descriptions can indeed be classified in the proposed 2-dimensional space “patient scope x provider scope”. The provider dimension may require some disambiguation, since it can apply to:

1. multiple organizations offering collaborative care through operational care networks, but also to
2. multiple (groups of) organizations sharing care process descriptions at the regional, national or international level.

Remarkably, the aforementioned six types of process descriptions apply to intra-organizational care as well as to inter-organizational (also called, transmural) care. Therefore, operational care networks are treated as single (virtual) organizations in our classification. Put differently, we use the second interpretation from the above list.

Also, it turns out that there are no specific names for distinguishing process models that are specializations of those that are used at the level of their organization. Concluding: the provider scope dimension only distinguishes between process models that are used within one (virtual) organization and those that are shared by multiple (virtual) organizations.

The classification is then as follows: when referring to the concrete goals and activities for one patient within one organization (=1, =1), one is considering Individual Care Pathways (ICPs) and Assigned Pathways (APs). When referring to the process descriptions for a group of patients within one organization (>...
In group, =1), one is considering care pathways (CPAs). In contrast, when for such a group of patients one is referring to an abstract process description that is shared by multiple organizations (> $1_{in\,group, >1}$) then one is considering reference pathways (RPAs). In the context of decision support for patients in general (regardless of groups), care protocols (CPRs) are descriptions used within organizational boundaries (> $1_{in\,general, =1}$) while clinical guidelines (CGs) are used beyond these boundaries (> $1_{in\,general, >1}$). The critical reader may wonder why there is no term in the classification cell (=1, >1). The cell represents process descriptions that are specific to the care of one patient yet used as a reference by organizations outside the (virtual) organization that has delivered that care. The cell is empty since such descriptions have no commonly used name in literature.

We have engaged in investigating metamodel-based language support for each class of process descriptions. This will enable the model-based analysis of the descriptions. It will also enable the development of model-driven transformation and traceability techniques in that context, which should ultimately lead to better managed health information systems.

The remainder of this paper focuses on our first practical result from this theory: we focus on the application of model-driven engineering techniques for managing better the relation between clinical guidelines, clinical protocols and their derived applications. This is a particularly interesting relation since it involves not only computerized transformations but also consensus building by various medical specialists. The medical literature often relies on flowcharts for documenting guidelines. In the following, we demonstrate that by strengthening the basis for these modeling initiatives, we enable new opportunities. Among others, we demonstrate how smartphone applications can be derived efficiently (i.e., with little effort) and effectively (e.g., with formal traceability) from guidelines.

4 MDE Support for Clinical Guidelines

This section first motivates the relevance of MDE for CGs, then outlines how we have enabled an applied MDE support for CGs. Finally, we describe where MDE tool frameworks could not adequately support us yet.

4.1 Relevance of MDE Support

Fig. 2 extends Fig. 1 by showing time as the third dimension of our classification. We include time to clarify the need for formally inter-connecting the models from our 2D space: $AP_{1v1}$ in Fig. 2 represents a patient-assigned pathway at a given point in time. $AP_{1v1}$ is an instance of individual care program $ICP_{1v1}$. It is important to represent the instantiation relation formally since the individual care program may evolve to a new version $ICP_{1v2}$. This new version may for example include an additional X-Ray test. Clearly, the dates and locations in the assigned pathway then need to co-evolve, leading to $AP_{1v2}$. An assigned pathway instance can also evolve independently of the individual care pathway:
for example, a date can be changed without affecting the ICP: $AP_{1v3}$ is still an instance of $ICP_{1v2}$.

In the scenario shown on Fig. 2, both ICPs specialize the same care pathway $CPA_{1v1}$. It is essential to store different versions of the CPA since otherwise it becomes impossible to reason about the adherence of ICPs to the corresponding CPAs over time. $CPA_{1v1}$ specializes two reference pathways in our example. Clearly, reference pathways may evolve over time. By formalizing CPAs and RPAs as models and by storing RPAs in versioned repositories, one can provide automated support for deriving and evolving reference pathway models.

Fig. 2 also shows a delegation link from care pathway $CPA_{1v1}$ to clinical protocol $CPR_{1v1}$, which in turn specializes two clinical guidelines $CG_{1v1}$ and $CG_{2v1}$. These delegation and specialization relationships are not yet explicit in today’s health information systems. In general, the theoretical instantiation, specialization, and update-of links from Fig. 2 are largely implicit in today’s health information system architectures. By making them explicit, one could better analyze how evidence-based descriptions of optimal medical care (CGs) relate to nurse management systems (CPRs) and patient logistic systems (CPAs and ICPs) and planning systems (APs). This first of all calls for metamodels that enable the formal representation of all artifacts in this space. Once metamodels are in place, links between the models and their individual elements can be provided conveniently.

### 4.2 Enabling MDE Support: Metamodel Definition

This subsection describes the fundamental steps that are needed to enable MDE support for any cell in the aforementioned space. We have carried out these steps only for clinical guidelines but tackle other cells in our ongoing and future work.

The first step in enabling MDE support is analyzing which modeling languages are relevant to the problems at hand. Clinical guidelines tend to be documented in plain text but in many cases they are also formalized using the flowchart notation. Unfortunately, the flowchart models are not published as primary artifacts, instead they are images in medical papers. Our analysis of such papers indicated that only a subset of the flowchart notation was used in many cases (i.e., activity and decision nodes). Moreover, we identified the
need to enrich the model elements with additional metadata. In the following, we demonstrate how we have used the Eclipse Epsilon suite (Eugenia in particular) to define the abstract and concrete syntax of our newly developed flowchart-based language for CG modeling.

```xml
@namespace (uri="http://tue.nl", prefix="tue.flowchart")
package flowchart;
@gmf.diagram(model.extension="flowchart")
class Flowchart {
  attr String flowchartName;
  val Node[*] nodes;
}
@gmf.node(label="name", label.icon="false")
abstract class Node {
  attr String name;
  attr String additionalInfo;
  attr String[*] referencePapers;
  ref Action[*]#next rnext;
  ref Decision[*]#pos rpos;
  ref Decision[*]#neg rneg;
}
@gmf.node(figure="rectangle")
class Action extends Node {
  attr String[1] action;
  @gmf.link(target.decoration="arrow", tool.name = "next")
  ref Node[0..1]#next next;
}
@gmf.node(figure="figures.Rhomb")
class Decision extends Node {
  attr String[1] question;
  @gmf.link(target.decoration="arrow", color="0,255,0")
  ref Node[1]#pos pos;
  @gmf.link(target.decoration="arrow", color="255,0,0")
  ref Node[1]#neg neg;
}
```

Fig. 3. Abstract and Concrete Syntax Definition for a CG-oriented Flowchart language.

Fig. 3 shows that the complete language definition requires just 30 lines of Eugenia code. The abstract syntax (i.e., the metamodel structure) is defined by classes and associations (references). Our flowchart language consists of the aforementioned two types of nodes (cfr., classes Action and Decision). Shared node attributes are inherited from a superclass Node. The multi-valued string attribute on line 12 enables to associate one or more medical paper to a flowchart node. This is useful for traceability between a CG model and its corresponding evidence.

The concrete syntax is defined by means of annotations (marked by the "@" symbol). For example, the annotations on lines 26 and 28 specify that for a Decision node the outgoing pos and neg references should be visualized by green and red arcs respectively. The purpose of these two edge types is the modeling of CG decisions based on binary decisions.
4.3 Using MDE Techniques: Generating Editor and Apps

Based on the preparation step from the previous subsection, we can put MDE techniques to action. First of all, we can generate a special purpose flowchart editor based on the definition from Fig. 3. Fig. 4 shows a screenshot of an example use of this editor. The left pane shows the editor palette, which enables the instantiation of the concepts from the syntax definition. The middle pane shows an example flowchart diagram. In the screenshot, the “Hematoma” node from the upper left is selected and its details are shown in the rightmost editor pane. That pane enables among others associating reference papers (medical evidence) to the node.

![Example Use of the MDE-based Clinical Guideline Editor Prototype.](image)

The above editor instantiates models in such a format (i.e., that of the Eclipse Modeling Framework) that they can be seemlessly processed by other special-purpose MDE tools. More specifically, various model transformation and verification tools (cfr., [http://www.eclipse.org/modeling/](http://www.eclipse.org/modeling/)) are seemlessly interoperable with the editor. As a proof of concept, we have developed a prototypical code generator for translating the flowchart models into source code files for mobile Android devices. The goal of this prototype was to demonstrate the potential of MDE techniques for the more rigorous development of mobile apps. In particular, the last author of this paper has been successfully developing mobile decision support apps using conventional programming techniques. As a neurosurgeon, this author already created flowchart models using a freehand drawing tool and then wrote conventional code for specific APIs [14]. Two of our three student co-authors have developed a prototypical Java-based code generator to generate the API-specific code seemlessly from the flowchart models. This case forms a nice basis for benchmarking the strengths and weaknesses of the wide array of available MDE code generation suites but this elaboration is still future work.
4.4 Gap: the need for a model-based text annotation framework

As indicated in section 4.1, our hypothesis is that MDE techniques can primarily contribute to the better management of related artifacts over time. Clearly, the special purpose, meta-model based editor discussed in the previous subsection provides a promising basis for storing CG models in a shared repository. In such a repository, CG models could be linked, discussed, updated automatically in the context of co-evolution, etc. Also, derived apps can be automatically regenerated as the CG models evolve. However, one crucial step has been overlooked in the previous sections: the primary publication artifact of a clinical guideline is still its related medical paper.

Our prototypical CG tool suite takes that into account by offering a component to annotate medical texts (scientific papers, localized guideline descriptions, etc.) Based on manually annotated texts, a flowchart skeleton can be derived automatically and links between text and model can be stored for traceability purposes. Unfortunately, we could not find generic MDE tool infrastructure to ease the development of the text annotation component. Therefore, we have implemented the annotation component as an ad-hoc Java Swing application and we call the MDE community to action: generic support for building interactive text to model derivation tools is urgently needed.

We emphasize that grammar-based automatic text-to-model transformation tools are largely irrelevant in this model mining context since the input texts do not adhere to grammatical rules. Grammar-based tools can play a meaningful role once structured guideline texts are available but we have focused on the more realistic case.

5 Elaboration for Healthcare Professionals

Fig. 5 shows our prototypical tool-chain from an end-user perspective: first, medical specialists annotate scientific articles. This can happen in the context of their personal continuous learning process or in the context of regularly planned guideline review cycles within a hospital. In this step, annotations should be stored in a computer-interpretable form. Second, the guideline annotations are transformed into a flowchart skeleton model. Third, the flowchart is manually refined. Finally, the flowchart is transformed into a CDSS app. Configuration files
for a more heavyweight CDSS could be generated too but this is not implemented at the time of writing. In the following, we demonstrate all implemented steps for one specific clinical guideline.

The selected guideline is that for surgical management of depressed cranial fractures (cfr., Fig. 6). The guideline is supported by the Congress of Neurological Surgeons (CNS) and the American Association of Neurological Surgeons (AANS). It was selected because it is rather concise, clear and structured, and because it contains enough complexity to show how the tool-chain works. Our tool-chain works for any guideline containing (i) a title, (ii) observations, (iii) recommended actions, and (iv) explanations (i.e., supportive arguments).

Fig. 6. Example of a Clinical Guideline recommendation (summary based on [15]).

5.1 System Walkthrough

Fig. 7 shows the text from Fig. 6 within the annotation tool. The title annotation is shown in green. The parts of the text that are considered observations have been annotated in yellow, the actions/treatments are shown in red, and the explanatory elements are shown in blue. The bottom left of Fig. 6 shows controls for creating new annotations while the top left shows a tree preview of the
guideline model that is under construction. By clicking the *compile* button, this representation is translated into the format of our flowchart editor.

![Flowchart Editor Screenshot]

Fig. 7. Demonstration of support for eliciting model elements using text annotations.

Fig. 8 shows the final flowchart CG model (screenshot taken directly from the Eclipse editor). The edges in the figure have been created manually. Note also that explanations are not shown in the current version of the flowchart editor. Clearly, they can be added easily by adapting the annotations in the concrete syntax definition of the editor (cfr., Fig. 3).

Fig. 9 demonstrates how the generated app realizes the behavior of this flowchart: Subfigure 9(a) shows how initially the app presents a searchable list of guidelines (derived from the title attribute of the flowchart models). For this demonstrator, there is just one guideline. Subfigure 9(b) shows the app after selecting our example guideline. The system prompts the question that corresponds to the root node of the decision tree. In our example scenario, the user answers respectively “yes”, “no”, “yes” (cfr., the “on”, “off”, “on” buttons) on the series of questions shown on Subfigures 9(b) to 9(d). Conform to the flowchart, this leads to the suggestion that there is evidence in favor of performing an early operation (Subfigure 9(e)).
Fig. 8. Example of a derived flowchart

(a) Select CG  (b) Q&A 1  (c) Q&A 2  (d) Q&A 3  (e) Suggestion

Fig. 9. Example execution of the generated app running on an Android smartphone.

5.2 Evaluation and Identification of Future Work

The implementation of the prototype has confirmed our confidence in the potential of MDE techniques for the development of better process-oriented health information systems: the full tool-chain has been implemented by two undergraduate students, who have spent about five ECTS each. The students did not have any prior MDE expertise but did master Java programming already. Students did receive guidance by one MDE expert, primarily in the use of Eugenia (cfr., Fig. 3). In the following, we first highlight some points for improvement that can make the prototype a stronger asset for research in a Health Information Technology (Health IT) setting, then we identify MDE related research issues that still relate specifically to clinical guideline support. Finally, we discuss strengths, weaknesses and opportunities in a more general setting.

Validation of the prototype with the medical specialist has learned us that the simplistic flowchart model as well as the simplistic code generator lead to usability problems for the generated mobile app. More specifically, since the flowchart variant only offers stepwise binary decisions the associated mobile user interface can produce annoyingly long sequences of binary questions. For example, the user interface derived from the flowchart shown on Fig. 8 can prompt up to 9 questions before advising non-operative management. By either extending the expressiveness of the modeling language or by making the code generator more...
advanced, one can merge all these questions on just one form with 9 questions. On the other hand, some users may find it disturbing that for other scenarios this means considering many irrelevant questions. Further analysis of related Human Computer Interaction (HCI) literature seems appropriate before drawing firm modeling language and code generation requirements from this example. Regardless of these HCI aspects, one should also further investigate to what extent the derivation of model skeletons from annotated texts can be automated further.

The prototype architecture can be strengthened from an MDE research point of view by re-developing the code generation step using an advanced code generation framework. Interestingly, we had identified Acceleo as an interesting support platform for the student project since Acceleo already provides advanced support for mobile device code generation [10]. Moreover, Acceleo provides unique model-to-code traceability views. Unfortunately, our students could not overcome the learning curve of the Acceleo platform and have chosen to implement the code generation in plain Java. A deeper analysis of the learning curves, quality advantages and longer-term developer efficiency forms an interesting research opportunity for the MDE community.

The development of the CG-specific prototype has also produced general insights for further research on the model-driven development and analysis of process-oriented information systems. In general, we point out that the probably strongest point of using MDE techniques has not yet been exploited: in our opinion, we should further explore the opportunities for automated analyses and transformations on the CG models. For example, we aim to enforce some (perhaps OCL-based) well-formedness rules (e.g., “the flowchart should have exactly one root node”). Also, we aim to exploit the opportunities for defining model transformations to various CG formalisms such as GLIF, Gaston, SAGE and others [11]. Moreover, it should be explored how CG models can be meaningfully stored in shared repositories. Among others, we should learn from building and using a demonstrator where CG models are linked to CPR, CPA, RPA and ICP models to fully exploit the richness in meta-data and information structuredness. A weakness of the current version of the Eugenia infrastructure as well as our ad-hoc text annotation tool is that they do not support collaborative editing (of models and text annotations respectively). For Eugenia, this weakness is not fundamental since the infrastructure can be extended without affecting artifacts such as the one shown in Fig. 3. Our ad-hoc annotation tool however should be completely revisited. In our ongoing efforts, we are considering which research communities may already have developed useful infrastructure for collaborative modeling and text annotation (ideally also on mobile devices). We aim to evaluate such infrastructure in the full breath of our classification.

See http://www.openclinical.org/gmmsummaries.html for an overview.
6 Conclusions

This paper presented a clarification and novel classification of the existing process-like descriptions in the healthcare domain in order to derive support for these processes through model-driven engineering techniques. We have identified two different dimensions to distinguish the various types of descriptions. For one of the types in our classification, clinical guidelines, we have developed a tool-chain to illustrate how MDE techniques can enable the stepwise development of mobile clinical decision support apps.

Ten years ago, Kent called for building more advanced MDE tool support. This paper has demonstrated that contemporary tool support enables non-experts to build metamodel-based clinical guideline model editors that easily interoperate with generic tools for advanced model management tasks such as automatic verification and transformation. This means that the MDE community has managed to align its tool development initiatives. We have also experienced a previously undocumented need for a new type of tool: since many medical artifacts consist of detailed yet unstructured text, MDE infrastructure for manually defining text annotations is needed. Such infrastructure should support online (collaborative) text annotation in a format that is compatible with existing MDE tools. In this paper, we have presented an ad-hoc implementation of such an annotation tool and we have demonstrated its relevance for interactively extracting clinical guideline models from the enormous body of medical evidence (texts from research papers).

We plan to conduct similar practical MDE experiments for all sections our classification. This should lead to metamodel-based and linked repositories of process-oriented models that can be leveraged for the more systematic development and analysis of health information systems. More specifically, such repositories should support the automated co-evolution of the various process-oriented artifacts over time. Also, these repositories should provide automatic transformations to specific formats, as imposed by human stakeholders, operational execution infrastructure (heterogeneous workflow and CDSS systems) and existing analysis infrastructure.

Acknowledgements

We would like to thank Rob Vanwersch, Ronny Mans and Simone van der Geer for their help and explanation in the medical domain and their valuable feedback on this work. Moreover, we would like to thank Ivo Deen and Joerik de Ruijter for their contributions to the literature survey.

References


