Acadela: A Domain-Specific Language for Modeling Clinical Pathways

Tri Huynh, Selin Erdem, Felix Eckert and Florian Matthes
School of Computation, Information and Technology, Technical University of Munich, Munich, Germany

Keywords: Domain-Specific Language, Domain-Specific Modeling Language, Clinical Pathway Modeling, Adaptive Case Management, Decision Support Tool.

Abstract: e-Health systems leverage clinical pathways (CPs) models as standardized and optimized procedures to execute and manage medical treatments. To model CPs in decision support e-Health systems, our study develops Acadela, a low-tech-oriented, text-based Domain Specific Language (DSL) with visualization capability. Acadela declares grammar to enforce textual syntax for modeling workflow, control flow, responsibility, medical data visualization, and communications with external systems. Furthermore, Acadela provides a model visualization to preview the CP and assist communication between medical and technical experts. To explore the DSL's expressiveness and usability, we conducted two separate descriptive user studies with six medical professionals and eight technical adepts. First, we modeled five CPs used by medical professionals in their daily routines. Through semi-structured interviews, we collected feedback regarding the language's expressiveness. Next, we invited the technical adepts to model a hypertension CP and debug a faulty model written in Acadela. Overall, the medical experts consider the modeled CPs accurately reflect their treatment procedure; and the technical adepts consider the language easy to use and applicable to model CPs. The results imply the DSL's potential to model CPs with various degrees of complexity in different medical fields while being user-friendly to modelers.

1 INTRODUCTION

E-Health applications are systems that apply a combination of electronic communication and information technology to transmit, store and retrieve digital data electronically for educational, clinical, or administrative purposes in the healthcare sector, both locally and remotely (Mitchell, 1999). Operating treatment processes for patients in e-Health applications requires modeling clinical pathways (CPs). In summary, CPs are patient-care management plans that define care goals with the process and timing necessary to achieve such goals with optimal efficiency (Panella et al., 2003; Campbell et al., 1998; Every et al., 2000). Applying CPs results in numerous benefits, from improving patient conditions (Panella et al., 2008), reducing complications (Preston et al., 2013), mortality (Hai et al., 2019), length of stay, and costs (Yang and Su, 2014). State-of-the-art solutions propose graphical or textual Domain Specific Languages (DSLs) to model CPs and leverage their benefits in e-Health systems. However, each approach has distinct advantages and limitations.

On the one hand, graphical DSLs enable modelers to build medical procedures by interacting with visual elements. The advantages of this approach are a user-friendly interface and learnable mechanism (Hermans et al., 2009, p. 433), as the applications display workflow elements following a logical and hierarchical structure (Wienands and Golm, 2009, p. 458). Furthermore, visual artifacts foster communication among technical and domain experts (Heß et al., 2015, p. 13), as they share a common understanding of notations for CP elements. However, existing workflow modeling languages, like BPMN, lack the notation to express aspects besides workflow execution, thus demanding extra effort to develop custom extensions for modeling and visualizing these aspects (Heß et al., 2015, p. 4), (Braun et al., 2014, p. 10).

On the other hand, textual DSLs model CPs using a text-based interface and grammars (Cook et al., 2007, pp. 15-17) or meta-models (Jouault et al., 2006, p. 2). The textual model representation combined with an IDE brings several benefits. First, extending the model is convenient by textually defining sub-DSLs of new domain concepts (Rieger et al., 2018). Second, validating models is manageable as textual DSLs only define constraints via grammar (Baar, 2015). Third, an IDE offers convenient support

Huynh, T., Erdem, S., Eckert, F. and Matthes, F.
Acadela: A Domain-Specific Language for Modeling Clinical Pathways.
DOI: 10.5220/0012076500003538
In *Proceedings of the 18th International Conference on Software Technologies (ICSOFT 2023)*, pages 46-58
ISBN: 978-989-758-665-1; ISSN: 2184-2833
Copyright © 2023 by SCITEPRESS – Science and Technology Publications, Lda. Under CC license (CC BY-NC-ND 4.0)

with *error warning*, *syntax highlighting*, and *auto-completion* of CP elements and their values (Cook et al., 2007, pp. 13-14), (Merkle, 2010). However, textual DSLs typically offer no CP visualization, which is essential to present a comprehensive overview using graphical notations (Frank, 2013, p. 134) familiar to both domain and technical experts, thus *fostering communication* and *workflow analysis* to increase the quality of the treatment.

Furthermore, from literature research, existing DSLs for CPs do not support the modeling of coordination with external systems or customizing the visual representation of medical data. External communication is vital to obtain decision-support data (e.g., adverse drug-drug interaction effects (Thakrar et al., 2007)), orchestrate and integrate with existing (legacy) systems (White, 2009, p. 11) (Kurz et al., 2015, p. 4) (Iroju et al., 2013, pp. 265-268), as they require necessary data (e.g., user information, medical status) from these systems to execute healthcare services for each patient case. Meanwhile, customizing data representation provides flexibility in displaying medical data (e.g., as a graph, image, or decorated text (Michel, 2020, p. 36)) to express the patient's condition comprehensively and vividly. An understandable visualization is prominent in medical treatment as they provide healthcare professionals with an overarching and insightful overview of medical states to ease decision-making, particularly in complicated scenarios (Suganthi and Poongodi, 2021).

In addition to the technical functionality, a DSL should be *user-friendly* to modelers. Therefore, we also explore user-friendliness and learnability from the modeler's viewpoint in our study.

To address the missing essential CP concepts and study the usability of our solution, we develop a generic, textual DSL named Acadela for CP modeling. Besides declaring essential CP elements, Acadela can 1) define when and how to send HTTP requests to external systems, 2) customize the graphical representation of medical data in the e-Health application, and 3) visualize the modeled CP using basic shapes and color code. Our goal is to integrate Acadela into e-Health applications without drastically modifying the system while increasing the usability and productivity of the CP modeling process.

The remainder of our paper consists of seven sections. First, we explained the research methodology applied in our study. The next section presents the related work of existing DSLs that identify the required concepts and features for modeling CPs discussed in the fourth section. The fifth section showcases concrete syntax to model the identified CP concepts in our DSL. The sixth and seventh sections report our descriptive user studies regarding the expressiveness and usability of the DSL, respectively. The final section concludes our study and proposes future work.

2 METHODOLOGY

To envision the development and evaluation of our DSL, we formulate three research questions focusing on 1) the concrete syntax, 2) expressiveness, and 3) usability as follows:

RQ1: How can a textual DSL model executable CPs that supports *external communication*, *customizable graphical representation* of medical data, and *visualization of CP models*?

RQ2: Can the DSL model CPs from *different medical fields* with *diverse complexity* while being *understandable* to clinical experts?

RQ3: Do the modelers regard the DSL and the development environment as *user-friendly* and *learnable* for modeling CPs?

We answer **RQ1** by identifying essential concepts for modeling CP from literature research. Additionally, we selected the Smart Adaptive Case Management (SACM) (Michel, 2020), the engine of the CONNECARE platform for integrated care that leverages CP as a decision support tool for operating treatments of chronic diseases (Vargiu et al., 2017). SACM defines CPs using a custom XML structure, thus we analyze the current XML syntax and SACM capabilities to devise approaches for optimizing the CP declaration with a DSL.

To model the CP concepts, we applied guidelines and best practices of DSL design (Karsai et al., 2014, pp. 3-5) to develop the Acadela syntax. First, our DSL shall include only the necessary domain concepts to model the identified CP requirements. Our study develops a textual DSL because we aim to create a lightweight, extensible DSL that is platform-independent, i.e., the DSL can generate a CP model in different formats (e.g., XML, UML) to be compatible with various e-Health systems. To realize this goal, we define the Acadela grammar using the *textX* metalanguage (Dejanović et al., 2017). The textX grammar parses the input CP model and generates the corresponding abstract syntax tree (AST). An Acadela Interpreter processes the AST to construct a CP meta-model object, and then translates the CP meta-model into a SACM-compatible format, as shown in Figure 1. SACM creates CP models that execute patient treatments at runtime from their meta-models.

Regarding usability, our DSL aims to provide low-technical syntax, concise constructs, and flexible syntactic rules extracted from popular programming lan-

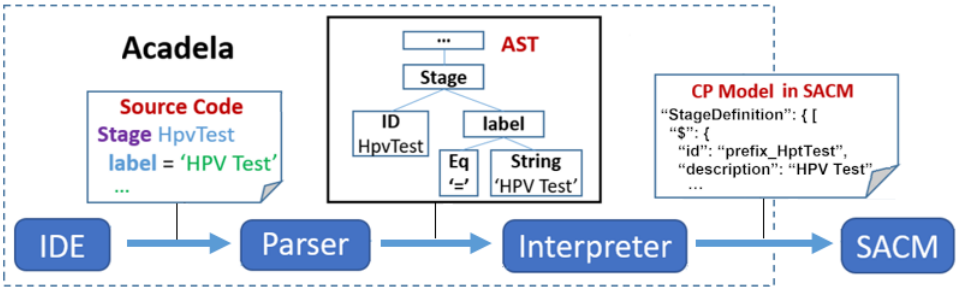

Figure 1: The process to generate an SACM CP meta-model from a CP definition code written in Acadela.

guages (e.g., no semicolon to end a statement, indent- and case-insensitive like in SQL). We select these features with the expectation that modelers with basic programming knowledge can quickly learn and apply our DSL's modeling concepts. Afterward, we follow the guidelines of design science from Hevner et al. (2004, p. 80) to begin the iterative design, development, and refinement cycle of our DSL.

Considering Acadela's expressiveness, **RQ2** states *the potential of modeling CPs with* ***different complexity*** *applied in* ***various medical departments***. Following this direction, we contacted six medical experts and collected five CPs from their fields of expertise. We modeled the CPs in Acadela, which then generates CPs in a format compatible with SACM. We demonstrated the execution of CP models to medical professionals in SACM to assess the DSL's accuracy and applicability from their perspective.

Finally, **RQ3** explores the DSL's usability from the perspective of modelers. Thus, we conducted descriptive user studies with technical staff in the healthcare sector or medical researchers with programming or modeling experience. The participants developed a hypertension CP using our Acadela IDE and gave us quantitative and qualitative feedback on the DSL's user-friendliness, learnability, and applicability.

3 RELATED WORK

Graphical DSLs. Several DSLs are developed to model CPs supporting evidence-based decisions in clinical treatments. The motivation is that standard business process modeling notations, like BPMN, do not fully support modeling clinical procedures that require discretionary execution paths to handle unpredictable events. In this direction, Burwitz et al. (2013) developed *CP-Mod* to support *Evidence-based Medicine* (EBM) with the modeling of 1) decision support, i.e., the recommended actions based on decision criteria, along with references to medical guideline documents; 2) *Treatment alternatives* and *probability* to indicate possible care approaches; 3) *time event* representing time slots, waiting period or temporal dependencies between activities.

Similarly, Braun et al. (2014) and Neumann et al. (2016, 2017) extended BPMN to develop BPMN4CP and BPMN^{SIX} . Both DSLs first construct a domain ontology through 1) requirement analysis and 2) domain analysis to examine whether a modeling language is appropriate to model the concepts, properties, and constraints of the domain. Next, they perform an equivalence check to assess which modeling concepts are constructible with default BPMN elements and to which extent. After that, the authors classify whether a domain concept is modelable by combining multiple BPMN elements (Equivalence by Composition) or using BPMN elements and domain-specific attributes (Equivalence by Specification). The authors then apply a BPMN extension procedure of Stroppi et al. (2011) to develop a domain class model (CDME) to extend concepts not modelable by standard BPMN elements. The CDME model is an abstract syntax that is derivable to construct the concrete syntax (graphical notations) in a BPMN extension model format (BPMN+X). As a result, BPMN4CP and BPMN^{SIX} create an extended BPMN model for CPs that includes *quality indicators* and management of human, equipment, consumables, and document resources concepts.

Textual DSL. Msosa (2018, 2019) developed FCIG, a textual DSL to model Clinical Practice Guidelines (CPGs) in Computer-Interpretable Guidelines (CIGs) forms. First, Msosa analyzed the domain concepts of a CIG *Guideline*, which comprises a set of *Recommendations* containing *Actions* and *Conditions* to specify which medical interventions apply to particular circumstances. Next, Msosa leveraged the Xtext (Eclipse Foundation, nda) DSL build tool to define FCIG grammar. When processing a Guideline, FCIG navigates through the AST to 1) retrieve all *Recommendations* in the *Guideline*, 2) for each *Recommendation*, return the list of *Actions* that satisfy the

Table 1: Features Comparison of DSLs for CP modeling.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	SF1	SF2
CP-Mod	●	●	●	●	○	●	○	●	○	●	○	○	○	●	●
DSML4CPs	●	●	●	●	●	●	●	●	●	●	○	○	○	●	●
BPMN4CP	●	●	●	●	●	●	○	●	●	●	○	○	○	●	●
BPMN^{SIX}	●	●	●	●	●	●	●	●	●	○	○	○	○	●	●
FCIG	●	●	○	○	○	○	○	◐	●	○	○	○	○	◐	○
Acadela	●	●	●	●	●	●	●	●	●	◐	●	●	●	◐	●

○ No Support ◐ Partial Support ● Full Support

Condition, and 3) generate a function that evaluates all *Guideline*'s *Recommendations* and returns applicable *Actions*. The advantages of FCIG are 1) concise syntax and 2) offering an IDE with syntax highlighting and auto-completion to support modeling.

While existing DSLs model key aspects of CP operation, they do not mention the support of external systems communication, dynamic graphical rendering, and importing CP elements. Our study discusses the necessity of the features, coded **F11**, **F12**, and **F13** in Section 4, and our modeling solutions in Section 5.

4 CP MODELING FEATURES

From the literature study of existing DSLs, e-Health systems, and CP definition and execution, we collected the requirements for modeling executable CPs in e-Health applications. Furthermore, we also collect features for textual DSLs to support technical staff in constructing CP models. Table 1 summarizes the implementation of identified CP modeling (**F#**) and supporting features (**SF#**) in DSLs from relevant literature and Acadela.

F1: The DSL shall model medical processes using concepts from the healthcare domains familiar to healthcare professionals. **Rationale:** First, constructing activities and phases of medical procedures is essential to declare structured or alternative tasks in the treatment. Second, concepts used in the DSL should map to the terminology which prospective users are familiar with (Frank, 2013, p. 136).

F2: The DSL shall model alternative treatment processes, their triggering conditions, and potential values that trigger other alternatives. **Rationale:** Care professionals frequently handle unpredictable scenarios during the planned treatment process. Thus, a DSL shall provide decision support to care professionals, patients, and their relatives (De Bleser et al., 2006, p. 562).

F3: The DSL shall model evidence classes with data assigned to them. Additionally, the DSL shall support referencing the evidence. **Rationale:** CP processes and decisions following EBM principles (Wang et al. (2011, 2021)) shall be supplemented with information on the evidence to decide the appropriate interventions.

F4: The DSL shall model iterative treatment processes. Parallel repetition should also be supported. **Rationale:** CPs shall support control flow by executing both optimal, sequential procedures and repetitive, unpredictable processes. Parallel execution is necessary to construct simultaneous tests (e.g. parallel PCR tests (Berdal et al., 2008), (Perchetti et al., 2020, p. 2)).

F5: The DSL shall model each medical process's treatment goal(s) based on medical practices and preferences of individual patients. **Rationale:** Expressing care goals based on the applied remedies and the patient's preferences fosters treatment transparency (Heß et al., 2015, p. 6). Furthermore, this feature encourages medical professional teams to establish unified remedies to achieve the care goal(s).

F6: The DSL shall support assigning responsibilities to medical processes. The responsibilities distinguish between staff that executes medical tasks and staff responsible for the treatment outcome. **Rationale:** Specifying the responsibilities of each healthcare professional optimizes the sequencing of medical procedures, indicates contact persons, and fosters communication among staff (Heß et al., 2015, p. 6).

F7: The DSL should model checklists associated with medical processes. **Rationale:** Checklists are widely accepted and helpful instruments to present details about the execution of medical processes (Healey et al., 2011, p. 3) (Wolff et al., 2004, pp. 430-431).

F8: The DSL should model various aspects and information needs of stakeholders in medical institutions. **Rationale:** CPs aim to optimize medical care from various perspectives. Therefore, a DSL should provide concepts to model treatment-related information, such as (multidisciplinary) medical or organizational aspects (Heß et al., 2015, p. 6). Besides, the DSL should include medical guidelines to assist care pro-

fessionals to decide on appropriate interventions.

F9: The DSL should model the allocation of resources to the medical process. **Rationale:** CPs also aims to manage and optimize resource consumption (Heß et al., 2015, p. 6). Resources comprise equipment or documents used during the medical process (Neumann et al., 2016, p. 4), medicine, and facilities (Braun et al., 2016, p. 3252).

F10: The DSL should model time constraints and explicit time events. **Rationale:** the definition of CP incorporates the timing necessary to achieve care goals (De Bleser et al., 2006), (Campbell et al., 1998), (Vanhaecht et al., 2006). Therefore, a DSL should provide features to declare due dates or timed activities such as periodic or time-lapsed tasks.

F11: The DSL should model external requests to orchestrate and integrate with existing or external systems (Michel, 2020, p. 37). **Rationale:** e-Health services may need to interact with legacy systems that store user data. Furthermore, delivering care services may require sharing data with partner systems at different process lifecycles (e.g., activated, terminated). For example, sending blood pressure data to a monitoring application when completing the measurement.

F12: The DSL should support customizing the graphical representation of activity. **Rationale:** Customizable UI display enhances extendability and supports the modeling of specialized clinical use cases because modelers can define dynamic visual representations of data without relying on predefined templates (Michel and Matthes, 2018). Besides, a comprehensive overview of medical status influences the analysis of the patient's condition and decision-making process of medical experts (Suganthi and Poongodi, 2021), which affects the treatment quality. Thus, a DSL shall assist modelers in designing comprehensive graphical representations of data.

F13: The DSL should support importing CP elements into the current CP definition. **Rationale:** Repeatedly declaring common procedures (e.g., patient admission) in each CP results in duplication and cumbersome modification for all affected CPs when the tasks change. Therefore, defining these shared elements once and importing them into CPs foster reusability and flexibility of the modeling process.

SF1: The DSL should offer an IDE. For textual DSLs, the IDE should have syntax highlighting and auto-completion capabilities. **Rationale:** The combination of auto-completion, syntax highlighting, and syntax checking can enhance productivity when defining CP elements, increase usability (Cook et al., 2007, p. 16-17) (Merkle, 2010), and prevent typos.

SF2: The DSL should visualize the modeled CP to demonstrate the reconstructed elements, such as phases, activities and their inputs or outputs, responsibilities, and control flows, using basic graphical notations. **Rationale:** A CP visualization provides visual clues that can enhance the understandability and readability of the models (Frank, 2010, p. 1). Another benefit is reducing the learning effort as the clinical experts do not have to understand modeling concepts in-depth (e.g., (Hermans et al., 2009, p. 433)). Additionally, one noticeable feedback from our interviews with technical staff is that Acadela needs a graphical visualization to provide an overarching overview of the CP model. These visual clues assist the debugging and analysis of the model.

Observation: Existing DSLs satisfy the **F1** feature by providing essential constructs to model generic CP process elements, e.g., *Activities* (Braun et al., 2016, p. 3252), *ProcessDefinition* (Michel, 2020, p. 45). In addition, the DSLs include concepts representing decision criteria with conditional expression for activating alternative treatment paths (e.g., *DecisionCriterion* (Braun et al., 2014; Burwitz et al., 2013), *Criterion* (Heß et al., 2015, p. 10), and *SentryDefinition* (Michel, 2020, pp. 45, 47)), thus supporting **F2**.

Most DSLs support features **F3**, **F4-F6**, and **F8**. We considered FCIG to fulfill **F8** partly because the DSL can use the *Action* concept to open a medical document at a particular URL. However, FCIG does not offer constructs to model organizational aspects (e.g., medical departments or teams). Most DSLs support modeling evidence classes and repetitive activities (**F3** and **F4**). Furthermore, existing DSLs can model resource management aspects in CP operations (**F6** and **F8**). Most DSLs can express treatment goals (**F5**), thus assisting care professionals in communicating and planning their therapies. Not all DSLs support features **F7** and **F10** because of the original scope of their CP application context.

Regarding the supporting features **SF1** and **SF2**, graphical DSLs offer an expressive IDE with extended concepts to help modelers construct CPs by interacting with graphical elements. FCIG, as a textual DSL, also develops an IDE with syntax highlighting, yet it does not auto-complete keywords or visualizing CIG models. Acadela auto-completion covers keywords but not element IDs in a CP meta-model.

The **F11**, **F12**, and **F13** features are the novel contributions of Acadela, which aim to support inter-system communications and provide flexible modeling capabilities to modelers. Existing DSLs did not discuss the three requirements, yet they are essential in modeling CPs in integrated, adaptive e-Health platforms as explained in the **Rationale**.

The following section presents our methods to

model CP that support the 15 identified features.

5 CP METAMODEL DEFINITION

In SACM, medical experts and modelers need to collaborate to define *Stages* and *Tasks* in the *care process*, *control flow*, *responsibilities*, and *communication to external systems* in a CP. The first three elements are foundational building blocks for developing standardized and multidisciplinary CPs. This section presents the Acadela grammar to enforce syntactic rules of CP declaration, followed by the CP modeling concepts and their concrete syntax in Acadela to address the CP modeling requirements.

5.1 Grammar

Acadela arranges CP elements in a subsumption order to provide a hierarchical structure of CP models. For example, a *Case* object contains multiple *Stages* and each *Stage* consists of *Tasks*. Each CP element has user-defined attributes following the key=value pattern. In addition, Acadela expresses attributes accepting only SACM-defined values as *directives*, which conforms to the #<attributeValue> pattern. The below code expresses the semantics of Acadela grammar in Extended Backus Naur Form (EBNF). To condense the grammar, attributes, and directives of a CP element are declared as <CpElement>Attrs or <CpElement>Drts, respectively.

In EBNF, the # sign denotes the order of rules within a group is interchangeable. The * sign means a CP element has zero or more instances, while the + sign allows one or more instances. The ? mark represents an optional element.

```
Start = (Import*)? (Workspace | Obj)
Workspace = "Workspace" ID
            "define" Case
Obj = Case | CaseSetting | Stage
    | Task | Form | AttributeValue
    | OutputField | InputField | Hooks
Hook = HttpHook | CaseHook
     | DualTaskHook
Case = "Case" ID (CaseAttrs CaseSetting
        Responsibilities (CaseHook+)?
        SummaryPanel Stage+)#
Responsibilities = (Group* User*)#
CaseSetting = ( CaseOwner CasePatient
                CaseSettingAttrs )#
Stage = "Stage" ID WorkflowDrts
        StageAttrs
       (HttpHook* Precondition* Task+)#
Task = HumanTask | DualTask
       | AutomatedTask
HumanTask = "HumanTask" ID WorkflowDrts
            (TaskAttrs Precondition*)#
            (HttpHook* Form)#
Form = "Form" ID (FormDrts)? Field+
Field = InputField* OutputField*
InputField = "InputField" ID
             InputFieldDrts
             InputFieldAttrs
OutputField = "OutputField" ID
              OutputFieldDrts
              OutputFieldAttrs
OutputFieldDrts = (Mandatory Readonly)#
Mandatory = "#" "mandatory"
          | "notMandatory"
OutputFieldAttrs = (Description
                    Expression ...)#
Description = "label" "=" STRING
```

Regarding *Tasks*, since the syntax of *HumanTask*, *DualTask*, and *AutomatedTask* are similar, the grammar only expresses *HumanTask*. Similarly, *OutputFieldDrts* and *OutputFieldAttrs* characterize the syntax of directives and attributes of other CP elements.

5.2 Modeling CP Elements

Care Process (F1, F3, F4, F7, F12). In SACM, a *Workspace* represents the medical institution providing healthcare services, and a *Case* is the CP for treating a particular disease. Acadela expresses treatment phases and activities in a CP using **Stages** and **Tasks** (**F1**). Since clinical processes are multi-stage procedures, they can comprise distinct and possibly repetitive phases. SACM supports a parallelly repeated *Stage* (Michel, 2020, pp. 80, 105), which Acadela expresses as a #parallelRepeat *directive* (**F4**). Each *Stage* comprises *Tasks* of type: *human*, *automated*, or *dual tasks*, which are, respectively, conducted by a person, third-party system, or both consecutively. Listing 1 shows the Acadela syntax to declare a repeated *Stage* and a *HumanTask* to measure the Body Mass Index (BMI) value. Figure 2 displays the resulting UI of Listing 1.

```
Stage Evaluation #repeatSerial
 label = 'Evaluation'

 HumanTask MeasureBmi
  label = 'Measure BMI'
  dueDateRef = 'Setting.DueDate'

  Form BmiForm
   InputField Height #number(0-3)
    label = 'Height (m):'
   InputField Weight #number(0-300)
    label = 'Weight (kg):'
   OutputField BmiScore #number
    label = 'BMI Score:'
```

```
    expression =
      'Weight / (Height * Height)'
    uiRef = 'colors(0 < yellow < 18.5
            < green < 25 < red < 100)'
```

Listing 1: Sample definition of a BMI Measurement Task in an Evaluation Stage.

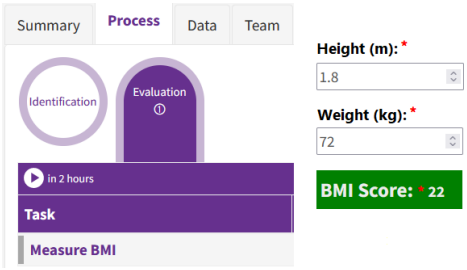

Figure 2: The UI output from the Stage Declaration in Listing 1. The Figure shows the activated Evaluation *Stage* and enabled MeasureBMI *Task* (left), and the UI of the *Task* (right).

Each *Task* has a *Form* containing *InputFields* to accept medical data to collect necessary medical conditions for an *evidence class*, *diagnosis*, or *documentation* (**F1**, **F3**). The medical information supports care experts in determining the appropriate interventions in EBM. Besides, Acadela represents a *checklist* (**F7**) by defining multiple-choice *InputFields* that accept multiple answers. Additionally, the *OutputFields* can visualize the input data in the SACM UI (**F12**).

Control Flow (F2): Medical experts execute alternative actions depending on the patient condition. Therefore, CPs shall support conditional activation of *Stages* or *Tasks* depending on specific medical data. Listing 2 demonstrates the syntax of defining transition conditions using the *Precondition* construct inside a Stage. The *previousStep* attribute states the prerequisite *Stage* or *Task*, while the *condition* attribute is a boolean expression for activating the *Stage*.

```
Stage Evaluation #repeatSerial
  ... // Stage Attributes
  Precondition
    previousStep = 'Identification'
    condition = 'Identification.
        FillConsentForm.Consent = 1'
```

Listing 2: Defining a transition condition of a *Stage*. Here the Evaluation *Stage* is activated if the Identification *Stage* is completed and the patient consents with the treatment.

CP Summary Panel (F5): Listing 3 demonstrates a *SummaryPanel* code displaying the treatment goal from an *InputField*. This element displays the value from an *InputField* or *OutputField* in the *Case Summary* page. *InputFields* can record the care goal while

```
SummaryPanel
  Section TargetBmi #left
    label = "Target BMI:"
  InfoPath Identification.
      SetTreatmentGoal.TargetBmi
```

Listing 3: SummaryPanel Definition to show the care goal. Acadela traces the *InputFields* from the *InfoPath* attribute by parsing the <StageId>.<TaskId>.<FieldId> pattern.

the *OutputFields* values indicate the progress or constraints to achieve the goal.

Responsibilities (F6). SACM supports referencing *individual experts* or *professional groups* in the database and grants them access rights for operating *Stages* or *Tasks* (**F6**), as shown in Listing 4. Modelers can optionally assign the unique ID of a *Group* or *User* to the staticId attribute.

```
Responsibilities
  Group StPaulNurses name = 'Nurses'
    staticId = 'efdc3952es75'
Setting
  Attribute Nurses
    #Link.#Users(StPaulNurses)
    label = 'Nurses'
  ...
Stage Evaluation
  owner = 'Setting.Nurses'
```

Listing 4: SummaryPanel Definition to show the care goal.

Document (F8). In case medical documents (e.g., guidelines, forms) necessary to provide care services (Heß et al., 2015; Braun et al., 2014) are accessible via URLs, modelers can declare a link to each document, as shown in Listing 5.

```
InputField MedicalGuideline
 #documentLink('https://stpaul.de/
     filepath')
 label="Link to Medical Guideline:"
```

Listing 5: Defining a link to a document in Acadela.

Due Date Definition (F10). SACM supports defining a *default due date* from the current time in the *Case Setting*. The due date is assignable to a *HumanTask* or a *DualTask*, as shown in Line 6 of Listing 1.

External System Communication (F11). SACM updates medical status in external systems by sending HTTP requests (*HttpHook*) with *Case*, *Stage*, or *Task* data when an *event* (e.g., activated, completed) occurs in these elements (Michel, 2020, p. 113). Listing 6 defines a *HttpHook* inside the MeasureBMI *Task*. Upon the *Task* completion, the *HttpHook* sends the

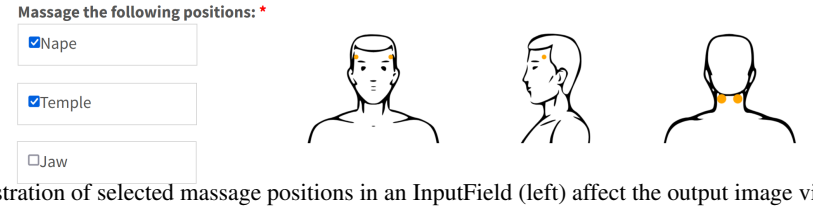

Figure 3: Illustration of selected massage positions in an InputField (left) affect the output image visualization (right).

Task data from its *InputFields* and *OutputFields* to the external system *URL* using the POST *method*. If the request fails, SACM shows the value of the *failureMessage* attribute to the user.

```
HumanTask MeasureBmi
  label = 'Measure BMI'
  ... // Other Task's elements
  Trigger
    On complete invoke 'https://
        partnersystem.de/update'
    method POST with failureMessage
        "Cannot update info!"
```

Listing 6: Defining an external request triggered upon a Task completion

Customizable Graphical Representation (F12). The SACM UI of a *Task* is a grid layout of three columns. Acadela specifies the column position of an *InputField* or *OutputField* using a directive, e.g., *#left*, *#center*, *#right* (**F12**). Besides, SACM supports applying color to a range of numeric values. Line 17 in Listing 1 shows how Acadela defines the green color of the BMI *OutputField* for a value from 18.5 to 24.9.

To render a customizable UI Template, an *OutputField* contains 1) a SVG image comprising graphical elements to visualize medical data, 2) the CSS style declaring the UI effect applied to the elements (e.g., hide, show, fore- or background color), and 3) a SACM command to dynamically modify the CSS style based on *InputField*(s) value. Figure 3 shows an example of displaying massage positions on a human head based on the selected areas.

Import CP Element (F13). Acadela leverages the textX scope provider to import CP elements defined in a file into the current CP. textX supports *FullyQualifiedName* (FQN) scope that expresses the location of an imported object based on the file path and object ID (Dejanović, ndc). For instance, Listing 7 shows how to import a Discharge *Stage* defined in the *discharge.aca* file under the *stage* folder into a CP model.

```
// File path: ./stage/discharge.aca
define Stage Discharge
  ... // Stage Attributes
// -------------
import stages.discharge
// import stages.discharge as di
```

```
Workspace StPaulClinic
  define Case OT1_Obesity
  ...
  Stage Evaluation ...
  use Stage Discharge
  //  use Stage di.Discharge
```

Listing 7: Importing the Discharge *Stage* (lines 1-4) into a CP. Lines 6 and 12 show how to import an element as alias.

Web-Based IDE (SF1). To enhance the usability and productivity of modelers, we developed a Web IDE powered by Monaco[1] with 1) an auto-complete feature to create CP elements from templates and 2) syntax highlighting to help modelers identify CPs elements.

CP Visualization (SF2). Acadela leverages GoJS[2] to develop a CP visualizer. GoJS provides functions to graphically present CP elements. From the JSON CP model generated by the backend, Acadela extracts each CP element type and attribute to construct nodes in the graph. Figure 4 shows the GoJS visualization of a Hypertension CP used in our usability evaluation.

The Acadela Wiki[3] documents the DSL syntax for defining CP elements.

6 EVALUATION

To explore the applicability of Acadela, we designed two descriptive surveys to assess 1) expressiveness, i.e., the ability to model CPs in different medical fields, and 2) usability from the perspective of modelers. We recruited **voluntary participants** from staff working in medical facilities or research institutions. For each evaluation, we collected quantitative data by a survey and applying a *Likert scale* from 1 (Strongly Disagree) to 5 (Strongly Agree) for each statement. We also interviewed participants to obtain feedback regarding the DSL's applicability and limitations.

[1]https://microsoft.github.io/monaco-editor/
[2]https://gojs.net/latest/
[3]https://cobalt-plot-a08.notion.site/Acadela-Wiki-ee58b3f2eb6a4627b55b470fe1717d9f

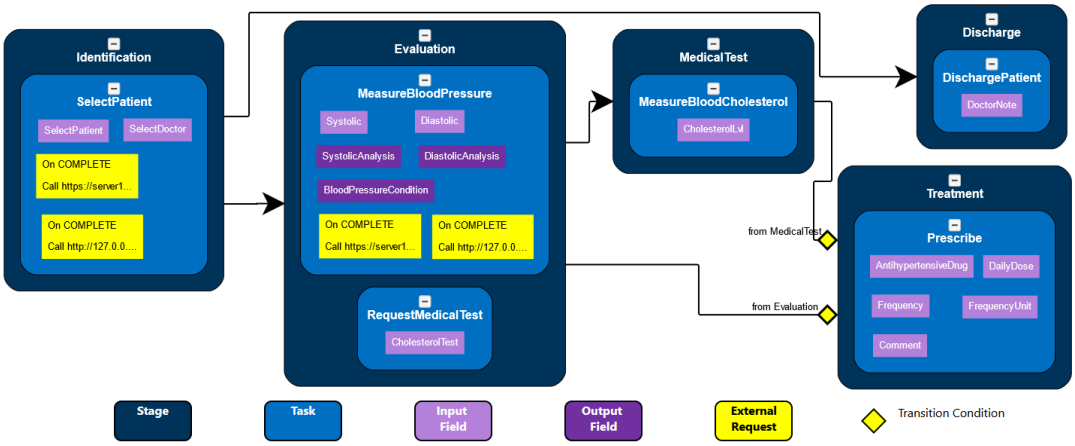

Figure 4: The visualization of the simplified Hypertension CP.

6.1 Expressiveness

Population Sampling: We contacted 16 medical professionals in Germany and received six responses. The response rate is moderate at 37.5 percent. The age range of participants is from 20 to 60 years old. Two medical professionals have more than 25 years of experience, while others have up to five years.

Study Design: The six medical professionals shared five CPs for treating Chronic Obstruction Pulmonary Disease (COPD), Schizophrenia, Chronic Headache, and two Diagnoses of Smoke Inhalation Injury and Cervical Cancer. The CPs have various complexity ranging from linear to dynamic, adaptive treatment procedures. The complex CPs demonstrate the language ability in modeling *transition conditions* for *unpredictable*, *personalized* treatment processes. The first three CPs are linear workflows, while the other two support the decision-making with transition conditions applied to input medical data. The last two CPs are complex due to their *adaptive* and *non-deterministic* nature.

Experiment Environment: We host Acadela and SACM in our computing devices and show them offline or share the screen control via Zoom to avoid being blocked by strict firewall policies in medical institutions. Due to the Corona pandemic, depending on the participant's preference, we presented the CP model, collected the survey, and interviewed them onsite or via Zoom.

Result: For each participant, we model their CP and presented it in the SACM web application at their workplace or via Zoom. After seeing the CP execution, we share the survey link with the participants and ask for their opinions. Figure 5 shows the assessment of medical experts on the accuracy of the modeled CP.

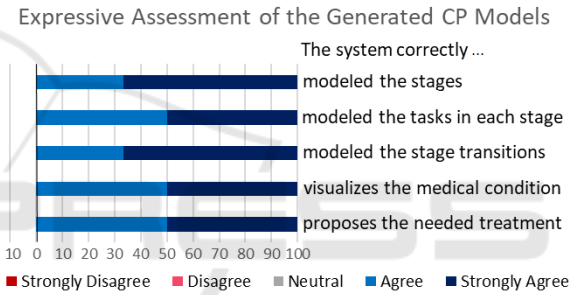

Figure 5: Medical professionals evaluation regarding the accuracy of the modeled CPs.

The result suggests that the *system constructs CPs accurately* with all participants *agreed* or *strongly agreed* with each statement. In other words, Acadela is potentially capable of reconstructing various medical procedures precisely, such that the SACM system can correctly show and execute the treatment process accordingly. Furthermore, in the subsequent interview, the participants considered the modeled CPs as correct. However, our Cervical Cancer Diagnosis CP model does not correctly show the period before the next test. Regarding the applicability, the participants regarded the CP models as "appropriate as a guideline". Nevertheless, they should "address treatment variations" to be practically applicable. The evaluation outcome implies a threat to internal validity, as our study focuses on standard treatment processes but did not substantially model potential alternative paths in a CP. However, medical experts consider modeling variations as a significant indicator of expressiveness.

When considering external validity, the limited number of CPs and involved medical professionals in our study suggests that Acadela may need additional

modeling concepts to express the medical terminology and procedures in other treatments.

6.2 Usability

Population Sampling: We invited 21 technical staff at medical or research institutions and recruited eight participants from Germany, Spain, the Netherlands, and Italy. The response rate is moderate at 33.3 percent. Seven participants were 18 to 45 years old, and one technical expert is over 46. Three participants have less than four years of experience, others have more than eight years.

Study Design: All participants joined the experiments online via Zoom. Each participant first attended a *training* session, in which we introduced how Acadela models CP elements and practiced defining a CP together by completing an exercise. Next, in a **modeling** session, the participants constructed elements of an incomplete hypertension CP and fixed bugs of another CP model by examining their error messages (EMs). We controlled the *evaluation tasks* and *bugs* such that all participants modeled the same CP elements and debugged the same syntax and semantic errors. Performing the same activities builds a common baseline to gauge usability and learnability.

Result: After the debugging session, we asked the experts to assess the usability of Acadela and its error validation. We presented a questionnaire with statements regarding the a) User-friendliness of the DSL, b) Helpfulness of EMs, and c) Overall usability assessment using the System Usability Scale (SUS). Figure 6 presents the evaluation outcome.

In the subsequent interview, overall, the participants comment on the syntax as ”simple”, ”elegant”, and ”easy to learn” for users with competent programming experience. In addition, EMs are considered ”useful” as they ”show the line number and the problem” to help users ”pinpoint the error”. Six out of eight experts considered Acadela to have the potential to model CPs in e-Health applications.

Regarding limitations, the auto-complete feature should ”show the names of elements in the code”, which is convenient for the users and prevents typing a non-existing CP component. Furthermore, three participants share an opinion that Acadela is ”fairly easy to use”, but ”depend on the background and education” of modelers. Therefore, modelers need fundamental programming skills to model and debug the Acadela code. One participant states that Acadela

ID	Syntax Usability Statement
S1	The syntax for creating CP elements, i.e. Stage, Task, Form, Field, is straightforward
S2	Editing CP elements is easy with the language
S3	Importing external modules is straightforward
S4	The language syntax was easy to learn and use

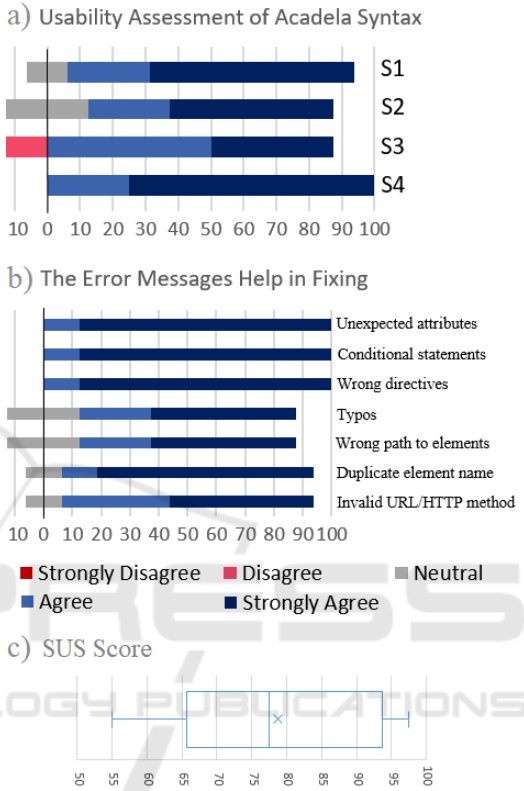

Figure 6: Usability quantitative evaluation result.

should model ”infographic”, ”videos”, and ”gamification factors” to further support medical staff.

7 CONCLUSION

Our study develops Acadela, a textual DSL for modeling CPs with visualization capability. Acadela supports the definition of *workflow phases and activities*, *control flow*, *responsibilities*, *medical document resources*, *timing constraints*, *communication to external services*, *dynamic graphical representation of data* and the *import* of these elements to foster reusability (**RQ1**). The language is overall considered user-friendly and learnable by professionals with technical knowledge.

We leveraged the textX meta-language, guidelines of DSL design, and error messages to develop the grammar, concrete syntax, and error validator of

Acadela. To evaluate our DSL expressiveness (**RQ2**), we modeled five CPs and presented the executable workflow in SACM, an e-Health platform for integrated care, to six care professionals in five medical fields. The result suggests that Acadela can *accurately* model CP workflows in *different medical departments* and *complexity*. However, both medical professionals and technical staff wished to see more modeling of visualization and variations in their CP.

To evaluate Acadela's usability (**RQ3**), we invited eight technical experts working in the healthcare industry or research to model a simplified CP for hypertension treatment and fix bugs of another CP based on their error messages (EMs). The result implies that Acadela is potentially usable and learnable to technical staff. Most participants deem the syntax "simple" and "easy to learn", while EMs help them locate, understand and fix errors. Therefore, they consider the DSL applicable to model CPs in e-health applications. However, we reflected on the result that Acadela is user-friendly for modelers with solid basic programming knowledge or above. In addition, the participants wished to see more capabilities, such as the rendering of statistics. and auto-completion of CP elements' ID. Another concern is more participants are needed to assess the DSL usability substantially.

Future Work. To further validate the applicability of Acadela, one can expand the evaluation for modeling CPs in more medical fields, such as Chinese medicine or Orthopedics. Operating CPs in practical settings enable us to address the daily-life needs and concerns of medical professionals while addressing the unique challenges of applying CPs in the medical field. Furthermore, different medical treatments require diverse forms of medical data visualization and additional medical concepts, hence we can investigate the applicability and limitation of CP modeling features provided by Acadela. This future practical study is the first cornerstone for identifying potential extensions or alternative solutions to assist medical professionals in delivering quality care to patients.

ACKNOWLEDGEMENTS

We sincerely thank the medical professionals, researchers, and technical staff for their time, support, and patience in providing clinical pathways, conducting experiments, and valuable feedback to our study.

REFERENCES

Baar, T. (2015). Verification support for a state-transition-dsl defined with Xtext. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, pages 50–60. Springer.

Berdal, K. G., Bøydler, C., Tengs, T., and Holst-Jensen, A. (2008). A statistical approach for evaluation of PCR results to improve the practical limit of quantification (LOQ) of GMO analyses (SIMQUANT). *European Food Research and Technology*, 227(4):1149–1157.

Braun, R., Schlieter, H., Burwitz, M., and Esswein, W. (2014). Bpmn4cp: Design and implementation of a bpmn extension for clinical pathways. In *2014 IEEE international conference on bioinformatics and biomedicine (BIBM)*, pages 9–16. IEEE.

Braun, R., Schlieter, H., Burwitz, M., and Esswein, W. (2016). BPMN4CP Revised–Extending BPMN for Multi-perspective Modeling of Clinical Pathways. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3249–3258. IEEE.

Burwitz, M., Schlieter, H., and Esswein, W. (2013). Modeling clinical pathways-design and application of a domain-specific modeling language.

Campbell, H., Hotchkiss, R., Bradshaw, N., and Porteous, M. (1998). Integrated care pathways. *Bmj*, 316(7125):133–137.

Cook, S., Jones, G., Kent, S., and Wills, A. C. (2007). *Domain-specific development with visual studio dsl tools*. Pearson Education.

De Bleser, L., Depreitere, R., Waele, K. D., Vanhaecht, K., Vlayen, J., and Sermeus, W. (2006). Defining pathways. *Journal of nursing management*, 14(7):553–563.

Dejanović, I., Vaderna, R., Milosavljević, G., and Vuković, Ž. (2017). TextX: a Python tool for domain-specific languages implementation. *Knowledge-based systems*, 115:1–4.

Dejanović, I. (n.d.c). Reference resolving expression language (rrel). http://textx.github.io/textX/3.0/rrel/.

Eclipse Foundation (n.d.a). Language engineering for everyone! https://www.eclipse.org/Xtext/.

Every, N. R., Hochman, J., Becker, R., Kopecky, S., and Cannon, C. P. (2000). Critical pathways: a review. *Circulation*, 101(4):461–465.

Frank, U. (2010). Outline of a method for designing domain-specific modelling languages. Technical report, ICB-research report.

Frank, U. (2013). Domain-specific modeling languages: requirements analysis and design guidelines. In *Domain engineering*, pages 133–157. Springer.

Hai, J.-J., Wong, C.-K., Un, K.-C., Wong, K.-L., Zhang, Z.-Y., Chan, P.-H., Lam, Y.-M., Chan, W.-S., Lam, C.-C., Tam, C.-C., et al. (2019). Guideline-based critical care pathway improves long-term clinical outcomes in patients with acute coronary syndrome. *Scientific Reports*, 9(1):1–9.

Healey, A. N., Nagpal, K., Moorthy, K., and Vincent, C. A. (2011). Engineering the system of communication

for safer surgery. *Cognition, Technology & Work*, 13(1):1–10.

Hermans, F., Pinzger, M., and Deursen, A. v. (2009). Domain-specific languages in practice: A user study on the success factors. In *International Conference on Model Driven Engineering Languages and Systems*, pages 423–437. Springer.

Heß, M., Kaczmarek, M., Frank, U., Podleska, L., and Täger, G. (2015). A domain-specific modelling language for clinical pathways in the realm of multi-perspective hospital modelling.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pages 75–105.

Iroju, O., Soriyan, A., Gambo, I., Olaleke, J., et al. (2013). Interoperability in healthcare: benefits, challenges and resolutions. *International Journal of Innovation and Applied Studies*, 3(1):262–270.

Jouault, F., Bézivin, J., and Kurtev, I. (2006). TCS: a DSL for the specification of textual concrete syntaxes in model engineering. In *Proceedings of the 5th international conference on Generative programming and component engineering*, pages 249–254.

Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M., and Völkel, S. (2014). Design guidelines for domain specific languages. *arXiv preprint arXiv:1409.2378*.

Kurz, M., Schmidt, W., Fleischmann, A., and Lederer, M. (2015). Leveraging CMMN for ACM: examining the applicability of a new OMG standard for adaptive case management. In *Proceedings of the 7th international conference on subject-oriented business process management*, pages 1–9.

Merkle, B. (2010). Textual modeling tools: Overview and comparison of language workbenches. In *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*, OOPSLA '10, page 139–148, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/1869542.1869564.

Michel, F. (2020). *A Collaborative Purely Meta-Model-Based Adaptive Case Management Approach for Integrated Care*. Dissertation, Technical University Munich, Munich.

Michel, F. and Matthes, F. (2018). A holistic model-based adaptive case management approach for healthcare. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 17–26. IEEE.

Mitchell, J. (1999). *From telehealth to e-health: the unstoppable rise of e-health*. Department of Communications, Information Technology and the Arts.

Msosa, Y. J. (2018). *Modelling evolving clinical practice guidelines: a case of Malawi*. PhD thesis, University of Cape Town.

Msosa, Y. J. (2019). FCIG grammar evaluation: A usability assessment of clinical guideline modelling constructs. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1141–1146. IEEE.

Neumann, J., Rockstroh, M., Franke, S., and Neumuth, T. (2016). BPMNSIX–A BPMN 2.0 Surgical Intervention Extension. In *7th workshop on modeling and monitoring of computer assisted interventions (M2CAI), 19th international conference on medical image computing and computer assisted interventions (MICCAI 2016), Athens, Greece*.

Neumann, J., Wiemuth, M., Burgert, O., and Neumuth, T. (2017). Application of activity semantics and BPMN 2.0 in the generation and modeling of generic surgical process models. *Journal of the International Foundation for Computer Assisted Radiology and Surgery: International Journal of Computer Assisted Radiology and Surgery*, 12:48–49.

Panella, M., Marchisio, S., Barbieri, A., and Di Stanislao, F. (2008). A cluster randomized trial to assess the impact of clinical pathways for patients with stroke: rationale and design of the Clinical Pathways for Effective and Appropriate Care Study [NCT00673491]. *BMC health services research*, 8(1):1–8.

Panella, M., Marchisio, S., and Di Stanislao, F. (2003). Reducing clinical variations with clinical pathways: do pathways work? *International Journal for Quality in Health Care*, 15(6):509–521.

Perchetti, G. A., Nalla, A. K., Huang, M.-L., Jerome, K. R., and Greninger, A. L. (2020). Multiplexing primer/probe sets for detection of SARS-CoV-2 by qRT-PCR. *Journal of Clinical Virology*, 129:104499.

Preston, S., Markar, S., Baker, C., Soon, Y., Singh, S., and Low, D. (2013). Impact of a multidisciplinary standardized clinical pathway on perioperative outcomes in patients with oesophageal cancer. *Journal of British Surgery*, 100(1):105–112.

Rieger, C., Westerkamp, M., and Kuchen, H. (2018). Challenges and Opportunities of Modularizing Textual Domain-Specific Languages. *MODELSWARD*, pages 387–395.

Stroppi, L. J. R., Chiotti, O., and Villarreal, P. D. (2011). Extending BPMN 2.0: method and tool support. In *International Workshop on Business Process Modeling Notation*, pages 59–73. Springer.

Suganthi, S. and Poongodi, T. (2021). Interactive Visualization for Understanding and Analyzing Medical Data. In *Exploratory Data Analytics for Healthcare*, pages 101–123. CRC Press.

Thakrar, B. T., Grundschober, S. B., and Doessegger, L. (2007). Detecting signals of drug–drug interactions in a spontaneous reports database. *British journal of clinical pharmacology*, 64(4):489–495.

Vanhaecht, K., WITTE, K. D., Depreitere, R., and Sermeus, W. (2006). Clinical pathway audit tools: a systematic review. *Journal of nursing management*, 14(7):529–537.

Vargiu, E., Fernàndez, J., Miralles, F., Cano, I., Gimeno-Santos, E., Hernandez, C., Torres, G., Colomina, J., de Batlle, J., Kaye, R., et al. (2017). Integrated care for complex chronic patients. *International Journal of Integrated Care*, 17(5).

Wang, S., Yu, H., Liu, J., and Liu, B. (2011). Exploring the methodology and application of clinical path-

way in evidence-based Chinese medicine. *Frontiers of medicine*, 5(2):157–162.

Wang, X., Chen, J., Peng, F., and Lu, J. (2021). Construction of clinical pathway information management system under the guidance of evidence-based medicine. *Journal of Healthcare Engineering*, 2021.

White, M. (2009). Case management: Combining knowledge with process. *BPTrends, July*.

Wienands, C. and Golm, M. (2009). Anatomy of a visual domain-specific language project in an industrial context. In *International Conference on Model Driven Engineering Languages and Systems*, pages 453–467. Springer.

Wolff, A. M., Taylor, S. A., and McCabe, J. F. (2004). Using checklists and reminders in clinical pathways to improve hospital inpatient care. *Medical Journal of Australia*, 181(8):428–431.

Yang, W. and Su, Q. (2014). Process mining for clinical pathway: Literature review and future directions. In *2014 11th international conference on service systems and service management (ICSSSM)*, pages 1–5. IEEE.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS