Tool support to aligning requirements and testing through behaviour-driven requirements patterns

Pollyana de Queiroz Ribeiro[1], **Ernesto F. Veiga**[2], **Mariana C. Martins**[2], **Auri M. R. Vincenzi**[3], **Taciana N. Kudo**[2], **Renato F. Bulcão-Neto**[2]

[1]Universidade Estadual de Goiás, Santa Helena de Goiás, Brazil

[2]Instituto de Informática, Universidade Federal de Goiás, Goiânia, Brazil

[3]Departamento de Computação, UFSCAR, São Carlos, Brazil

pollyana.queiroz@ueg.br, auri@ufscar.br, {taciana,rbulcao}@ufg.br

Abstract. *The software industry still struggles with adverse effects of a weak alignment between requirements and testing. The Software Pattern Metamodel (SoPaMM) aligns requirements and test patterns under the influence of agile practices. However, these patterns will be more beneficial for professionals if development activities are supported by a software tool. This paper presents the behaviour-DRivEn Application Model generator (DREAM) tool, automatically generating requirements and test specifications from SoPaMM-based patterns. We show how DREAM supports requirements elicitation and specification, test case elaboration, and software documentation using a patterns catalogue for electronic health record systems.*

1. Introduction

Requirements reuse is a feasible approach to lessen the negative impacts on software projects whenever Requirements Engineering (RE) activities are inadequately executed [Irshad et al. 2018]. Reusing the knowledge acquired in previous projects allows making the RE activities more prescriptive and systematic while facilitating the reuse of existing requirements artifacts and improve teams' productivity [Barros-Justo et al. 2018].

Among the requirements reuse approaches existing, we have investigated the theory and practice of software requirements patterns (SRP). An SRP is an abstraction that combines behaviours and services common to several systems and that can be reused in similar software [Withall 2007]. The use of SRP in software projects positively affects RE activities, e.g., saving time and improving the quality of requirements regarding completeness, uniformity, consistency, and clarity [Da Silva and Benitti 2013].

Recent work [Kudo et al. 2019a] shows that those benefits acquired with SRP are primarily focused on the RE process, despite the intrinsic relationship between RE outcomes and the remaining software development phases. For instance, consider the established association between requirements and system tests and between business cases and acceptance tests. However, despite the knowledge accumulated over the years, the industry still copes with adverse effects resulting from a weak alignment between requirements and testing [Bjarnason and Borg 2017].

In light of this, we elaborated on a metamodeling solution to relate SRP to outcomes produced over the software development life cycle [Kudo et al. 2022]. Our solution called

Software Pattern MetaModel (SoPaMM) associates SRP with software test patterns (STP) that are high-value testing solutions to recurrent behaviours in various scenarios. Besides, we implemented the Terminal Model Editor (TMEd) tool for supporting the development process of pattern catalogues as SoPaMM instance models [Kudo et al. 2020]. Using TMEd, we have built pattern catalogues, e.g., for Brazilian Electronic Health Record (EHR) systems. [Kiatake et al. 2021, Kudo et al. 2020].

However, aligning requirements and testing through SoPaMM, pattern catalogues, and TMEd is not enough for the software industry. TMEd-supported pattern catalogues compile recurrent requirements and test cases for a given application domain. These catalogues are built upon domain experts' knowledge and represented under the SoPaMM grammar, i.e., in such abstract form not suitable for professionals [Kudo et al. 2020]. Thus, there is a need for a tool that automates RE and testing activities, reducing software team members' cognitive load.

This paper presents the behaviour-DRivEn Application Model generator prototype (DREAM). The DREAM's current version focuses on the automatic generation of software artifacts from pattern catalogues built using TMEd. We show how DREAM maps each element of a pattern catalogue to produce specification documents of requirements and test cases and a requirement traceability matrix. Information items applicable to RE and testing [ISO/IEC/IEEE 2018, ISO/IEC/IEEE 2013] and their content influenced the generation of documents by the DREAM tool.

To illustrate the DREAM capabilities, we describe a scenario in which it reads the pattern catalogue we made for EHR systems [Kudo et al. 2020]. The DREAM tool is the first ongoing work capable of generating software specifications from behaviour-driven requirement pattern catalogues to the best of our knowledge. As developed, we believe DREAM can automate RE and testing-related tasks, save development time, and reduce the team's workload.

This paper is organised as follows: Section 2 analyses related work; Section 3 presents theoretical background; Section 4 describes how DREAM supports RE and testing activities; and Section 5 brings final remarks and future work.

2. Related Work

The SERS tool [Benitti and Silva 2013] provides requirement analysts with user and project management facilities, search, selection, and reuse of requirement patterns, requirements recommendation and tracing, and requirements specification documentation and printing. Similarly, PABRE-Proj [Palomares et al. 2013] users can manage projects, import and browse catalogues, reuse requirement patterns, and generate requirements specifications. Also, PABRE-Proj allows the generation of calls for tender documents and usage statistics for patterns. PABRE-Proj is the one that most resembles DREAM because the catalogues' information items also rely on a metamodel.

Besides aiding requirements elicitation and specification through pattern catalogues reuse, the computational support proposed in [Barcelos and Penteado 2017] provides requirement professionals with pattern catalogues specification and management. This functionality is similar to TMEd's, whose pattern catalogues generated are DREAM inputs. In turn, the InstRP Editor [Beckers et al. 2013] assists professionals in selecting and instantiating cloud-based security requirements using a cloud system analysis pattern.

In brief, every tool, including DREAM, reads requirement pattern catalogues, allows patterns reuse aiding elicitation, and generates requirements specifications as instances of the catalogue's content. As the SoPaMM-based catalogues bridge requirement and test patterns, DREAM goes further by also documenting test case specifications traced to requirements.

3. Background

This section presents components required to comprehend the DREAM support in our behaviour-driven requirement pattern approach: the Software Pattern MetaModel (SoPaMM) and the Terminal Model Editor (TMEd) tool.

3.1. The SoPaMM Metamodel

The SoPaMM metamodel describes how requirement patterns and test patterns are specified, related, and classified [Kudo et al. 2022]. The Catalogue is the coarsest grained reuse unit for collecting software pattern-related concepts in SoPaMM. A Software Pattern Bag (SPB) comprises Software Pattern (SP) elements that, in turn, are extensible points to allow different types of SP, such as requirement, test, or other software patterns. Thus, the SPB concept allows organising, in the same catalogue, software patterns for problems at various software development stages. SoPaMM is flexible regarding relationship types between catalogues, SPBs, and SPs by not predefining them (e.g., dependence and usage [Withall 2007]).

We have focused on how to bridge functional requirement patterns (FRP) and acceptance test patterns (ATP). Under the influence of the BDD agile methodology, an FRP is a composition of Features, written as user stories, whose behaviours are described as scenarios using the BDD's Gherkin syntax.

Consider the following excerpt of the FRP for user account creation. Its feature describes the administrator user allowed to register a new user for the system. This feature has two behaviours represented: a successful and an unsuccessful scenario, both linked to the same precondition (the attempt of creating a new user). However, the scenarios' execution steps are distinct regarding the user data's validity. Similarly, one different outcome is represented for each scenario, i.e., the new user registration and the display of an error message.

Finally, the Example concept allows defining and linking multiple data to each scenario. Thus, each scenario has one data instance so that one scenario registers a new user successfully. In contrast, the other scenario does not due to an invalid example of user identification number. This FRP-Feature-Scenario-Example representation is what defines our *behaviour-driven functional requirement pattern* approach.

```
FRP Name: FRP_User_Creation
  FEATURE User creation
    As: an administrator
    I_can: create a new user
    So_that: he/she is registered for the system
    SCENARIO Successful user creation
      Given: I am trying to create a new user
      When: I enter <ITRN>, <name>, <gender>, <date of birth>,
            <father's name>, <mother's name>, <role>
      Then: the system should register a new user with these data
      EXAMPLE
        735.101.320-92 | Carlos Chagas | Male | 01.01.2000 |
        Jose Chagas | Carla Chagas | Doctor
    SCENARIO Not successful user creation - Invalid ITRN
      Given: I am trying to create a new user
      When: I enter an invalid <ITRN>
      Then: the system should display the <message> error message
      EXAMPLE
        735.101.320-00 | "This is an invalid individual taxpayer
        registration number (ITRN)"
ATP Name: ATP_User_Creation
  TEST CASE Successful user creation
    InputData = EXAMPLE of SCENARIO Successful user creation
    OutputData = EXAMPLE of SCENARIO Successful user creation
  TEST CASE Not successful user creation - Invalid ITRN
    InputData = EXAMPLE of SCENARIO Not successful user creation -
    Invalid ITRN
    OutputData = EXAMPLE of SCENARIO Not successful user creation -
    Invalid ITRN
```

Aligned to the FRP_User_Creation, the ATP_User_Creation is composed of two test cases, each related to a particular test scenario: (un)successful user creation. Each test case contains preconditions (Given), expected results (When), and postconditions (Then), as well as input and output test data from the respective Example of Scenario. Thus, this is how an ATP verifies compliance against a specific FRP in our approach. Further details about the SoPaMM metamodel can be found elsewhere [Kudo et al. 2019b, Kudo et al. 2022].

3.2. *Terminal Model Editor* (TMEd)

SoPaMM borrows the MetaObject Facility's (MOF) four-layered architecture [OMG 2002] in which lower-layer models are instances of immediately upper-layer models. In the M2 layer, SoPaMM has the MOF metametamodel as a reference model. In the M1 layer, SoPaMM instances are pattern catalogues (or terminal models) built on our TMEd tool. These pattern catalogues follow the SoPaMM grammar constructors presented in Section 3.1, such as Catalogue, SPB, FRP, Feature, Scenario, Example, ATP, etc.

As pattern catalogues' development is not a trivial task, domain experts usually build these through tool support. Developed on top of the Eclipse Modeling Framework (EMF), the TMEd's user interface allows the creation and edition of every SoPaMM component (e.g., SPB, FRP, and ATP) and the generation of an XML output file following the SoPaMM grammar. That output file then feeds the DREAM tool to generate real-world software specifications (or application models, as in the M0 layer).

4. The DREAM tool

DREAM is a web tool developed in these technologies: Java and JavaScript languages, Spring framework (backend), Angular with TypeScript (frontend), and PostgreSQL as DBMS. DREAM supports RE and testing activities through pattern catalogues produced by the TMEd tool. These catalogues include FRP and ATP represented following the SoPaMM metamodel grammar presented previously. DREAM helps requirements teams in the elicitation, specification, and documentation activities. Besides, it also aids testing professionals in elaborating and documenting test cases.

Overall, the DREAM's main requirements include, but are not limited to: user account management; basic software project management; TMEd-based pattern catalogue importing;

imported catalogues search; the exhibition of each catalogue element's contents (e.g., SPB and SP); the association of catalogues with a software project; the reuse of individual catalogue elements supporting requirements elicitation and test case elaboration; and the automatic generation of requirement and test case specifications based on international standards, such as ISO/IEC 29148 [ISO/IEC/IEEE 2018] and ISO/IEC 29119-3 [ISO/IEC/IEEE 2013].

4.1. Supporting requirements elicitation and test case elaboration through patterns reuse

DREAM provides a set of functionalities to support requirements elicitation and test case development through reuse and the automatic generation of artifacts, i.e., software requirements specification (SRS), test case specification (TCS), and traceability matrix (TM). The DREAM user can create and manage software projects as well as register new catalogues or reuse ones already registered. Figure 1(i) depicts a DREAM user interface with the following functionalities: (A) project update; (B) project members; (C) project statistics (mainly related to reuse); (D) associate a patterns catalogue to the current project; (E) view patterns reused; (F) reuse patterns catalogues; (G) edition of requirements and test cases specifications; and (H) automatic generation of SRS, TCS, and TM.

During requirements elicitation, a user can reuse a specific pattern (FRP, NFRP, or ATP), a set of patterns, or the complete catalogue. Os ATPs estão vinculados aos FRPs e/ou NFRs em um mesmo catálogo. This is achieved by accessing the "Associate Catalog" option available on the project's main page, as shown in Figure 1(i)D. A list of the catalogues registered into the DREAM database is exhibited so that the user choose one or more patterns.

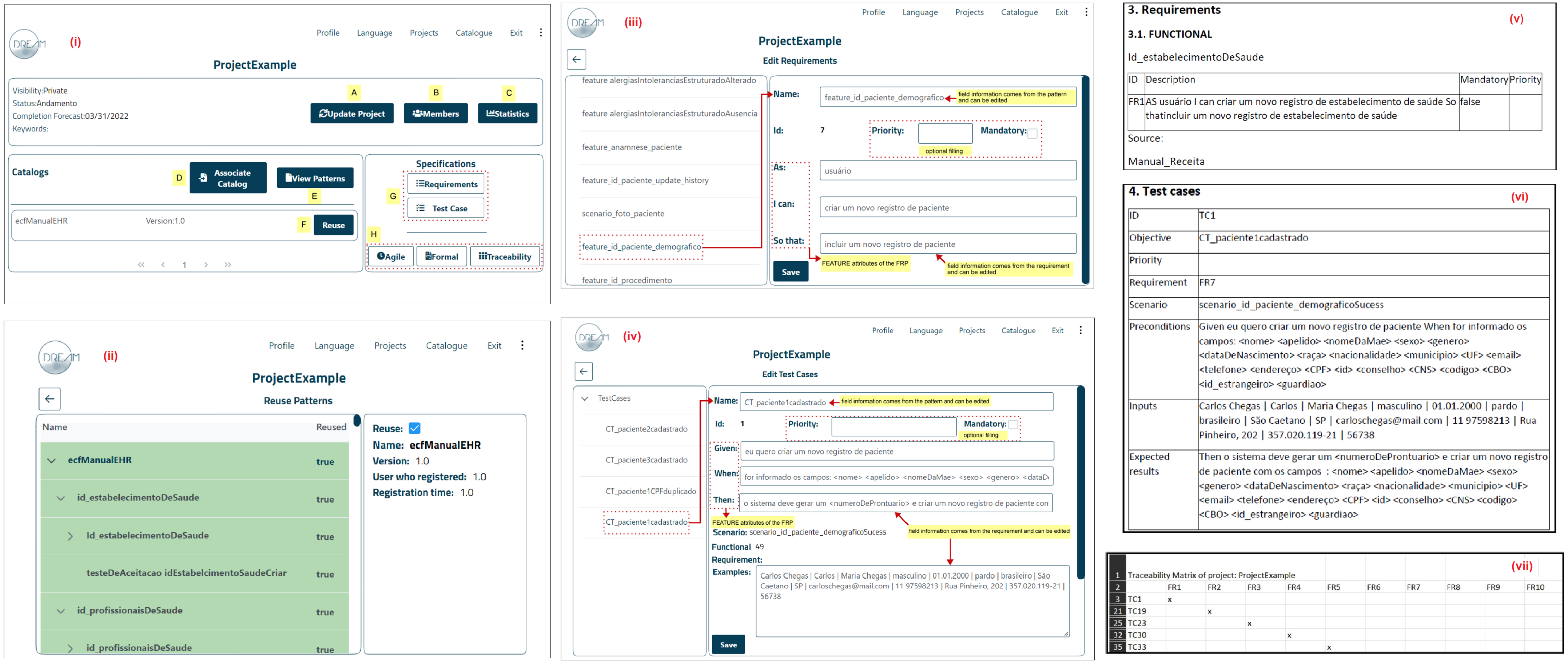
When clicking on the "Reuse" option, the user is redirected to the screen in Figure 1(ii), which contains detailed information about each element of the pattern catalogue (FRP, NFRP, or ATP). Reuse options include one or more patterns or the entire catalogue. We implemented such reuse options in DREAM to make requirements elicitation faster. The same effect is expected for test case elaboration when the user reuses ATPs linked to FRPs.

4.2. Supporting requirements and test cases specification

DREAM allows users to edit patterns reused according to a project's needs. The fields on the requirements editing screen, shown in Figure 1(iii), are automatically filled in with the reused patterns' content and can even be edited (except for the reserved words As, I can, So that.

Filling the Priority (1 to 5) and Mandatory (yes or no) fields is optional. The remaining fields (Name, As, I can, So that) are automatically filled by the information from the corresponding FRP reused. Their contents can also be edited by the user if necessary. Information about the source catalogue and the original pattern bag (NOT shown in Figure 1(iii)) can not be changed to maintain reuse traceability and generate software pattern usage reports.

Similarly, Figure 1(iv) illustrates the test case editing screen in which the fields Name, Given, When, Then, and Examples are extracted from the corresponding ATP reused. Likewise, the Scenario's name and the FR id come from the FRP-ATP aligned but are not editable.



Visibility:Private
Status:Andamento
Completion Forecast:03/31/2022
Keywords:

ecfManualEHR Version:1.0

Name	Reused
ecfManualEHR	true
id_estabelecimentoDeSaude	true
Id_estabelecimentoDeSaude	true
testeDeAceitacao idEstabelcimentoSaudeCriar	true
id_profissionaisDeSaude	true
id_profissionaisDeSaude	true

Reuse: ✓
Name: ecfManualEHR
Version: 1.0
User who registered: 1.0
Registration time: 1.0

feature alergiasIntoleranciasEstruturadoAlterado
feature alergiasIntoleranciasEstruturadoAusencia
feature_anamnese_paciente
feature_id_paciente_update_history
scenario_foto_paciente
feature_id_paciente_demografico
feature_id_procedimento

Name: feature_id_paciente_demografico ← field information comes from the pattern and can be edited
Id: 7 Priority: Mandatory: (optional filling)
As: usuário
I can: criar um novo registro de paciente
So that: incluir um novo registro de paciente
FEATURE attributes of the FRP
field information comes from the requirement and can be edited

CT_paciente2cadastrado
CT_paciente3cadastrado
CT_paciente1CPFduplicado
CT_paciente1cadastrado

Name: CT_paciente1cadastrado ← field information comes from the pattern and can be edited
Id: 1 Priority: Mandatory: (optional filling)
Given: eu quero criar um novo registro de paciente
When: for informado os campos: <nome> <apelido> <nomeDaMae> <sexo> <genero> <dataDe
Then: o sistema deve gerar um <numeroDeProntuario> e criar um novo registro de paciente con
FEATURE attributes of the FRP
field information comes from the requirement and can be edited
Scenario: scenario_id_paciente_demograficoSucess
Functional Requirement: 49
Examples: Carlos Chegas | Carlos | Maria Chegas | masculino | 01.01.2000 | pardo | brasileiro | São Caetano | SP | carloschegas@mail.com | 11 97598213 | Rua Pinheiro, 202 | 357.020.119-21 | 56738

3. Requirements

3.1. FUNCTIONAL

Id_estabelecimentoDeSaude

ID	Description	Mandatory	Priority
FR1	AS usuário I can criar um novo registro de estabelecimento de saúde So thatincluir um novo registro de estabelecimento de saúde	false	

Source:

Manual_Receita

4. Test cases

ID	TC1
Objective	CT_paciente1cadastrado
Priority	
Requirement	FR7
Scenario	scenario_id_paciente_demograficoSucess
Preconditions	Given eu quero criar um novo registro de paciente When for informado os campos: <nome> <apelido> <nomeDaMae> <sexo> <genero> <dataDeNascimento> <raça> <nacionalidade> <municipio> <UF> <email> <telefone> <endereço> <CPF> <id> <conselho> <CNS> <codigo> <CBO> <id_estrangeiro> <guardiao>
Inputs	Carlos Chegas | Carlos | Maria Chegas | masculino | 01.01.2000 | pardo | brasileiro | São Caetano | SP | carloschegas@mail.com | 11 97598213 | Rua Pinheiro, 202 | 357.020.119-21 | 56738
Expected results	Then o sistema deve gerar um <numeroDeProntuario> e criar um novo registro de paciente com os campos : <nome> <apelido> <nomeDaMae> <sexo> <genero> <dataDeNascimento> <raça> <nacionalidade> <municipio> <UF> <email> <telefone> <endereço> <CPF> <id> <conselho> <CNS> <codigo> <CBO> <id_estrangeiro> <guardiao>

1	Traceability Matrix of project: ProjectExample										
2		FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10
3	TC1	x									
21	TC19		x								
25	TC23			x							
32	TC30				x						
35	TC33					x					

Figure 1. The DREAM user interface: (i) main functionalities for a sample project; (ii) patterns reuse; (iii) requirements edition; (iv) test cases edition; and excerpts of (v) a requirements specification document, (vi) a test case specification document, and (vii) a traceability matrix.

4.3. Automatic documentation of software artifacts

Once a user has reused and edited the definitions of requirements and test cases from pattern catalogues, the DREAM tool also allows the automatic generation of software artifacts. For example, Figure 1(v) illustrates an excerpt of a requirements specification document describing an FR1 functional requirement for the registering of a new health institution. Specified as a user story, FR1 is not mandatory and its source is the electronic prescription section of our catalogue for EHR systems. This document structure, encoded in the .docx format, conforms to the ISO/IEC 29148 standard [ISO/IEC/IEEE 2018].

In turn, Figure 1(vi) depicts an excerpt of a test case specification document describing a TC1 test case for the registering of a new patient. The TC1 contents include preconditions, inputs, and expected results. In addition, observe that TC1 links to an FR7 functional requirement for tracing purposes. This document structure, encoded in .docx, adheres to the ISO/IEC 29119-3 standard [ISO/IEC/IEEE 2013].

Finally, DREAM can generate a traceability matrix document encoded in the .xlsx format. Figure 1(vii) presents the association between FR and TC following the relationships between FRPs and ATPs imported from pattern catalogues.

The generation of agile documentation is a work in progress – see H in Figure 1(i). We have produced a graphical view with individual cards with user stories and acceptance test scenarios to support professionals' tasks. This functionality will be available soon.

5. Final remarks and future work

The DREAM tool is innovative in relating different types of software patterns, i.e., FRP and ATP, to assist requirements and testing activities. None of the similar tools represents and processes the alignment between requirements and testing. In addition, DREAM maintains traceability between reused standards when generating documentation, allowing the management of these requirements throughout a project.

We are aware that, as a CASE tool, DREAM is at an early stage of the investigation. But, on the other hand, we claim we advance state of the art by developing tool support for our reuse approach based on requirement patterns linked to acceptance test patterns.

In future work, we will conduct experimental studies using DREAM in software requirements courses to evaluate to which extent it supports elicitation, specification, documentation, and even validation of requirements through acceptance tests. The goal is to gather the first pieces of evidence about the appropriateness of the DREAM-supported reuse approach. Experimentation results will also be used to identify the need for new features and promote the continuous improvement of the tool. We also plan to experiment with DREAM in professional scenarios later.

References

Barcelos, L. and Penteado, R. (2017). Elaboration of software requirements documents by means of patterns instantiation. *J Softw Eng Res Dev*, 5(3):1–23.

Barros-Justo, J. L., Benitti, F. B. V., and Leal, A. C. (2018). Software patterns and requirements engineering activities in real-world settings:a systematic mapping study. *Comp. Standards & Interfaces*, 58:23–42.

Beckers, K., Heisel, M., Côté, I., Goeke, L., and Güler, S. (2013). Structured pattern-based security requirements elicitation for clouds. In *2013 International Conference on Availability, Reliability and Security*, pages 465–474.

Benitti, F. and Silva, R. (2013). Evaluation of a systematic approach to requirements reuse. *Journal of Universal Computer Science*, 19:254.

Bjarnason, E. and Borg, M. (2017). Aligning requirements and testing: Working together toward the same goal. *IEEE Software*, 34(1):20–23.

Da Silva, R. and Benitti, F. (2013). Evaluation of a systematic approach to requirements reuse. *Journal of Universal Computer Science*, 19(2):254–280.

Irshad, M., Petersen, K., and Poulding, S. (2018). A systematic literature review of software requirements reuse approaches. *Inf. Softw. Technol.*, 93(C):223–245.

ISO/IEC/IEEE (2013). *29119-3:2013 Software and systems engineering - Software testing*. International Organization for Standardization, 5 edition.

ISO/IEC/IEEE (2018). *29148:2018 Systems and software engineering - Life cycle processes - Requirements engineering*. International Organization for Standardization, 2 edition.

Kiatake, L. G. G., Junior, L. A. V., da Silva, M. L., and Sanzovo, O. A. C. (2021). *Manual de Certificação de Sistemas de Registro Eletrônico em Saúde*. Sociedade Brasileira de Informática em Saúde. Versão 5.1. Instituído e regido pela Resolução CFM nº 1821/2007.

Kudo, T. N., Bulcão Neto, R. F., Macedo, A. A., and Vincenzi, A. M. R. (2019a). A revisited systematic literature mapping on the support of requirement patterns for the software development life cycle. *Journal of Software Engineering Research and Development*, 7:9:1–9:11.

Kudo, T. N., Bulcão Neto, R. F., and Vincenzi, A. M. R. (2019b). A conceptual metamodel to bridging requirement patterns to test patterns. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 155–160, New York, NY, USA. ACM.

Kudo, T. N., Bulcão-Neto, R. F., and Vincenzi, A. M. R. (2020). Uma ferramenta para construção de catálogos de padrões de requisitos com comportamento. In *Proceedings of WER20 - Workshop em Engenharia de Requisitos, São José dos Campos, Brazil, Agosto 24-28, 2020*, pages 1–14. Editora PUC-Rio.

Kudo, T. N., Bulcão-Neto, R. F., Graciano Neto, V. V., and Vincenzi, A. M. R. (2022). Aligning requirements and testing through metamodeling and patterns: design and evaluation. *Requirements Eng*, pages 1–19.

OMG (2002). Meta Object Facility (MOF) Specification, version 1.4. *Object Management Group, Inc.*

Palomares, C., Quer, C., and Franch, X. (2013). Pabre-proj: Applying patterns in requirements elicitation. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, pages 332–333.

Withall, S. (2007). *Software Requirement Patterns*. Best practices. Microsoft Press, Redmond, Washington.