Asia Pacific Journal of Contemporary Education and Communication Technology
ISBN (eBook): 978 0 9943656 8 2 | ISSN : 2205-6181
Year: 2016 , Volume: 2, Issue: 3

CONTEXT DEFINITION FOR BDD SCENARIOS UPON DEMO METHODOLOGY

Jiri Matula [a], Frantisek Hunka [b], Jaroslav Zacek [c]
[abc] University of Ostrava, Ostrava, Czech Republic
Corresponding email: jiri.matula@osu.cz

Abstract

Behaviour-driven development (BDD) methodology is the approach of how to keep track of the user's requirements during the software development. The paper deals with utilization of DEMO methodology to improve accuracy of context given in BDD scenarios. Ontological nature of transactions described in DEMO methodology helps to focus on production and coordination acts and facts in order to support important company's business processes. Integration of transactions in the form of user stories into BDD scenarios sets their context of feature definition upon ontological description of business in its existential essence. Thanks to domain-specific languages like Gherkin, a kind of BDD scenario is also executable and applicable in an automation process of software development. The beneficial consequence is the fact that developers are introduced to essential business goals and implemented features directly correspond with activities performed by employees in companies.

Keywords: Behaviour-driven Development, DEMO Methodology, User Requirements, Acceptance Testing

1.Introduction

An accurate definition of user requirements specification is the foundation stone of any good information systems. However, this part of software development is also very prone to errors and misunderstandings. According to Standish Group 2015 Chaos Report there is still significant number of failing software project - almost 19% - and this number is not decreasing (17% in 2014). There are many techniques adopted by agile approaches how to gather user requirements specification precisely. Rational Unified Process uses Use Cases and scenarios to control the development process to ensure that requirements are always in a first place (Use Case-driven approach). This technique visualizes a relationship only between an actor and the system without any other context (e.g. transactions, non-functional requirements) and this technique fits well for bigger information systems. Requirements of the gathering process in current methodologies (Scrum, Kanban) still rely on a one-way confirmation and inherently cannot provide instant automated feedback during software development. These methodologies have only one kind of feedback – user acceptance testing, in most cases performed manually by testers.

BDD methodology works well with a declarative approach. Therefore, this paper tries to deal with user requirements using declarative semantic. Using a declarative approach to describe business contracts can be found in Pesic & Van der Aalst, 2006 and authors use finite automata theory to simplify a relationship between elements. In another paper, authors use XML as a data source and brings a new extension to Courteous Logic Programs (Grosof, Labrou & Chan, 1999).

There are also attempts to use a semantic driven approach for user requirements verification (Gigante, Gargiulo & Ficco, 2015). However, the paper lacks necessary verification. Fully ontological approach can be found in Skjæveland et. al, 2015 to access generic data source. Some research tries to define a link between data mining and business process management (de Leoni, Maggi & van der Aalst, 2015). This paper specifically points the fact that constraints are described by a declarative process model. Authors also state that is possible to discover that the model is based on event data. However, if all states are not presented on the model (typically if unknown information system is being built with no best practices), proper technique is still missing to determine all states in small and middle size systems.

DEMO (Design & Engineering Methodology for Organizations) methodology (Dietz, 2006) defines an organization as a composition of people (social individuals) that perform two kinds of acts - production acts and coordination acts. The result of successfully performing a production act is a production fact. An example of a production fact may be that the package that has been delivered has been paid, or offered service has been accepted. All realization issues are fully abstracted out. Only the facts as such are relevant, not how they are achieved. The result of successfully performing a coordination act is a coordination fact. Examples of coordination acts are requesting and promising a production fact. Coordination and production acts and facts are arranged into a transaction pattern.

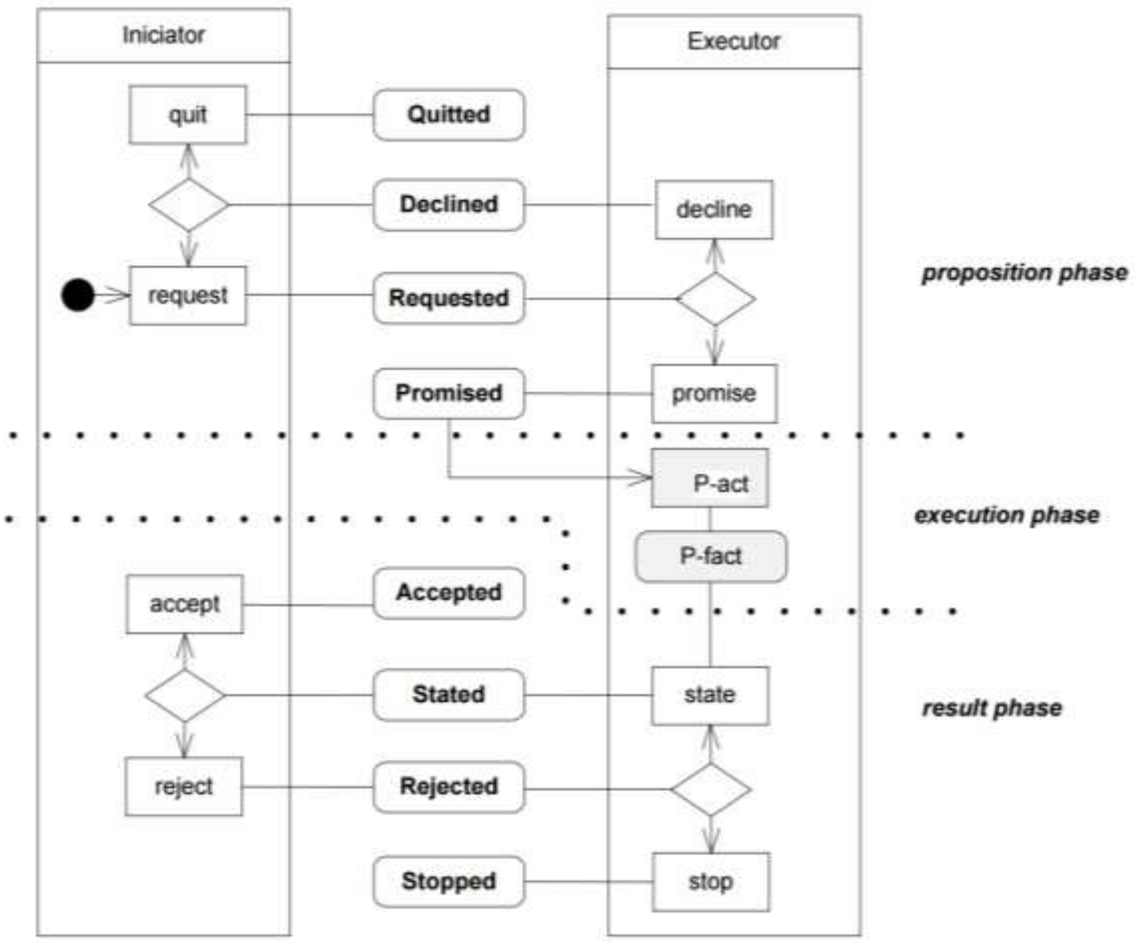

Figure 1: The standard transaction pattern. Source: Kervel (2012)

DEMO distinguishes the basic, standard and complete transaction patterns according to the numbers of transaction steps. The diagram in Fig.1 shows interrelated acts and facts (states) of the standard pattern. The partition of the initiator contains the coordination acts and the decisions are represented by diamonds in the diagram. The production act and the production fact are depicted in grey colour in the partition of the executor. The reason for locating the production fact in the executor partition is that the production is usually placed separately from the initiator partition. The coordination facts are situated in the middle of the figure as states in bold format. The complete transaction pattern is extended by four cancellation patterns regarding to the standard transaction pattern. The advantage of this methodology is completely defined as a state machine inside the transaction pattern. The all essential states are defined in underlying infrastructure.

The BDD technique which has been developed from the test-driven development technique utilizes principles of user stories and test driven development approach (Smart, 2014). At the same time, user stories technique is the foundation stone for the BDD testing scenario template, which is observable from a comparison of BDD and user story template below. User stories typically follow this recommended template (Cohn, 2004):

As a <type of user>, I want <some goal> so that <some reason>.

In comparison, the recommended template for BDD scenarios looks as following (Smart, 2014):

Feature [title]

    In order to [benefit]
    As [role]
    I want [feature]
    Scenario: [title]
        Given [context]
        And [some more context]
        ...
        When [some event occurs]
        And [some other event]
        ...
        Then [outcome]
        And [some other outcome]
        ...
        Scenario: [title]
        ...

Figure 2: Standard BDD scenario template.

There is a strong similarity between these two templates. In the authors' previous research, the method has been presented which allows to transform transaction into the form of user stories upon DEMO methodology. The modified version of the template for user stories which utilizes transactions defined in DEMO methodology is following:

As an <initiator/executor>, I perform a task in <transaction> so that <result of transaction>.

Such defined user stories have their ontological origin. The user type is every time explicitly defined, which excludes misinterpretation of intended user type. Particular transactions are coupled with real business processes, which prevents inclusion of artificially constructed processes or user's wishes unrelated to ontological essence of business. In addition, particular transactions strictly define expected results from the process. Eventually, additional evaluation criteria for the results of transactions might be set. Derived user stories in accordance with

DEMO methodology fulfil C4E qualities (coherent, consistent, comprehensive, concise and essence). Also, they always relate to a concrete business process and an actor.

2. Giving context to BDD scenarios

A previously mentioned fact that user stories are part of BDD scenario template gives an opportunity to apply modified version template into a BDD template scenario. In the context of user stories in the form of transactions, proposed modified version of template for BDD scenario looks as follows:

```
Feature [title] - [transaction ID]
        In order to [outcome of the transaction]
        As [initiator/executor]
        I perform task in a [transaction]
        Scenario: [title]
                Given [context]
                And [some other context]
                ...
                When [some event occurs]
                And [some other event]
                ...
                Then [outcome]
                And [some other outcome]
                ...
        Scenario: [title]
                ...
        ...
```

Figure 3: Modified BDD scenario template.

Role has been replaced for executor or initiator who takes a part in the transaction. Particular scenarios describe the business situation with aim to fulfil business goals denoted as outcomes of transactions. All the scenarios have to respect user story given in the feature description.

As an explanatory case study which can be used is a company which delivers packages to their customers. BDD methodology does not strictly recommend how to specify user story for feature description. In case of proposed concept, there exists only one proper definition of user story represented as transaction which is composed into BDD scenario. This determines context for the scenarios given in feature description. According to proposed concept, the BDD scenario is as follows:

```
Feature Package delivery – T01
      In order to package being delivered.
      As driver
      I want to deliver a package to recipient.
      Scenario: Planning the route to destination
                Given I have a list of addresses scheduled for :day
                When I choose the closest available customer
                Then I am able to find the optimum route to destination via Google Maps
```

Figure 4: Example of real BDD scenario according to proposal.

Also, such definition follows the syntax of domain-specific language Gherkin and it is executable via BDD testing frameworks like Behat, Cucumber and others. For further illustration, the output from testing framework Behat is following:

```
Feature: Package delivery – T01
        In order to package being delivered.
        As messenger
        I want to deliver a package to recipient.

        Scenario: Planning the route to destination
        # features/delivery.feature:6
                Given I have list of addresses for scheduled for today
                # FeatureContext::IHaveListOfAddresessFor ScheduledForToday()
                When I choose the closest address for the delivery
                # FeatureContext::iChooseTheClosestAddress
                Then I am able to find the fastest route to destination via Google Maps
                # FeatureContext::iAmAbleToFindTheOptimumRouteToDestinationViaGoogleMaps()

1 scenario (1 passed)
3 steps (3 passed)
0m0.01s (9.55Mb)
```

Figure 5: Output from Behat BDD testing framework executed over the previous BDD scenario.

Feature scenarios are validated against production code. It ensures that production code follows activities in company business and it is tested within every build pushed on the server.

3.Discussion

Derived user stories, in accordance with DEMO methodology, do not have duplicities nor contradictions and always are addressed to a definite business process and actor. All these qualities are included also in BDD scenario. User stories without DEMO analysis do not strictly differentiate user types. For example, a role client and potential client are often as the same role customer. In general, this may easily lead to confusion of roles and cause undesirable refactoring due to an incorrectly modelled role in information systems. Especially when developers are not familiar with entities which are using the information system.

Integration of transactions defined in DEMO methodology into BDD feature scenarios improves accuracy of the context in BDD testing scenarios due to its relation to existing coordination and production acts and facts. Nevertheless, ontological nature of defined transaction presumes an existence of essential business processes. Hence, the proposed method is suitable, especially for development of software products, which support business processes in companies. Once the transactions became a part of BDD scenario it involves the developer or analyst to understand purpose why the feature is implemented. Also, it sets boundaries for the particular BDD scenarios which are consequently linked to existing essence of the business. Thanks to fact that the BDD template structure of feature description is compatible with domain-specific language Gherkin which allows an automatic verification of user requirements against production code. On the other hand, analysis of transactions requires a deep analysis of business processes in company. If the analysis is not performed in accordance with DEMO methodology, it may lead to the same result as in the case of user stories, which are written purely up to recommendations of the agile community. Also, it comes to considerations maps particular states of transactions directly to scenarios steps defined in BDD which will be the objective of further research.

Acknowledgment

The paper was supported by the grant provided by Ministry of Education, Youth and Sport Czech Republic, reference no. SGS15/PRF2016 and no. SGS03/UVAFM/16.

References

i. Cohn, M., 2004. User Stories Applied: For Agile Software Development. Boston: Addison-Wesley, pp. 31-41.

ii. de Leoni, M., Maggi, F. M. & van der Aalst, W. M., 2015. *An Alignment-Based Framework to Check the Conformance of Declarative Process Models and To Pre-Process Event-Log Data*. Information Systems, pp. 258-277.

iii. Dietz, J. L. G., 2006. *Enterprise Ontology: Theory and Methodology*. New York: Springer, pp. 16-31.

iv. Gigante, G., Gargiulo, F. & Ficco, M., 2015. A Semantic Driven Approach for Requirements Verification,. *Intelligent Distributed Computing*, 8, pp. 427-436.

v. Grosof, B. N., Labrou, Y. & Chan, H. Y., 1999. *A Declarative Approach To Business Rules in Contracts: Courteous Logic Programs in XML*. Proceedings of the 1st ACM Conference on Electronic Commerce, pp. 68-77.

vi. Pesic, M. & van der Aalst, W. M., 2006. *A Declarative Approach for Flexible Business Processes Management, Business Process Management Workshops*. Springer Berlin Heidelberg, pp. 169-180.

vii. Skjæveland, M. G., Giese, M., Hovland, D., Lian, E. H. & Waaler, A., 2015. *Engineering Ontology-Based Access to Real-World Data Sources*. Web Semantics: Science, Services and Agents on the World Wide Web, pp. 112-140.

viii. Smart, J. F., 2014. *BDD in Action: Behavior-Driven Development For The Whole Software Lifecycle*. New York: Manning Publications Company, pp. 3-32.

ix. Van Kervel, S. J. H., 2012. *Ontology driven Enterprise Information Systems Engineering*. Doctoral Dissertation, SIKS Dissertation series nr. 2012-50, Delft University of Technology.