DOI: 10.15439/2017F366
ISSN 2300-5963 ACSIS, Vol. 12

Identification of Business Relevant Features in Information

Jiri Matula, Jaroslav Zacek
University of Ostrava,
Faculty of Science, Dvorakova 7,
Ostrava 701 03, Czech Republic
Email: {jiri.matula,
jaroslav.zacek}@osu.cz

Abstract—**The paper is devoted to the utilization of DEMO enterprise ontology (Design & Engineering Methodology for Organizations) for refactoring purposes in software development. The main contribution of the paper resides in presentation of the method which interconnects ontological models of business processes with information system features implementation. Also, it allows the evaluation of their relevancy for enterprise. In contrast to other methods based on best practices, the proposal uses ontological description of business processes defined upon the theory in DEMO methodology. This makes the proposal unique compared to other approaches. Moreover, it provides a clear differentiation of features which are important for performing tasks by employees in company.**

I. Introduction

GENERALLY, information systems (IS) provide information for task execution of many entities living in enterprise. Nevertheless, development of information systems is very prone to errors and challenges during its whole lifecycle. As a company develops in time and changes its internal rules and roles, information systems may tend to get old and some features do not fit to company needs. This fact often leads to refactoring, whereas developers are not able to satisfy user requirements due to limitations of the current implemented solution.

There are many techniques adopted by agile approaches how to gather and define user requirements for refactoring of information systems. Rational Unified Process uses Use Cases and scenarios to control the development process to ensure that requirements are always in 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 gathering process in current methodologies (Scrum, Kanban) still relies 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 (Behavior-Driven Development) technique allows to get automatic feedback and works well with a declarative approach. Using a declarative approach to describe business contracts can be found in [7] 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 [8]. There are also attempts to use a semantic driven approach for user requirements verification [11]. However, this approach lacks the necessary verification. Some research tries to define a link between data mining and business process management [9]. This paper specifically points to the fact that constraints are described by a declarative process model. Authors also state that is possible to discover this model based on event data. However, if all states are not presented on the model (typically if unknown information system is being built without best practices), the correct technique is still missing to determine all states in small and middle size systems. A fully ontological approach can be found in [10] to access generic data source. In comparison, the DEMO methodology utilizes theoretical foundations to describe business processes, which makes this approach unique compared to the above mentioned approaches because they are mostly based on best practices.

The combination of BDD technique and DEMO methodology allows to link ontological descriptions of business processes directly to production code with the possibility of testing automation in a continuous integration process. Interconnection of information system features and coordination or production acts makes possible to determine which features of the information system can be removed, newly implemented, or refactored according to ontological descriptions derived upon DEMO methodology. Thanks to this fact, a new method is presented. It identifies features of information systems which are important for the execution of coordination and production acts performed by employees in companies.

II. Transactions in DEMO Methodology

DEMO methodology [3] defines an organization as a composition of people (social individuals) that perform two kinds of acts – production 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.

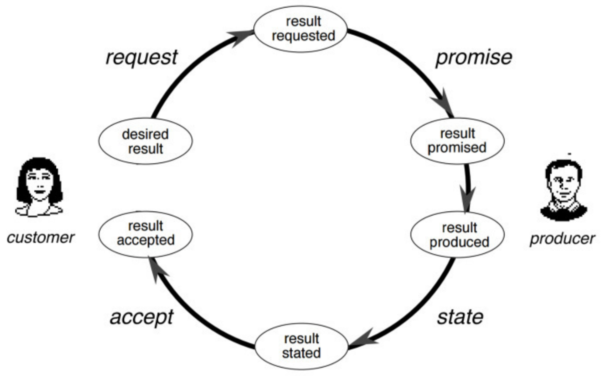

Fig. 1 Basic transaction pattern. Source: [3]

Transaction pattern states that there are always two roles in a transaction, initiator (customer) and executor (producer). Initiator is someone who has a request and executor is responsible for fullfilling initiator needs. More detailed explanation of transaction pattern is depicted in the Fig. 2. White rectangles represents coordination acts, white rounded rectangle is used for coordination fact. Production act is depicted as grey rectangle. Gray rounded rectangles stands for production fact. The lifespan of every transaction has three phases – order (proposition), execution and result phase.

In the order phase, the initiator and the executor work to reach an agreement about the intended result of the transaction, i.e., the production fact that the executor is going to create as well as the intended time of creation. In the execution phase, this production fact is actually brought about by the executor. In the results phase, initiator accepts or rejects result (production fact) of the transaction [3].

According to DEMO methodology, it is possible to analyze gathered text descriptions of business processes of a company and extract transactions which represent ontological essence of the enterprise [6]. These transactions can be served as a source of information for revising features during refactoring process and they also form the theoretical basis which is implemented using BDD technique. Consequently, these specifications can be executed via DSL languages (Domain Specific Language) like Cucumber, Behat, etc.

III. Conversion of Transactions to BDD scenarios

The BDD technique which has been developed from the test-driven development technique utilizes principles of user stories and test-driven development approach [2]. User stories typically follow this recommended template.

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

Fig. 3 User story template. Source: [1]

At the same time, user stories technique is the foundation stone for the BDD testing scenario template, which is observable from a comparison of user story template above and BDD scenario template below.

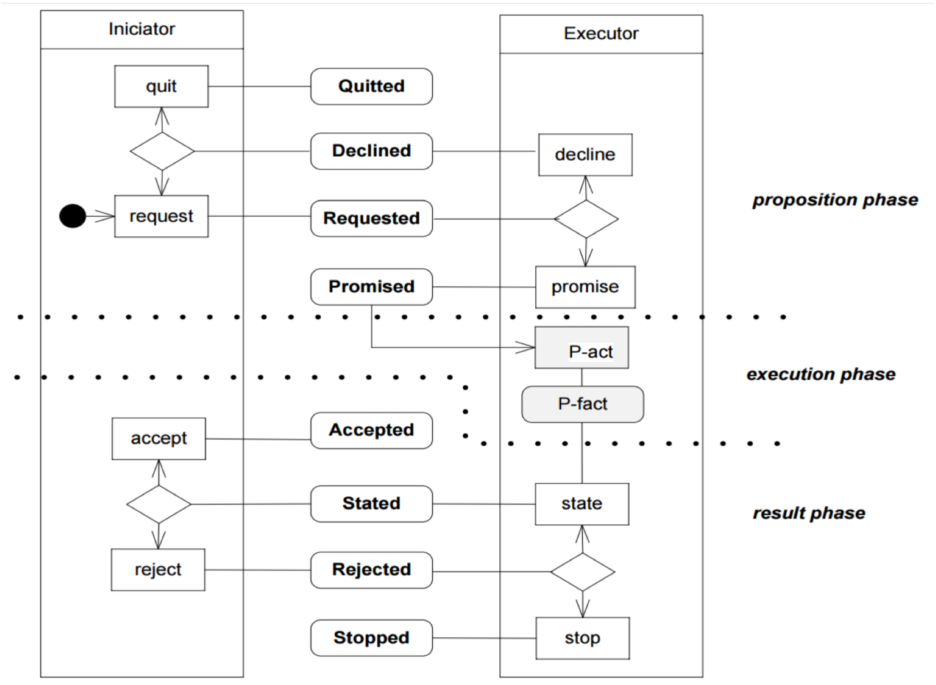

Fig. 2 Detailed view of transaction pattern. Source: [3]

Feature [title]
In order to [benefit]
As [role]
I want [feature]
Scenario: [title]
Given [context]
And [some more context]
...
When [an event occurs]
And [a further event]
…
Then [outcome]
And [another outcome]
...
Scenario: [title]
…

Fig. 4 Standard BDD scenario template

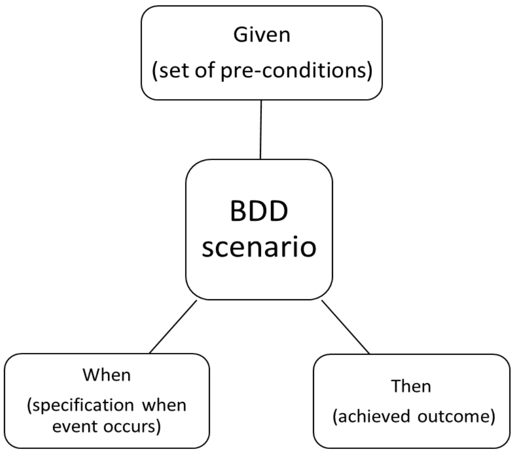

Fig. 5 Composition of BDD scenario

A previously mentioned fact is that user stories are part of BDD scenario template give 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 following.

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

Fig. 6 Modified template for user story. Adapted from: [4]

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

Modified template structure starts with a feature title which is linked to related transaction unique ID. After feature identification, it is necessary to perform the next step – the outcome of the transaction – which is going to be fulfilled when the transaction is completed. So far, context for coordination/production act is defined and follows a transformation text description of coordination or production act into the form of BDD scenarios. The scenario part should have covered all possible situations during the execution of the coordination/production act. All transformed coordination/production acts into BDD scenarios must have its reference in gathered text descriptions of business processes. Every coordination/production act must result in coordination/production fact.

Feature [title] – [transaction ID]
In order to [coordination/production fact]
As [initiator/executor]
I want to perform coordination/production act in [transaction]
Scenario: [title]
Given [context]
And [another context]
...
When [an event occurs]
And [another event]
...
Then [result – coordination/production fact]
...
Scenario: [title]
…

Fig. 7 Modified BDD scenario template. Adapted from: [5]

As an explanatory case is used a company where the messenger (executor) delivers packages to their customers. A company (initiator) usually comes with a request to perform a delivery. The initiator and executor performs coordination and production acts in order to deliver a package. These coordination and production acts are expressed in the scenario part of the modified BDD template. Messenger's daily schedule includes the list of the addresess where is necessary to make a delivery. The messenger picks the closest customer and asks about his availbility. When customer approves the delivery, the messenger plans a route to the destination and delivers a package. According to proposed concept the BDD scenario looks as following:

Feature Package delivery – T01
In order to deliver a package.
As messenger (executor)
I want to plan route to destination
Scenario: Planning route to destination
Given I have a list of addresses scheduled for today
When I choose the closest address for the delivery
Then I can find the optimum route to destination via Google Maps

Fig. 8 Example of BDD scenario according to modified template

BDD methodology itself does not strictly recommend how to specify user story for feature description. For the modified approach only one proper definition exists of user story represented by complement transaction composed into BDD scenario. This determines context for the scenarios given in feature description.

BDD scenarios can be validated against production code. They ensure that production code follows activities in company business processes. Also, BDD scenarios are executable and its verification is possible with every upcoming change of information system within continuous integration. The example from Behat framework for the previous BDD scenario is depicted in the Fig. 9.

```
Feature: Package delivery – T01
    In order to deliver a package.
    As messenger
    I want to plan route to destination

Scenario: Planning route to destination
#features/planning.feature:6
    Given I have list of addresses for scheduled for today
    #FeatureContext::iHaveListOfAddressesForScheduledF
    orToday()
    When I choose the closest address for the delivery
    #FeatureContext::iChooseTheClosestAddressForTheDe
    livery()
    Then I can find the optimum route to destination via
    Google Maps
    #FeatureContext::iCanFindTheOptimumRouteToDesti
    nationViaGoogleMaps()
1 scenario (1 passed)
3 steps (3 passed)
0m 0.01s (9.55Mb)
```

Fig. 9 Output from Behat testing framework after execution of story derived from DEMO transaction

IV. Method for Identification of Business Relevant Features

DEMO transactions and BDD scenarios are foundation stones for the proposed method which evaluates relevancy of information systems features. Each step of the method in the list below will be explained in this chapter.

The method includes following tasks:

1. Identification of transactions according to DEMO methodology.
2. Convert identified transactions into the form of BDD scenarios.
3. Map BDD scenarios to current implementation of features.
4. Identify supported and unsupported coordination and production acts by the information system.
5. Identify features of IS to be removed or refactored due to inconsistency to its ontological description.

As an explanatory case is used a brief description of existing company in the Fig. 10.

The company supply of electricity for customers and offers "smart measuring" service which makes them a possibility to monitor the consumption of electricity online. Customers have provided the information system which **reports electricity consumption and savings for each period (T01)**. *Measuring devices broadcast consumption data. This data is stored to database (C01).*

At the beginning, a client contacts the company and *salesman gives to a potential customer a detailed overview about offered services (C02).* **When a client signs a contract (T02),** *contract details are entered into the IS (C03).* Consequently, *manufacturing of devices is requested (C04).* The device manufactory department has their own employees and stock of material. Upon the contract, **device arrangements are complemented (T03)**. Once devices are prepared to expedition, *the service department is notified about necessity of installation contracted devices* (C05). Firstly, installation place is examined by technician who will *decide whether installation is feasible (C06).* After that, **installations of appliances are planned (T04)**. Planning of appliance installation is a complex process which *considers availability of company cars (C07), booking of accommodation (C08), customer confirmation and skills of technicians (C09). The manager also assigns a specific task to technicians if the customer is available.* **Once the device is installed (T05)**, a *customer signs the montage sheet (C10).* When the *installation of devices is confirmed (C11),* a new client is *entered to information system (C12)* and contracted **services starts to be billed (T06)**.

Fig. 10 Text description of company business processes

In the first step of procedure, several transactions have been identified. In the Fig. 10, bold text refers to production facts and blue italic to coordination and production acts. Each transaction consists of coordination and production acts.

- **T01** – Consumption of electricity is reported for each period.
- **T02** – Client signed contract.
- **T03** – Devices arrangements are complemented.
- **T04** – Appliance installation is planned.
- **T05** – Contracted devices are installed.
- **T06** – Services started to be billed.

The second step requires conversion of transactions to BDD scenarios. The case example for the transaction **T04** is depicted in the Fig. 11.

```
Feature Package delivery – T04
    In order to plan the appliance installation
    As manager
    I want to assign task to the technician
    Scenario Outline: Task planning
        Given manager has chosen <date>
        And car is available on <date>
        And technician has no others task on <date>
        When customer confirmed availability on <date>
        Then task is assigned to technician
        Examples:
        | date         |
        | 2017-05-06   |
        | 2017-05-07   |
```

Fig. 11 Converted coordination act defined in the transaction **T04** (Gherkin DSL language syntax)

In the third step, converted transactions into BDD scenarios are mapped to production code. This is usually done via frameworks like Cucumber, Behat and the others. Authors recommend to follow instructions for the chosen framework.

```
/**
 * @Given car is available on :date
 */
public function carIsAvailableOn($date)
{
  assertEquals(true,
    $this->carpark->
    hasAvailability($date));
}


/**
 * @Given technician has no others task
 * on :date
 */
public function
technicianHasNoOthersTaskOn($date)
{
  assertEquals(true, $this->
    technician->isAvailable($date));
}


/**
 * @When customer confirmed availability
 * on :date
 */
public function
customerConfirmedAvailability($date)
{
  assertEquals($date,
    $this->order->installDate);
}

/**
 * @Then task is assigned to technician
 */
public function
taskIsAssignedToTechnician()
{
  assertEquals(true, count(
    $this->technician->tasks) > 0);
}
```

Fig. 12 Example of mapping of modified BDD scenarios to production code (Behat framework implementation)

The fourth step identifies supported (green) and unsupported (red) coordination and production acts by the information system. Unsupported coordination/production act means that it has no reference to any BDD scenarios which have been successfully mapped in the previous step. Result of the fourth step is depicted in the Fig. 13.

The company provides supply of electricity for customers and offers "smart measuring" service which makes it possible for them to monitor their consumption of electricity online. Customers have provided the information system which **reports electricity consumption and savings for each period (T01)**. *Measuring devices broadcast consumption data. This data is stored to database and verified (C01).*

At the beginning, a client contacts the company and *salesman gives to a potential customer a detailed overview about offered services (C02).* **When a client signs a contract (T02),** *contract details are entered into the IS (C03).* Consequently, *manufacturing of devices is requested (C04).* The device manufactory department has their own employees and stock of material. Upon the contract, **device arrangements are complemented (T03)**. Once devices are prepared to expedition, *the service department is notified about necessity of installation of the contracted devices (C05).* Firstly, installation place is examined by technician who will *decide whether installation is feasible (C06)*. After that, **installations of appliances are planned (T04)**. Planning of appliance installation is a complex process which *considers availability of company cars (C07), booking of accommodation (C08), customer confirmation and skills of technicians. Manager also assigns a concrete task to technicians (C09).* **Once the device is installed (T05)**, a *customer signs the prepared montage sheet (C10).* When the *installation of devices is confirmed (C11),* a *new client is entered to information system (C12)* and **contracted services starts to be billed (T06)**.

Fig. 13 Differentiation of supported and unsupported coordination/production acts

Unsupported acts are listed for further investigation of whether such acts could be automated or supported by information systems. Thereby, highly relevant information is discovered for development of future features to support business processes in the company.

Summary of unsupported coordination/production acts:

- Request for device manufacturing.
- Providing information about offered services to customer.
- Accommodation booking process.
- Notification about appliance installation.
- Making decision whether installation is feasible.
- Checking availability of cars in a company car park.
- Preparation of montage sheet to be signed by a customer.

The fifth step involves the identification of feature specifications (user story, use case, etc.) which are not included in any transaction. In other words, they have no reference to user stories (coordination/production acts) mentioned in mapped BDD scenarios. These features are candidates to be either refactored or removed from the information system. Unfortunately, this should be consulted with product owner, or domain expert. Some functionalities might be foundation elements for the information system, for example user administration. Features which are included in coordination/production acts defined in BDD scenario are linked by identification number from the text description. Feature specifications might differ from project to project. Presented case uses user scenarios technique. The example is depicted below.

- UC 1 Appliance installation evidence.
  - US 1.1 Customers [C12, C09]
  - US 1.2 Reporting from installation [C01]
- UC 2 Device management
  - US 2.1 Data broadcast testing [C01]
  - US 2.2 Remote reset [ref is missing]
- UC 3 Planning of appliance installations
  - US 3.1 Task planning [C03]
    - US 3.1.1 Customer confirmation [ref is missing]
    - US 3.1.2 Technician skill evidence [ref is missing]
  - US 3.2 Technician utilization overview [C09]
- UC 4 Announcements
  - US 4.1 News board [ref is missing].
  - US 4.2 Personal messages [ref is missing]

Fig. 14 Identification of business irrelevant features

V. Discussion and Future Research

Ontological nature of transactions presumes an existence of essential business processes. Hence, the proposed method is suitable especially for development of software products which supports business processes in companies. Once the transactions became a part of BDD scenarios, it involves the developer or analyst to understand purpose why the feature is implemented. Also, it sets boundaries for the BDD scenarios which are consequently linked to the existing essence of the business.

The most important benefits of proposed approach:

1. Text descriptions for derivation of transactions are humanly-readable, hence there is no problem to have descriptions validated by employees in company.
2. The method detects instantly which coordination and production acts are supported by information system.
3. Also, it finds useless feature which is possible to remove or not maintain anymore.
4. Production code base, respectively features are linked to ontological description of business processes.
5. Instant feedback through automated testing.

Contrary, the method does not work well with the development of software which is not going to support business processes. Another problem is the fact that companies evolve over time and change their business processes. Therefore, text descriptions need to be updated as development goes on. Consequently, it is necessary to propagate modifications into scenarios and information system to avoid technical debts. Unfortunately, this will always stay up to responsibility of the company and software developers.

The identification of feature relevance is one of the most challenging part in the refactoring process, especially for applications with huge technical debt, which desire radical refactoring where is hardly possible to save all existing features in order to settle technical debt quickly. In addition, relevancy of features for business intentions is critical in terms of further investments for already existing information systems. Authors also work as software developers and due to their experience, irrelevant features are not something that rarely occur in the implementation of any information system.

The presented method also discovered another important finding. The action model defined in DEMO methodology represents internal business rules for coordination and production acts. Such business rules should be reflected in the information systems. The example of action model definitions is depicted below.

```
on stated T02(M,Y)

    if <installation is feasible> ≥ accept T02(M,Y)
    €not <installation is feasible> ≥ reject T02(M,Y)
    fi

    if <some other condition> ≥ reject T02(M,Y)
    €not <some other condition> ≥ state T03(M)
    fi

no
```

Fig. 15 Action model example

These definitions are also executable. Nevertheless, the question remains whether it is also possible to include them into BDD scenarios and verify during software testing. This will be an objective of further research.

VI. Conclusion

The main contribution of the paper resides in presentation of method which interconnects ontological models of business processes directly to information system features implementation and allows to evaluate their relevancy for enterprise, whereby it provides a clear differentiation of features which are or not important for performing tasks by employees in enterprise. This method could help to reduce the technical debt of current information systems and identify main endpoints and interfaces that are candidates for refactoring.

Acknowledgment

The paper was supported by the grant provided by Ministry of Education, Youth and Sport Czech Republic, reference no. SGS15/PRF/2017.

References

[1] M. Cohn, User stories applied: for agile software development. Boston: Addison-Wesley, 2004, pp. 31–41.

[2] J. F. Smart, BDD in action: Behavior-Driven development for the whole software lifecycle. New York: Manning Publications Company, 2014, pp. 3–32.

[3] J. L. G. Dietz, Enterprise ontology: theory and methodology. New York: Springer, 2006, pp. 16–31.

[4] J. Zacek, J. Matula, and F. Hunka, Context definition for BDD scenarios upon DEMO methodology, 2nd International Conference on Theory and Practice, Sia Pacific Institute of Advanced Research, 2016, pp. 164–169, ISBN 9780994365613.

[5] J. Matula, J. Zacek, and F. Hunka, Relevant User Stories by Using DEMO Analysis, Proceedings of the 11th Scientific Conference Internet in the Information Society, University of Dabrowa Górnicza, Cieplaka, 2016. pp. 21 –30. ISBN 9788365621009.

[6] S. J. H. Van Kervel, Ontology driven enterprise information systems engineering, 2012.

[7] M. Pesic and W. M. Van der Aalst, A declarative approach for flexible business processes management, Business Process Management Workshops, Springer Berlin Heidelberg, 2006, pp. 169–180.

[8] B. N. Grosof, Y. Labrou, and H. Y. Chan, A declarative approach to business rules in contracts: courteous logic programs in XML, Proceedings of the 1st ACM conference on Electronic commerce, 1999, pp. 68– 77.

[9] M. de Leoni, F. M. Maggi, and W. M. van der Aalst, An alignment-based framework to check the conformance of declarative process models and to pre-process event-log data, Information Systems, 2015, pp. 258–277.

[10] M. G. Skjæveland, M. Giese, D. Hovland, E. H. Lian, and A. Waaler, Engineering ontology-based access to real-world data sources, Web Semantics: Science, Services and Agents on the World Wide Web, 2015, pp. 112–140.

[11] G. Gigante, F. Gargiulo, and M. Ficco, A semantic driven approach for requirements verification, Intelligent Distributed Computing VIII, 2015, pp. 427–436.