



Open Archive TOULOUSE Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in : <http://oatao.univ-toulouse.fr/>
Eprints ID : 18848

The contribution was presented at WISE 2016:
<http://www.wise-conferences.org/2016/>

To cite this version : Valente, Pedro and Rocha Silva, Thiago and Winckler, Marco Antonio and Nuno, Nunes *Bridging Enterprise and Software Engineering Through an User-Centered Design Perspective*. (2016) In: International Conference On Web Information Systems Engineering (WISE 2016), 7 November 2016 - 10 November 2016 (Shanghai, China).

Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr

Bridging Enterprise and Software Engineering Through an User-Centered Design Perspective

Pedro Valente^{1,2(✉)}, Thiago Silva¹, Marco Winckler¹,
and Nuno Nunes²

¹ Institut de Recherche en Informatique de Toulouse (IRIT),
Université Paul Sabatier, Route de Narbonne, 118, 31400 Toulouse, France
pvalente@uma.pt, {rocha,winckler}@irit.fr

² Madeira Interactive Technologies Institute (MITI), University of Madeira,
Caminho da Penteada, 9020-105 Funchal, Portugal
njn@uma.pt

Abstract. The development of Web-based Information Systems is crucial in the quest to maintain and develop the enterprise competitiveness. However, capturing requirements from Business Processes (BP) is still an issue, as existing methods mostly focus, or on human aspects and the user interface, or on business concerns as rules and workflow coordination, and therefore do not specify all the Software Architectural components which are relevant for software development. We present the Goals Approach, which analyzes BPs and User Tasks and details them in the process of methodically designing and structuring the User Interface, the Business Logic and the Database of the Information System given a Model-View-Controller (MVC) architectural pattern. In this paper we focus on how to obtain the Goals business model of requirements based on the DEMO method. The approach can be used for in-house software development, and the method is straightforward fitting Small and Medium Enterprises agility needs.

Keywords: Web-based applications · Enterprise engineering · Software engineering · User-Centered Design · Software architecture

1 Introduction

Software development within enterprises still lacks effectiveness as project full-success rates are still as low as about 30 % [1, 2]. Despite this fact, efforts in SE have at least taken us from a chaotic state of the practice [3], to a more inspiring situation where expertized executive management support, user involvement in the development process and agile techniques are appointed as factors of project success [4, 5].






In our quest to integrate the enterprise and the software engineering perspectives as a solution to align business and Information Technology (IT) and create the conditions to increase software success rates, we bridge both domains by means of a User-Centered Design perspective that allows the modeling of Web-based applications. We presents the Goals Approach, which models the business and uses it as the back-bone of the software architecture. This paper focuses on the business model elaboration from DEMO [6], as a way to enhance business analysis performance and explain our method.

1.1 Foundations and Software Development Process

The Goals Approach is founded on five methods: Wisdom [8], which is a software engineering and architectural method; Goals [9], which establishes a relation between business and software architectures; DEMO [6], that models the enterprise by means of an ontology; Activity Modeling (AM) [10], which models human activity and designs the user interface, and BDD [11], which models user interface and system behavior.






The Software Development Process defines a method that integrates the Enterprise Engineering and Software Engineering perspectives, concerning a given Business Process Improvement (BPI) [12], in two phases. The Analysis Phase identifies Business Processes (Step 1), User Tasks (Step 2), Interaction Spaces (Step 3), Business Rules (Step 4) and Data Entities (Step 5), composing an Enterprise Structure of business requirements, which components are presented in Table 1.

Table 1. Enterprise structure's component's definition, origin and symbol.

Component	Brief Definition	Origin	Symbol
Business Process (BP)	<i>A Network of UTs that lead to a Goal</i>	DEMO	
User Task (UT)	<i>A Complete Task within a BP</i>	AM	
Interaction Space (IS)	<i>The Space that supports a UT with the same BRs and DEs.</i>	Wisdom	
Business Rule (BR)	<i>A Restriction on the DE's Structural Relations</i>	DEMO	
Data Entity (DE)	<i>Persistent Information about a Business Concept</i>	Wisdom	

The Design Phase applies a User-Centered Design perspective to the Enterprise Structure in order to specify User Tasks (Step 6), design the User Interface (Step 7), structure the Business Logic (Step 8) and the Database (Step 9), finishing (Step 10) by elaborating the Software Architecture based on a MVC architectural pattern [13] in order to support for any possible combination of BPs that may structure the enterprise service.

Table 2. Software Architecture components definition, origin and symbol.

Component	Definition	Origin	Symbol
Aggregation Space	<i>A User Interface</i>	Hydra	
Interaction Component	<i>Tool of a User Interface</i>	Goals	
Interaction Object	<i>A User Interface Object that triggers SRs</i>	Goals	
User Interface SR	<i>A SR that provides support for User Interface presentation</i>	Goals	
Database SR	<i>A SR that manages Data Entities</i>	Goals	

Each Software Architecture component is presented in Table 2, where SR stands for System Responsibility. The Software Architecture is elaborated by means of composing one Aggregation Spaces [14] per each User Task (UT), which architecturally uses the ISs which are associated to the UT, ensuring the application of BRs over identified DEs and ensuring traceability between business and software implementation.

Goals establishes a relation with DEMO by means of the concepts of BP, UT, BR and DE which are compatible with the DEMO concepts of Transaction, Coordination Act, Action Rule and Object Class, respectively. Goals adds the Interaction Space (IS) which as the key to build-up the Enterprise Structure. We define three patterns of derivation (A, B and C) which are used to identify Goals component from DEMO models. The patterns are introduced in Fig. 1, and Steps 1 to 5 which explain the derivation of components in a top-down process are presented in Sects. 2.1, 2.2, 2.3, 2.4 and 2.5.

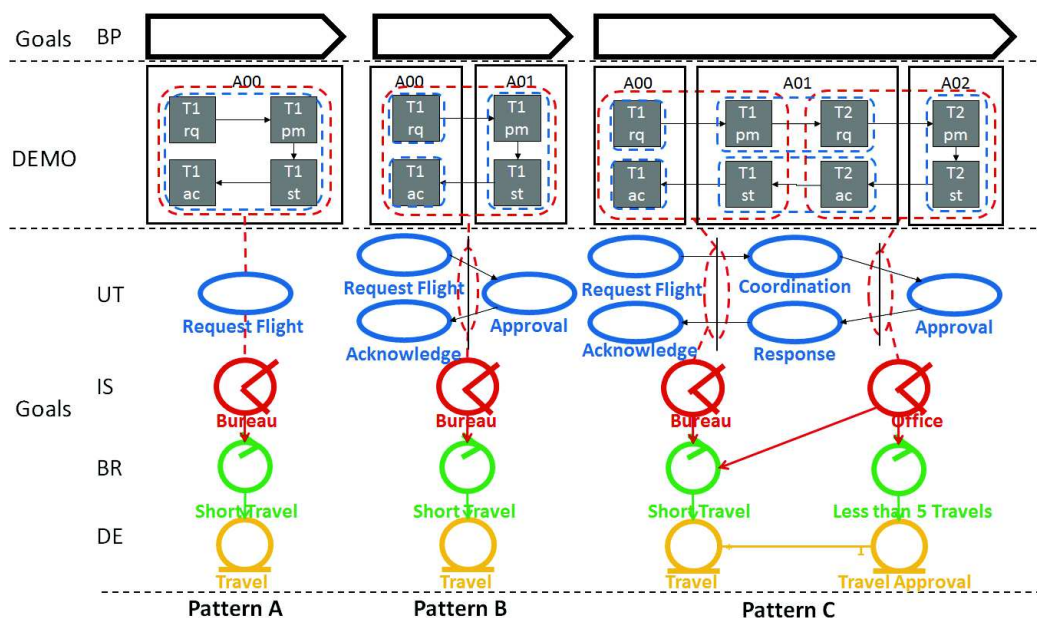


Fig. 1. Patterns A, B and C of component (BP, UT, IS, BR and DE) derivation from DEMO.

2 Analysis Phase

The elaboration of the Enterprise Structure is presented in Steps 1 to 5.

2.1 Step 1 – Business Process (BP) Identification

Goals definition of BP is compliant with the notion of Business Process provided by DEMO as a “set of interrelated or enclosed Transactions”. One Transaction is a

sequence of Coordination Acts {namely: request (rq), promise (pm), state (st) and accept (ac)} performed by two actors, or by a single actor directly in the system.

Figure 1 presents the BP derivation patterns based on the DEMO Process Structure Diagram (PSD). Pattern A includes a single Transaction (T1) performed by Actor A00, pattern B includes a single Transaction (T1) performed by two Actors (A00 and A01), and pattern C has two Transactions (T1 and T2) performed by three Actors (A00, A01 and A02). In all cases the relation between Goals and DEMO BPs is of one-to-one.

2.2 Step 2 – User Task (UT) Identification

Contrarily to DEMO, Goals considers that an Actor always carries on a only single task (a UT) and never two consecutive tasks or Coordination Acts (C-Acts). This aims Business Process clarification, user performance and software conception efficiency in order to deploy the necessary tools for the execution of the task by reducing articulatory distance and therefore, the user effort [15]. Hence, Goals considers any consecutive DEMO C-Acts {request (rq), promise (pm), state (st) and accept (ac)} as a single UT.

Figure 1 presents the derivation of UTs from the DEMO PSD. In pattern A a single UT is considered for the four consecutive C-Acts {rq, pm, st and ac}. In pattern B the consecutive C-Acts {pm and st} performed by Actor A01 are considered as a single UT (“Request Flight”), and in pattern C, Actor A01 is responsible for transposing the BP execution from Actor A00 to A02 and viceversa by carrying on the UTs “Coordination” and “Response”, which are merged from consecutive C-Acts, namely {T1 pm - T2 rq} and {T2 ac - T1 st} respectively.

2.3 Step 3 – Interaction Space (IS) Identification

One IS supports the interaction between two users in person or remotely while each one carries on his own UT. Even if many UTs are carried by many Actors, the UTs will still be different, and if two Actos carry on the same UT of the BP remotely, then they are performing cooperative work [16]. The derivation of ISs does not depend on DEMO models as this method does not consider the space where human activity occurs.

Figure 1 illustrates the derivation of the IS from the relation of UTs, as each IS (e.g. “Bureau” in Pattern B) supports the communication between any two or more Actors, the line that divides the swim-lanes of each Actor represents an IS. Given that DEMO only predicts the interaction between two Actors, when applied to DEMO models, this pattern of derivation will always result in a direct relation between one IS (e.g. ISs “Bureau” and “Office” for Transactions T1 and T2 in Pattern C) per Transaction.

2.4 Step 4 – Business Rule (BR) Identification

BRs represent regulations or requirements that should be elicited during the Analysis Phase in order to facilitate the understanding of the restrictions which the user is subject to when carrying on a User Task within a certain Interaction Space, and represent

restrictions which are applied to existing Data Entities. BRs are the grounding foundation of the Business Logic (given an MVC pattern), as they are the more specific programmed system responsibility regarding the structuring of this layer, the middle-ware of the system.

Figure 1 illustrates a situation in which both T1 and T2 define a BR each (“Short Travel” and “Less than 5 Travels”), which are also used by the Interaction Spaces of Transactions T1 and T2. BRs are constantly executed in order to ensure that a given restriction is ensured regarding the transfer of information between Interaction Spaces and Data Entities. BR should also be used by Interaction Spaces in order to restrict the introduction of invalid information by the user, therefore preventing usage mistakes.

2.5 Step 5 – Data Entity (DE) Identification

DEs are business concepts which are recognized within the enterprise domain by those who have knowledge about it (the enterprise). DEs are compliant with the concept of Class and relation of Classes used in UML [17]. And this definition is compatible with DEMO Object Classes (OC) which structures facts and Transactions. Goals derivation of DEs is carried out using the Object Fact Diagram (OFD) by establishing a direct relation between one DE per OC.

Figure 2 which presents the relation between OCs and Transactions in the OFD horizontal swim-lane. Transactions 1 and 2 use each a single DE, and these are related in a multiplicity of 1 to many (from “Travel” to “Travel Approval”). The resulting Enterprise Structure is presented above the DEs representation, and is composed by every identified component until this moment with no changes and is representative of the social interaction in terms of stable and essential norms, which is known as the organizational kernel [18].

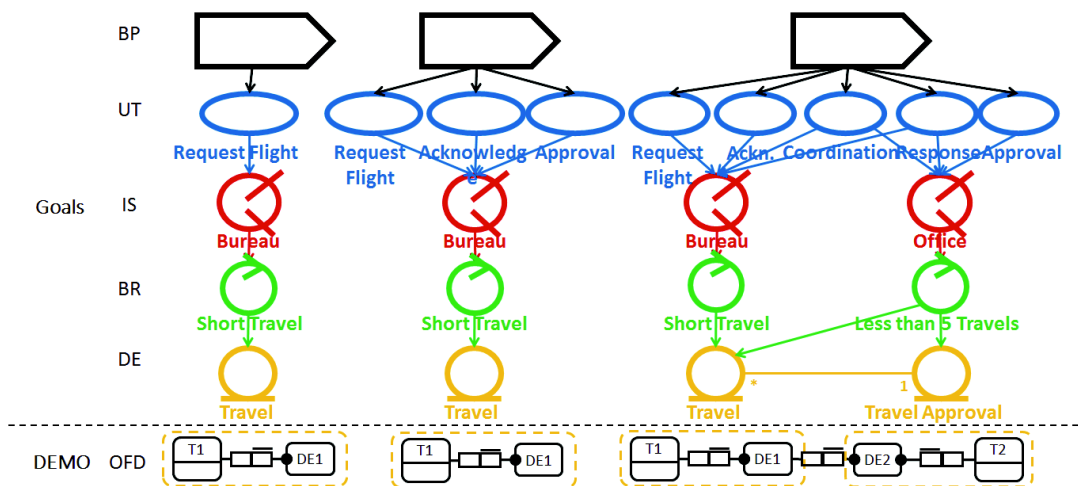


Fig. 2. Enterprise structure and derivation of DEs from OFD diagram.

3 Analysis Phase

The Design Phase elaborates the Software Architecture which is conceived in a top-down process that detailing the User Interaction (Step 6), the User Interface (Step 7), the Business Logic (Step 8) and the Database layer (Step 9), finishes with the composition of the Software Architecture (Step 10).

3.1 Step 6 – Task Model

The Task Model details User Tasks (UTs) in order to obtain information in order to carry on the User Interface design, which happens in Step 7. The Task Model follows the technique applied in the Wisdom methodology in order to specify the UT in terms of User Intentions (steps that the user takes to complete the task) and System Responsibilities (that provide the necessary information), following a traditional decomposition of an Essential Use Case (EUC) by means of the application of the Concur Task Trees (CTT) technique [19].

3.2 Step 7 – Interaction Modeling

The User Interface Design is carried out by means of the application of the Behavior Driven Development (BDD) method [11]. BDD is an agile software development method that produces pseudo-code as User Stories in order to specify a system feature (a UT) which is used within a certain scenario (an Aggregation Space).

User Stories specify a flow of user interactions that matches the User Intentions of the Task Model, specifying one Interaction Components per User Intention, and one Interaction Object per User Interaction, and related system behavior in terms of User Interface and Database System Responsibilities (SRs).

Figure 3 presents a User Story example for User Task “Request Flight” where three Interaction Components (A, B and C) and three SRs (the last SR is always a Database SR) are identified following the specification of four User Interactions. The User Interface Design specifies the Aggregation Space which is composed by the Interaction Components and the Interaction Objects (one to support each User Interaction).

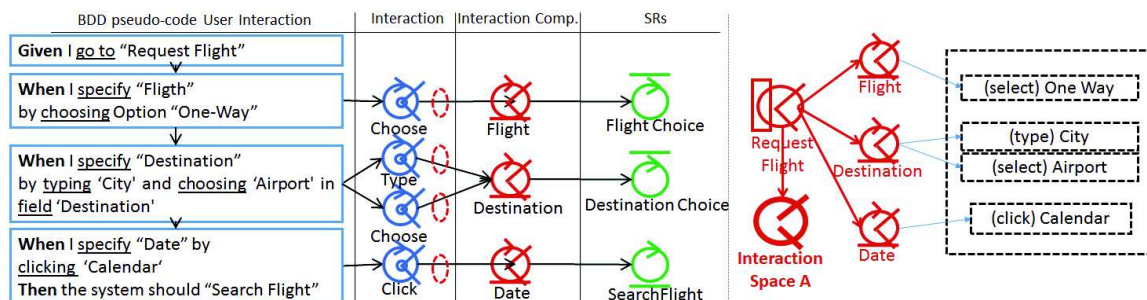


Fig. 3. Interaction model and user interface example.

3.3 Step 8 – Business Logic Structuring

The Business Logic Structuring is carried out by defining the relations that each System Responsibility (SR) to the existing to Data Entities (DE) based on the semantics and current state about identified business concepts. Given the current example and given Pattern A of derivation, we assume that DEs “Travel” and “Approval” are inherited from the Enterprise Structure. “Flight Choice” has been mapped to “Travel”, and it is assumed that the “Airport” Fields belongs to a new DE “Airport”. By means of the analysis of “SearchFlight”, we assume that it uses a new DE “Flight”.

3.4 Step 9 – Database Structuring

The Database Structuring is possible once all new DEs and Fields are already identified. The structuring od carried out according to the principles of elaboration of a Domain Model [17], in terms of Classes and Attributes which suffer simple transformation in order to structure the final Database [20]. According to our example, two new DEs have been identified (“Flight” and “Airport”), and for purposes of exemplification, we assume that DE “Travel” can only be related to a single record of those new DEs, and that DE “Flight” can is related to more than one “Airport” (usually two).

3.5 Step 10 – Software Architecture Composition

The composition of the Software Architecture is carried out by relating in a single diagram the every identified component by means of the execution of Steps 1 to 9. Figure 12 presents the Software Architecture, which relates all the identified components in a single Software Architecture, including the User Interface components associated to UT “Request Flight”, the elaborated Business Logic and Database components, including the components which are architectural inherited from the Enterprise Structure.

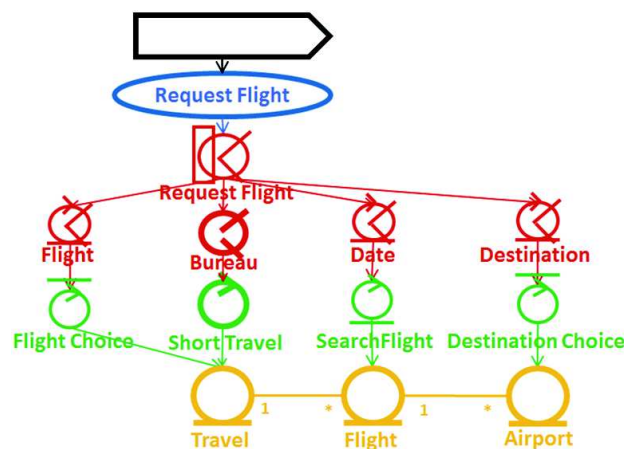


Fig. 3. Software Architecture example.

4 Related Work

Our approach can be compared to ArchiMate [21] and BPMN [22] in the perspective that it provides an Enterprise and Software Structuring language. It is different in the perspective that it applies a methodology to derive software implementation specifications. Regarding the specific User-Centered Design perspective, the closest solutions are methods settle for user interface conception based on user task and domain models, such as Sukaviriya's [23] and Sousa's [24]. Our approach is different as it complementarily conceives the Business Logic layer based on enterprise business rules and coordination structures that operate the user interface and domain processing execution. Considering the enterprise-driven development, it is different from the DEMO-based GSDP [25] as it specifies a structured user interface.

5 Conclusions

Our approach inherently aims at facilitating requirements elicitation, focuses on user needs, and simplifies traceability between business requirements and software implementation, which match project management needs and user involvement in the SDP. The Goals Approach strategy, which is based on BPI, fits most successfully sized projects. Based on Standish Group statistical reports, projects under 1 M\$ (one million dollars) cost are believed to be up to 10 times more successful than 10 M\$ projects [4]. It suits Small and Medium Enterprises (SME) in-house development needs of agility concerning the achievement of tangible results in limited amounts of time [7].

References

1. The Standish Group: Chaos Report 2014 (2014)
2. Valente, P., Aveiro, D., Nunes, N.: Improving software design decisions towards enhanced return of investment. In: Proceedings ICEIS 2015, pp. 388–394 (2015)
3. Morgenshtern, O., Raz, T., Dvir, D.: Factors affecting duration and effort estimation errors in software development projects. *IST* **49**, 827–837 (2007)
4. The Standish Group: Chaos Report 2013 (2013)
5. Version One. The 10th Annual State of Agile Report (2016)
6. Dietz, J.: Enterprise Ontology - Theory and Methodology. Springer, Berlin (2006). ISBN 978-3540331490
7. Gerogiannis, V., Kakarontzas, G., Anthopoulos, L., Bibi, S., Stamelos, I.: The SPRINT-SMEs. In: Proceedings of ARCHIMEDES III (2013)
8. Nunes, N.: Object modeling for user-centered development and user interface design: the wisdom approach. Ph.D. thesis, Universidade da Madeira (2001)
9. Valente, P.: Goals Software Construction Process: Goal-Oriented Software Development. VDM Verlag Dr. Müller, Germany (2009). ISBN 978-3639212426
10. Constantine, L.: Human Activity Modeling - Toward a Pragmatic Integration of Activity Theory and Usage-Centered Design. Springer, Berlin (2009)

11. Chelimsky, D., Astels, D., Helmkamp, B., North, D., Dennis, Z., Hellesoy, A.: *The Rspec Book* (2010). ISBN: 1934356379
12. Lodhi, A., Köppen, V., Saake, G.: Business process improvement framework and representational support. In: *Proceedings of the 3rd International Conference on Intelligent IHCI* (2011)
13. Zukowski, J.: The model-view-controller architecture. In: *John Zukowski's Definitive Guide to Swing for Java 2* (1999). ISBN: 978-1430252511
14. Costa, D., Nóbrega, L., Jardim Nunes, N.: An MDA approach for generating web interfaces with UML ConcurTaskTrees and canonical abstract prototypes. In: Coninx, K., Luyten, K., Schneider, K.A. (eds.) *TAMODIA 2006*. LNCS, vol. 4385, pp. 137–152. Springer, Heidelberg (2007)
15. Winckler, M., Cava, R., Barboni, E., Palanque, P., Freitas, C.: Usability aspects of the inside-in approach for ancillary search tasks on the web. In: Abascal, J., Barbosa, S., Fetter, M., Gross, T., Palanque, P., Winckler, M. (eds.) *INTERACT 2015*. LNCS, vol. 9299, pp. 211–230. Springer, Heidelberg (2015). doi:[10.1007/978-3-319-22668-2_18](https://doi.org/10.1007/978-3-319-22668-2_18)
16. Grudin, J.: Computer-supported cooperative work: history and focus. *Computer* **27**, 19–26 (1994)
17. Booch, G., Jacobson, I., Rumbaugh, J.: *The Unified Modeling Language Users Guide*. Addison-Wesley, Wokingham (1998)
18. Stamper, R.: On developing organisational semiotics as an empirical science: the need for scientific method and rigorous debate. In: *Proceedings of 14th ICISO*, pp. 1–13 (2013)
19. Paternò, F.: *Model-Based Design and Evaluation of Interactive Applications*. Springer, London (1999)
20. Awang, M., Labadu, N.: Transforming object oriented data model to relational data model. *New Comput. Archit. Appl.* **2**(3), 402–409 (2012)
21. Archimate Foundation: *Archimate Made Practical* (2008)
22. Völzer, H.: An overview of BPMN 2.0 and its potential use. In: Mendling, J., Weidlich, M., Weske, M. (eds.) *BPMN 2010*. LNBIP, vol. 67, pp. 14–15. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-16298-5_3](https://doi.org/10.1007/978-3-642-16298-5_3)
23. Sukaviriya, N., Sinha, V., Ramachandra, T., Mani, S., Stolze, M.: User-centered design and business process modeling: cross road in rapid prototyping tools. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) *INTERACT 2007*. LNCS, vol. 4662, pp. 165–178. Springer, Heidelberg (2007)
24. Sousa, K., Mendonça, H., Vanderdonckt, J., Rogier, E., Vandermeulen, J.: User interface derivation from business processes: a model-driven approach for organizational engineering. In: *Proceedings of 2008 ACM SAC*, pp. 553–560 (2008)
25. Kervel, S., Dietz, J., Hintzen, J., Meeuwen, T., Zijlstra, B.: Enterprise ontology driven software engineering. In: *Proceedings of ICsoft 2012* (2012)