Comparing DEMO with i_Star
In Identifying Software Functional Requirement

Tarek Fatyani, Junich Iijima and Jaehyun Park
Graduate School of Decision Science and Technology, Tokyo Institute of Technology, Tokyo, Japan
{fatyani.t.aa, iijima.j.aa, park.j.ai}@m.titech.ac.jp

Keywords: Demo model, Enterprise engineering, Requirement engineering, i* (i-Star) framework, Goal modelling.

Abstract: Information systems development (ISD) has encountered a variety of challenges in terms of identifying the requirements among multiple stakeholders. This is due to the complexity of the related information. Therefore, an abstract model of the enterprise is needed to focus on people and their needs before developing any information system. To respond to this need, new modeling methodologies that focus on modeling the enterprise as a social system have got a wide acceptance. DEMO and i* are an example of these modeling methodologies. They focus on modeling the people and the interaction between them. Although DEMO is a based on strong theories, it is not used much as i* in requirement engineering. Therefore, this research compares these two modeling methodology in identifying the functional requirements for developing information system. The comparison is to highlight the strong and the weak part of both modelings. Moreover, this research draws guidelines for improving both methodologies in modeling enterprise as a prior step in developing information system. As a result, the concept of modeling the interaction between DEMO and i* is different. DEMO is more formal inmodeling the interaction rather than i*. Moreover, DEMO models both the structure and the behavior through its different diagrams. But i* does not capture the behavior. In contrast, i* allows to model the non-functional requirements, too. Sometimes it is useful to analysis them during the first stages of requirements analysis.

1 INTRODUCTION

Current information systems are getting more complex in terms of the number of the stakeholders who benefits from these systems as well as in terms of the related information to be in the system. Therefore, information systems development (ISD) encountered a variety of challenges in terms of identifying the requirements among multiple stakeholders. How can one differentiate between what the users want and what they really need. One of the common problems in requirement analysis is requirements conflicts. Therefore analyzing the requirements is crucial for software development (Mazón, J.N., Pardillo, Juan, 2007). Scoping and requirements engineering are the most important challenges that SMEs faces during information systems development (Silva, Neto, O'Leary, Almeida, Meira, 2014). Therefore, before the requirement analysis stage, an abstract model of the enterprise is needed. This model must describe the essence of the enterprise. It should describe the structure of the organization and its interaction with its environment (Tuunanen, Rossi, Saarinen,Mathiassen, 2007). Failure to do so may lead to a requirement uncertainty (Michalik, Keutel, Mellis, 2014).

In the last two decades, practitioners and researchers seeked an alternative for modeling the enterprise as a social system. This means that enterprise consists of individuals who interact with each other to deliver a particular product or service to the environment.Therefore, new modeling methodologies were developed to model the enterprise as a prior model to any implementation.It has proved that such a modeling with analysis provides benefits at every stage of the requirements engineering process (Jose, Jesús, Juan, 2007). DEMO and i* are good examples of these methodologies. DEMO provides a formal model of the enterprise, including the structure and the behavior.Although DEMO is based on strong theories, it is yet to be used in many real-world scenarios in developing information systems (Kervel, Hintzen, Meeuwen, Vermolen, Zijlstra, 2011). On the other hand, i* is widegly accepted modeling methodolgoy in many

Fatyani T., Iijima J. and Park J.
Comparing DEMO with i_star – InIdentifyingSoftwareFunctionalRequirement.
DOI : 10.5220/0005886401490155
In Proceedings of the Fifth International Symposium on Business Modeling and Software Design (BMSD 2015), pages 149 – 155
ISBN : 978 – 989 – 758 – 111 – 3

fields (Yu, Giorgini, Maiden,Mylopoulos, 2011). In particular it used in modeling the goals of the information system before developoing it. There are many frameworkds for develping the requirements based on i*. It is similar to DEMO in providing a better understanding of the decision-making process and the rationales behind it by providing an abstract model of the enterprise (Átila, Monique, Emanuel, Josias, Fernanda, Jaelson, 2011).

This research aims to understand the difference in popularity of the two methodologies i* and DEMO by comparing them in one real case study. By this comparison, we can highlight the pros and cons of using DEMO. It also helps in developing a framework for developing information system based on DEMO similar to the frameworks that i* has.

As a result of this research, it is clear that DEMO is implemntation independent methodology. But i* is implementation dependent emthodology. This means that DEMO model does not change according to the implementation method. It is up to the designer to select the implementation that fits the enterprise needs. And DEMO model will not be changed before and after the implementation unlike the i* model. However, i* can capture not only the social aspects of the enterprise, but also the rational aspects, too. The rational dependency model of i* can model the processes as they are in the implementation. This is useful for developing the information systems.

The rest of the paper is as follows. First, literature review provides an explanation about i* and DEMO with their recent research in the field of requirements engineering. Second, a real world case study is introduced, then modeled by both i* and DEMO. Third is the conclusion with the discussion about the similarities and the differences between i* and DEMO.

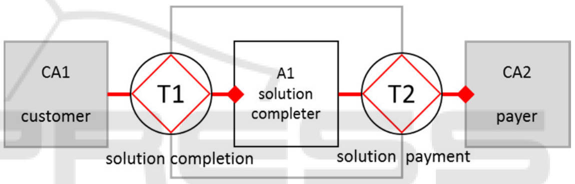
2 LITERATURE REVIEW

In this section, the main concepts of DEMO and i* are explained. a running example will be used to explain the way of modeling of DEMO and i*. Where, customers request the enterprise a particular IT solution. And they pay the fees for this service.

2.1 The DEMO Model

DEMO, which stands for "Design and Engineering Methodology for Organizations", is based on PSI (Performance in Social Interaction)-theory.In this theory, an enterprise (organization) is considered as an interaction of social individual subjects. DEMO helps in 'discovering' an enterprise's ontological model, basically by re-engineering from its implementation.The main elements of DEMO models are actor roles and transactions. Any transaction within an enterprise is carried out by an interaction of two actor roles. The first actor role is responsible for initiating the transaction while the other actor role is responsible for executing the transaction (Dietz, 2006).

DEMO consists of four models. The construction model (CM) specifies the structure of the enterprise in relation to its environment, including the transactions, actor roles, information banks and links between them. The process model (PM) specifies the details of the transactions in the CM. Though the CM does not specify a sequence in which the transactions are executed, the PM does while indirectly indicating the timeline. The fact model (FM) specifies the object classes, which consist of a fact kinds and transaction result kinds. The action model (AM) formulates the business rules for executing each process step in the PM.



transaction kind	product kind
T1 solution completion T2 solution payment	P1 Solution is completed P2 the fee of Solution is paid

Figure 1: CM model of simple case.

Figure 1 shows CM model of a simple case. CM consists of OCD (organization structure diagram) and TPT (transaction product table). Customer requests a solution from IT Department store, which is represented, by the actor role solution completer. This actor role is the executor of T1. Therefore, a small diamond appears at the end of the link to T1. The solution completer then asks for the payment by initiating the transaction T2. The grey rectangle represents the scope of interest (IT Department store). "A" stands for actor role. White actors are elementary actors. And grey actors are composite actor roles. TPT shows the product of each transaction after completion.

In DEMO, the ontology, infology and datalogy levels are clearly differentiated. In the ontology level, actors (human beings) initiate and execute transactions that result in original facts, for example, purchasing. At the infology level, transactions only manipulate information from one shape to another, for example, calculating salary. At the third level, datalogy, transactions store and retrieve data without any manipulation. DEMO provides a high level of abstraction of the enterprise.

DEMO has already proved its powerfulness in capturing a high abstract conceptual model of the enterprise. DEMO models can be used in process re-engineering as well in enterprise engineering. However, using DEMO models in a stage prior to requirement engineering for developing information system is still in progress. A few frameworks are developed for developing enterprise information system based on DEMO. Nevertheless, very few real world case studies applied the frameworks. Therefore, more real world case studies are needed to enhance and justify those frameworks. For example, DEMO processors that compile and execute the DEMO models have been developed (Kervel, Dietz,Hintzen, Meeuwen, Zijlstra, 2012).

On the other hand,i* model is a previously established framework in requirement engineering for developing information systems (Pandey, Suman, Ramani, 2010). By comparing i* with DEMO, we can thus formulate a framework for developing information system based on DEMO models.

2.2 i* (i-Star) Framework

The i* (i-star) framework is one of the most widely adopted modeling approaches by several communities (e.g., requirements engineering, business process reengineering, organizational impacts analysis and software process modeling). It is a goal- and agent-oriented modeling and reasoning framework that defines models that describe the systems with the environments in terms of intentional dependencies among strategic actors (Pandey, Suman, Ramani, 2010). It is used for comparing many different scenarios for the same system by changing the dependencies between the agents (Liu, Yang, Wang, Ye, Liu, Yang, Liu, 2014). There are two different models: (1) the strategic dependency (SD) describes information about dependencies and (2) the strategic rationale (SR) defines the actor details. The SR model complements the information provided by the SD model by exploiting internal details of the strategic actors to describe how the dependencies are accomplished.

Figure 2 shows SD of the same simple case that represented in CM model in DEMO. Actors are represented by circles. Customer asks the solution provider an IT solution. Here, the solution represented by oval to show that it is goal type. The solution provider asks the payment from the customer. Payment is represented by a rectangle because it is considered a resource type.

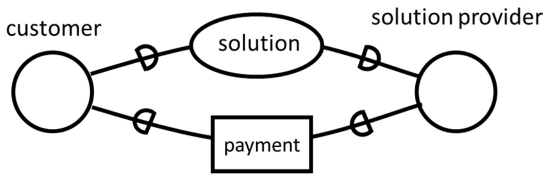


Figure 2: SD of a simple case.

In i*, there are four types of dependencies that are characterized according to the dependum. The dependum can be a soft-goal, a goal, a task or a resource. Soft-goals are associated with non-functional requirements (NFRs), while goals, tasks, and resources are associated with system functionalities (Castro, Lucena, Silva, Alencar, Santos, Pimentel, 2012).

3 CASE STUDY

The following passage describes a case study modeled by both DEMO and i*. Those models will be used later for specifying the requirements to develop information system. This section consists of four parts. First is a textual description of the case study. Second is explaining the objectives of developing the information system. Third is the DEMO model with its explanation. Fourth is i* model with its explanation.

3.1 Background of the Selected Case

SMA offers its customers IT solutions by developing software based on the customer's needs or by providing consultation. SMA is a project-based company. Every project belongs to one client who may have more than one project with the company. The employees do not form a structure based on the organization chart, rather they are flexible based on the projects they have. This flexibility allows the company to respond quickly to the changes in the market.

In every project there is one project manager leading a team of developers. The project manager is responsible for planning the project, as well following it to completion.

Based on the project, different person from project manager would be responsible for delivering the product to the customer. This person may also be responsible for taking care of the payment from the

customer. Otherwise, the project manager does the delivery and receives the payment from the customer. When a new project arrives, the project manager begins by planning the project.
As a result of the planning, a list of tasks with their schedules is made. After breaking down the project into tasks, the project manager assigns the tasks to the developers. During the execution of the project, new tasks may pop up. Therefore, every developer may assign a new task to himself/herself or assign it to the other developers. All the tasks must be recorded in the information system to be developed. This is very important to follow up the completion of each task.

The project manager controls the completion of the tasks every week. The project manager then looks at the completed tasks and the remained tasks. He/she reassigns the tasks from developers with work overload, to those who have less work. This provides a work balance for every developer, to allow efficient project completion. At the same time, the manager may control the execution of the tasks by prioritizing them according to their importance.
Employees receive their salaries based on their work time. Therefore, they record the time for completing every task they do. In addition, at the end of the month, the accountant calculates the work time for each employee. For each employee, there is a specific hourly salary rate. Based on this rate, the actual payment is calculated. The salary is the sum of work time multiplied by the salary rate plus the reward.

Employees may ask for bonuses or other rewards. This is done after evaluating their performance. The project manager analyzes the performance of all employees based on their task completion rate. Moreover, based on the performance analysis, the salary rate may increase or a reward for a particular project may be given.

Employees are free to choose their time to work, i.e., day or night, as long as the projects are proceeding as scheduled. This flexibility gives them responsibility for their time.

To keep the level of the skills in the company up to date, SMA frequently hires new highly qualified developers.

3.2 Objectives of IS to be Developed

The information system to be developed has three main objectives.

The first objective is to follow up the execution of all the projects. During the execution of the project, the project manager needs to know the statutes of the tasks and the workload for each employee. This helps to follow up on the project to meet timeline, quality and cost limits.

The second objective is automation. Because the salary of each employee is based on the tasks that are executed by the employee, there are many calculations needed. To reduce the cost of these calculations, they should be automated.

The third objective is the performance analysis. To analyze the performance of each employee, a record of his/her achievements should be archived. Because each task is associated with an execution time, the productivity of the employee may be estimated.

3.3 The DEMO Model

Based on the description in the previous two paragraphs, the DEMO model of SMA can be constructed as follows. Because of pages limitations, only the organization construction diagram (OCD) of the construction model (CM) will be detailed. The unit of business service of SMA is the provision of an IT solution to the customer. Therefore, the first transaction to be identified is the (T1) project completion. The customer (CA1) initiates this transaction by requesting an SMA employee to provide an IT solution. This employee will be called the project completer (A1). A1 initiates three transactions: (T2) project fee payment, project planning (T3) and task completion (T4). It must be said that T4 cannot be requested before the plan of the project is done. Since the project manager controls the execution of the plan, then he or she is taking the role of task manager (A5). This actor initiates and executes periodically (every week) the transaction task management (T5). The execution of this transaction leads to change the project plan. Therefore, plan revision transaction is needed (T6). To model the salary and the reward that are mentioned in the description, salary payment control (T7) and reward management (T10) transactions are needed. To execute T7 we need to calculate the salary. Because there is no original fact in T8 (only calcluation), then it is infological transaction (green). To execute T10, we need two sub transactions, reward decision making (T9) and employee evaluation (T11). Based on the previous paragraph that describes the objectives of the information system to be developed, the scope of the information system is shown by a green rectangle. The customer

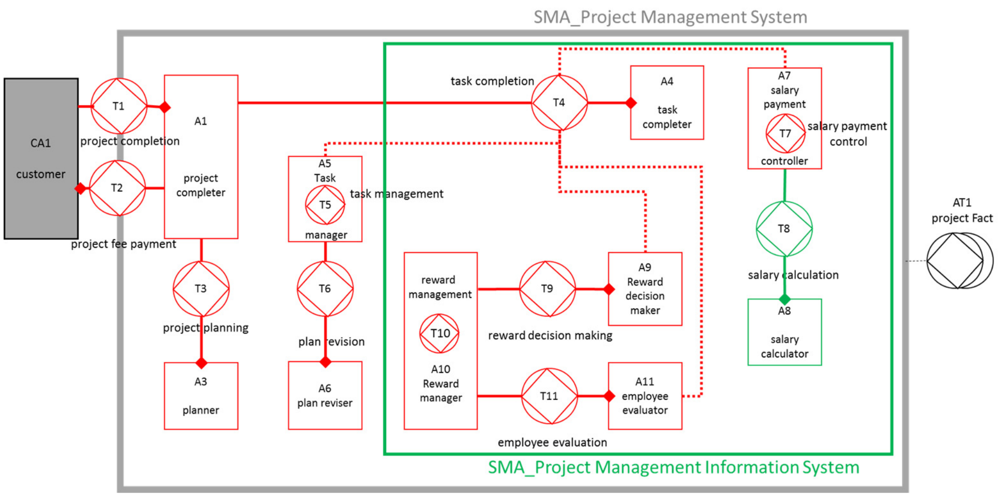


Figure 3: Actor transaction diagram of SMA.

has no relationships with the information system. This is because SMA prefers to always communicate with the customer directly to form good human relationships. The OCD of the CM is shown in Figure3.

3.4 i* Model

Because the objective of the model is to develop an information system, then only the actors who are involved in the system will be modeled. In i*, unlike in DEMO, we starts by modeling the actors. There are three actors: manager, employee and accountant. The manager has a dependency relationship with the employee by asking him/her to achieve a particular task. The dependency is of the task type because it is a task. The employee has two dependency relationships: salary and reward. Because they refer to the money to be given from the accountant to the employees, both dependencies are of the resource type. To give the reward, a performance analysis is needed. Therefore, the accountant depends on the manager to do the performance analysis for the employee before giving the reward. This is a resource type dependency. The strategic dependency (SD) model is shown in Figure 4.

For each actor in the SD, there is a rational dependency (RD) model. In this research, only one RD will be modeled. In Figure5, the RD for the manager is shown. The RD shows the internal tasks that the actor performs to respond to the external dependencies of the other actors. In SMA, the manager controls the tasks for each employee. The task can be decomposed into two tasks: assigning a task and releasing a task. These two tasks influence the soft-goal balanced load. At the same time, controlling the tasks is required for evaluating the achievement of an employee. The result of the evaluation is the performance analysis, which is delivered to the accountant.

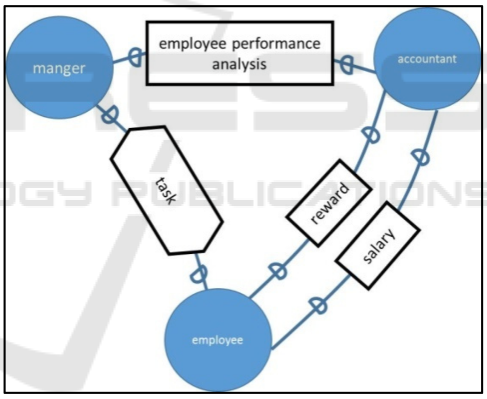


Figure 4: Strategic dependency model of SMA.

4 DISCUSSION AND CONCLUSION

4.1 Discussion

First, Both DEMO and i*are social modeling methodologies. In their models, enterprise consists of actors (actor role in DEMO and agent in i*) that interact with each other through relationships (transaction in DEMO and dependency in i*). However, the concept of actors and relationships are different. In i*, humans are modeled by an actor with concrete names, for example, manager, employee and

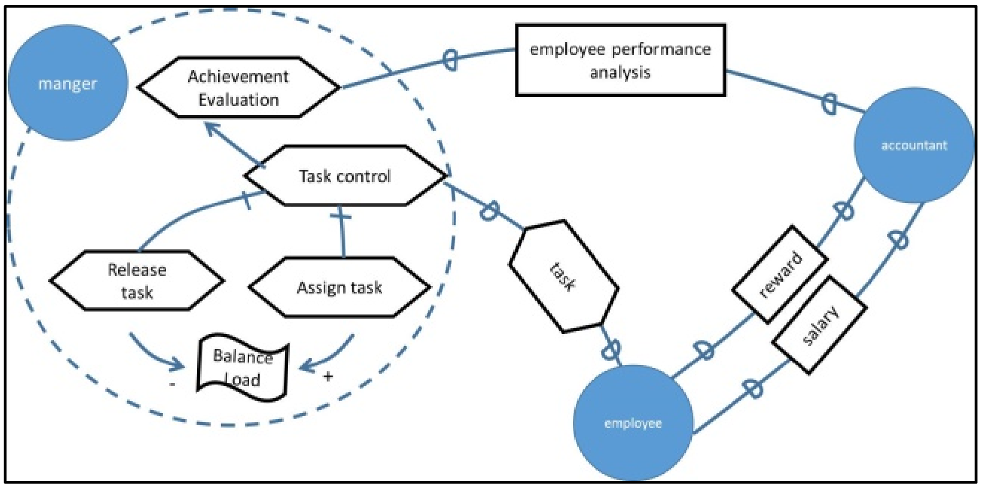


Figure 5: Rational dependency model of SMA.

accountant. However, in DEMO, it is more abstract such as planner and task controller. After presenting these two models to the stakeholders, it was more easy for them to understand i* model than DEMO. This is because i* uses more concrete roles that are familiar to the stakeholders. The following table matches the actor role in DEMO with the actor in i*.

Table1: Actors in DEMO and i*.

DEMO actor role	i* actor
task manager, plan reviser, reward manager, and employee evaluator	Manager
Task completer	Employee
Salary payment controller, salary calculator, and reward decision maker	Accountant

From the Table 1, we can see that every actor in i* can be broken down into actor roles in DEMO. Therefore, a composite actor role can be used in DEMO to make it easier to be understood. Every composite actor role can be later decomposed into its elementary actor roles. This facilitates discussing the model with the stockholders.

Second, there is a difference between a transaction in DEMO and a dependency in i*. In DEMO, transactions are divided into ontological, infological and datalogicalcategories according to the abstraction level. The differences between them are clear by definition. Whereas in i*, the dependency is divided into goal, resource, task and soft-goal. However, the difference between goal, resource and task is not clear. This looseness may not be important in some situations. However, in others situations, such as model-driven development, it is very important (Lidia, Xavier, Jordi, 2014).For example, in our model, both reward transaction and salary transaction are modeled in i* as a resource. However, task completion transaction in DEMO is modeled in i* as a task. Therefore, no automatic transformation between transaction in DEMO into i* could be done.

Third, i* is capable of capturing the rational aspect of the system by RD model. Agent in i* could be decomposed into rational elements. This facilitates the implementation by automating them. However, DEMO is considering only the social part of the system. This makes it difficultfor implementing the system in later stages.

Fourth, i* models soft-goals. This is useful for modeling the nonfunctional requirements in the early stages. Service quality and service speed are examples of nonfunctional requirements. But DEMO considers only the functional requirements.

Fifth, DEMO has four perspective models (construction, process, fact and action) that captures a holistic view of the enterprise. This is very important in developing any information system (Figueiredo, Souza, Pereira, Prikladnicki, Audy, 2014). However, i* does not have equivalent to fact and action models of DEMO.

4.2 Conclusion

This research compares between two modelling methodologies named DEMO and i* in modelling enterprise as a prior stage of requirements analysis. By the comparison between DEMO and i*, both of them are social modeling methodologies. They focus on human and human interaction in their modeling. DEMO is implementation independent. Therefore, the DEMO model does not change before or after implementing any IT solutions. However, i* is implementation dependent methodology. DEMO provide more formal and rigour model of the enterprise. That makes it a good potential modelling methodology to understand the enterprise before implementing any IT solution. On the other hand, i* allows us to capture both the rational and the social aspect of the enterprise. The rational aspect is important in developing any information system. Therefore, DEMO should be extended to capture the non-social aspect, too.

There are some frameworks for developing information system based on i*. Since we showed the strength of DEMO, the next step is to develop such a framework like the i* has. Another point to be considered in the future is extending DEMO to capture the rational aspect of the enterprise like i*. This is important for developing information systems.

REFERENCES

Mazón, J.N., Pardillo, J.,Juan, J.T., 2007. A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses. In: *ER 2007 Workshops CMLSA*, FP-UML, ONISW, QoIS, RIGiM,SeCoGIS, New Zealand, pp. 133--142.

Silva, I.F,Neto, P.A., O'Leary, P., Almeida, E.S., Meira, S.R.L., 2014: Software product line scoping and requirements engineering in a small and medium-sized enterprise: An industrial case study. *J. Journal of Systems and Software*. 88, 189—206

Michalik, B., Keutel, M., Mellis, W., 2014: Coping with Requirements Uncertainty -- A Case Study of an Enterprise-Wide Record Management System. In: *47th Hawaii International Conference on System Sciences (HICSS)*, pp. 4024--4033. IEEE Press, Waikoloa, HI

Tuunanen, T., Rossi, M., Saarinen, T.,Mathiassen, L., 2007: A Contingency Model for Requirements Development. *J. Journal of the Association for Information Systems*. 8, 11, 569--597

Yu, E.,Giorgini,P., Maiden, N.,Mylopoulos, J., 2011: Social Modeling for Requirements Engineering. *Massachusetts Institute of Technology*, United States

Átila, M., Monique, S., Emanuel, S., Josias, P., Fernanda, A., Jaelson, C., 2011: iStarTool: Modeling requirements using the i* framework. In: CEUR *Proceedings of the 5th International i* Workshop* (iStar 2011), pp. 163--165.

Kervel, S.V., Hintzen, J., Meeuwen, T.V., Vermolen, J.,Zijlstra, B., 2011: A Professional Case Management System in Production, Modeled and Implemented using DEMO. In: *30th International Conference, ER* 2011, pp. 62--77. Brussels

Dietz, J.L.G., 2006: Enterprise Ontology: Theory and Methodology. *Springer-Verlag* Berlin Heidelberg, New York

Kervel, S.V.,Dietz, J.L.G.,Hintzen, J.,Meeuwen, T.V.,Zijlstra, B., 2012: Ontology driven enterprise information systems engineering. In: *7th International Conference on Software Paradigm Trends*, pp. 205--210. Rome

Pandey, D., Suman, U., Ramani, A.K., 2010: An Effective Requirement Engineering Process Model for Software Development and Requirements Management. In: *Second International Conference on Advances in Recent Technologies in Communication and Computing*, pp. 287--291. IEEE Press, Kottayam

Liu, L., Yang, C.,Wang, J.M., Ye, X.J., Liu, Y.P., Yang, H.J., Liu, X.D., 2014: Requirements model driven adaption and evolution of Internetware. *J. Science China Information Sciences*. 57, 6, 1--19

Castro, J., Lucena, M.,Silva, C.,Alencar, F.,Santos, E., Pimentel, J., 2012: Changing attitudes towards the generation of architectural models. *J. Journal of Systems and Software.* 85,3, 463—479

Figueiredo, M.C., Souza,C.R.B., Pereira,M.Z., Prikladnicki,R., Audy,J.L.N., 2014: Knowledge transfer, translation and transformation in the work of information technology architects. *J. Information and Software Technology*. 56, 10, 1233—1252