Facilitating Business Improvement by Information Systems using Model Transformation and Metrics

Haruhiko Kaiya[1,2], Shunsuke Morita[1], Kenji Kaijiri[1], Shinpei Hayashi[3] and Motoshi Saeki[3,2]

[1] Dept. of Computer Science, Shinshu University, Nagano 380-8553, Japan
kaiya@shinshu-u.ac.jp
12KA110E@shinshu-u.ac.jp, kaijiri@cs.shinshu-u.ac.jp
[2] National Institute of Informatics (NII), Tokyo 101-8430, Japan
[3] Dept. of Computer Science, Tokyo Institute of Technology, Tokyo 152-8552, Japan
saeki@se.cs.titech.ac.jp, hayashi@se.cs.titech.ac.jp

Abstract. We propose a method to explore how to improve business by introducing information systems. We use a meta-modeling technique to specify the business itself and its metrics. The metrics are defined based on the structural information of the business model, so that they can help us to identify whether the business is good or not with respect to several different aspects. We also use a model transformation technique to specify an idea of the business improvement. The metrics help us to predict whether the improvement idea makes the business better or not. We use strategic dependency (SD) models in i* to specify the business, and attributed graph grammar (AGG) for the model transformation.

Keywords: Requirements Analysis, Strategic Dependency Model, Model Transformation, Metrics

1 Introduction

To develop an information system supporting business, we typically perform the following steps; 1) analyzing current business and construct its model, so called *as-is* model, 2) identifying the problems that lurk in the as-is model, 3) evolving the as-is model to the *to-be* model that can solve the identified problems. In these steps, how to evolve the as-is to the to-be is one of crucial topics because most intellectual activities of human analysts are necessary to create the solutions of the identified problems. Although many techniques and tools are available for supporting these steps, little supporting techniques in the evolution of to-be models exist. For example, idea generation methods such as Brain Storming are considered as a useful technique. Although they can help human analysts to create the idea as solution of the problems, they are too general and weak to support the evolution step more effectively. Activity based cost (ABC) method [1] evaluates as-is activities only from aspects of cost and time spent in executing them, while Balanced scorecard [6] requires well skilled and experienced analysts to set up evaluation criteria. Rather, best practices of the past evolutions allow the ordinary analysts to create the to-be of higher quality with their less efforts and a technique to accumulate and utilized the best practices is necessary. The purpose of this paper is to explore the technique to formalize reusable best practices of how to create to-be models.

Many techniques and tools are related to modeling languages such as BPMN, Workflow Languages, the usages and the extension of UML diagrams, etc. Goal-oriented Requirements Analysis (GORA) can be applied to describe an as-is model and a to-be model of the business, and is useful to clarify its business goals. In [14], the authors developed a language called i* which contains GORA, and used it to represent as-is models and to-be in the organizational context. Almost all of these modeling languages are essentially graphs with types and attributes. Thus, the evolution from an as-is model to a to-be model can be considered as graph transformation and be formalized with graph grammar. This paper proposes a technique to specify the evolution with Attribute Graph Grammar (AGG) [12]. To specify types and attributes on graphs, we use a meta-modeling technique. In addition, we should define metrics to detect the problems in an as-is and the metrics can be defined on the meta-model. Our technique is for formalize best practices in creating to-be models with graph grammar and metrics definitions and accumulating them as reusable assets.

The rest of the paper is organized as follows. Section 2 is for preliminaries and we explain the existing techniques that we use in this paper, i.e. an extended i* and a graph re-writing system based on AGG. We use i* diagrams to represent as-is models and to-be ones as examples. In section 3, we illustrate how to define metrics and graph transformation as the evolution practices from an as-is model into a to-be model. Metrics play an important role because they are used to select applicable and suitable transformation rules and to clarify which aspects on a to-be model can be improved. One of the significant reasons why an information system is developed is to reduce human's responsibilities and efforts in achieving business goals by automating them. A Strategic Dependency (SD) model of i* can represents responsibilities of the stakeholders to goals, and this is a reason why we focus on SD diagram of i* in this paper. However, our technique is not limited to SD diagrams and we can apply other diagrams by defining metrics and graph grammar on its certain meta-model. Section 4 is for listing up related works.

2 Preliminaries

In this section, we briefly introduce i* strategic dependency (SD) modeling and attributed graph grammar (AGG) because our research in this paper uses these two techniques.

2.1 i* SD modeling

Before specifying an information system introduced in business, we have to analyze what kinds of goals human wants the system to achieve instead of human. For example, a human secretary wants a meeting scheduling system to summarize schedules of all staffs because he does not have to do it with the system. A modeling language i* strategic dependency (SD) model [14] is useful for us to analyze it because a dependency about the ownership and the responsibility for each goal is clearly represented. In i*, an as-is model is used for specifying current dependencies in business, and a to-be model is used for specifying expected dependencies in the business normally with information

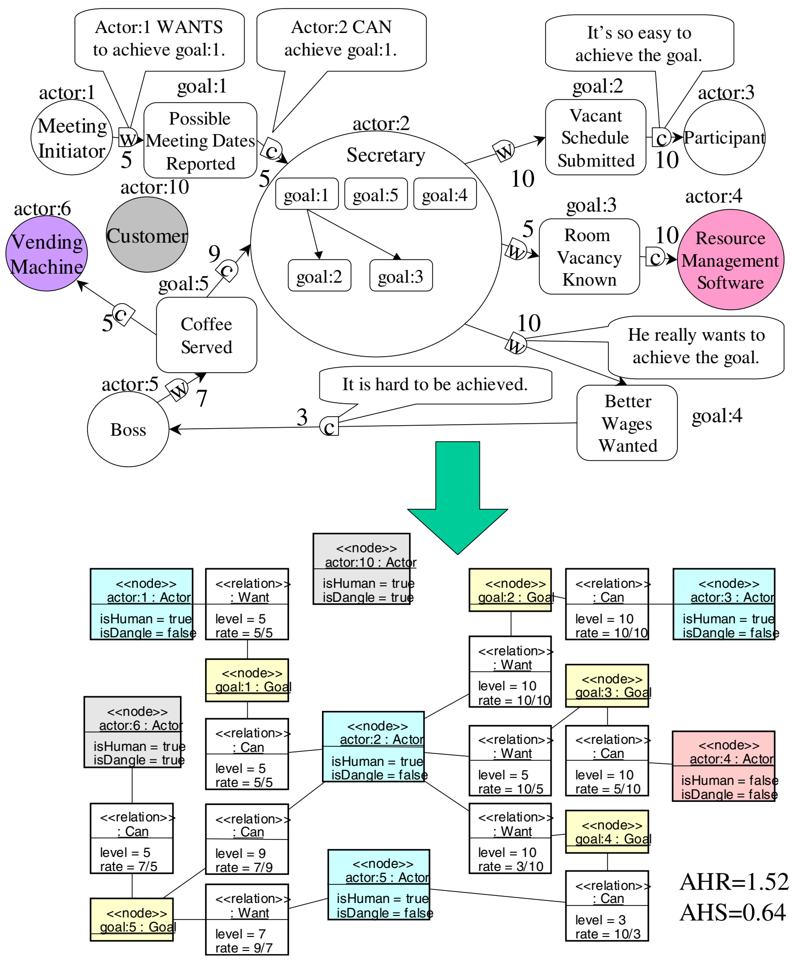


Fig. 1. An Example of an extended i* Model (top) and its Internal Representation (bottom). The internal representation is based on the meta-model in Figure 2.

systems. The to-be model should be thus better than the as-is model with respect to human actors in the to-be model.

To enable the model transformation in a SD model effectively, we extend the syntax of the SD model, and the extended models partially violate syntax of the original SD model [13]. The extended syntax will be explained in detail in the next section. Here we explain how and why we extend the SD model by using an example in Figure 1 in addition to the original syntax of the SD model.

A SD model consists of several pairs of an actor (called a *depender*), a goal and another actor (called a *dependee*). In each pair, these two issues are modeled; an actor wants to achieve a goal, and another actor can achieve the goal. Actor:1, goal:1 and actor:2 at the top-left side in Figure 1 shows such a pair. We call a relationship between the goal and an actor who wants to achieve the goal as a *want-relation*, and another relationship between the goal and another actor who can achieve the goal as a *can-relation*. We attach an attribute called "level" to the want-relation and the can-relation respectively. The attribute takes a value from 1 to 10. When an actor really wants to achieve a goal, the level of its want-relation takes large value. Otherwise, the level takes small one. When an actor can achieve a goal almost completely, the level of its can-relation takes large value. For example in Figure 1, the level of a want-relation between actor:2 and goal:4 takes ten because the secretary (actor:2) really want to get better wages (goal:4). However, the level of a can-relation between actor:5 and goal:4 takes 3 because it is hard for his boss (actor:5) to give better wages. Can-relations and want-relations have another attribute called "rate". The attribute will be explained in the next section because the attribute is an intermediate attribute to calculate metrics of a SD model.

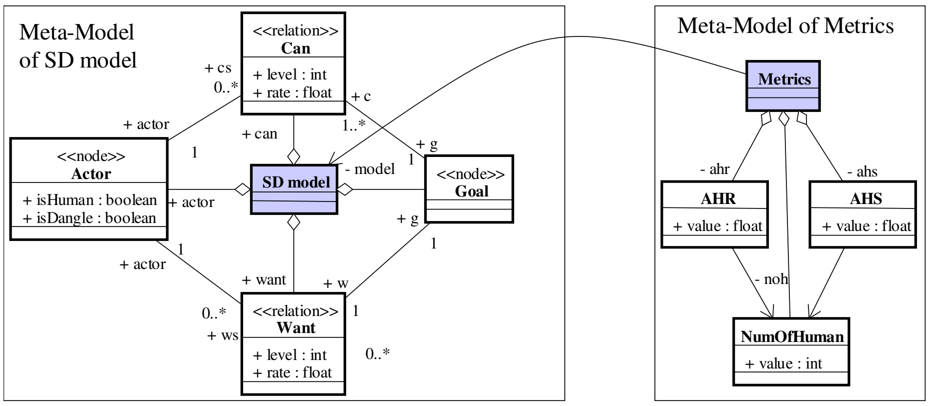
We explain two attributes on actors: *isHuman* and *isDangle*. Both attributes take Boolean value. Distinguishing human actors from other actors is important because one of the policies of our model transformation is to minimize the responsibility of human and to maximize the satisfaction of human. An attribute isHuman is used for this purpose. In original i*, completely isolated actors such as actor:10 in Figure 1 are called dangling actors, and such actors should not be contained in a model [13]. In our extension, we call any actors with true value in isDangle attribute as dangling actors such as actor:10 and actor:6 in Figure 1. We then permit a model to contain such dangling actors because we want to record potential alternatives of dependers and dependees in the model so that we can easily explore another possibilities of actor dependencies. In addition, we permit a goal to have more than one candidate of dependees in the same reason. However, only one dependee has to have isDangle attribute as false and the others has to have isDangle attribute as true even if a goal has more than dependees. This constraint is defined as an OCL expression (context Goal) of the meta-model in Figure 2. For example in Figure 1, goal:5 has two dependees actor:6 and actor:2, but only actor:2 takes false value in isDangle.

Original i* has several types of goals (more precisely intentions) such as goals, soft-goals, tasks and resources. However, we only use goals in our SD model. Goals in our SD model can be used as goals and soft-goals in original i* because goals between a can-relation with level 10 and a want-relation with level 10 can be regarded as goals in original i* and others can be soft-goals. We mainly focus on very early stages of requirements definition, but tasks and resources seem to be related to architecture and/or implementation issues. We thus do not use tasks and resources in our model.

2.2 Graph Rewriting System

In Model Driven Development (MDD), one of the technically essential points is the model transformation. Since we use a class diagram to represent a meta-model, a model, i.e. an instance of the meta-model can be considered as a graph, whose nodes have types and attributes, and whose edges have types, so called attributed typed graph. In this paper, the model transformation is thus defined as a graph rewriting system, and graph-rewriting rules dominate allowable transformations. Here we briefly introduce a graph rewriting system.

A graph rewriting system converts a graph into another graph or a set of graphs following pre-defined rewriting rules. There are several graph rewriting systems such



```
context Can
inv: rate == g.w.level/level

context Want
inv: rate == g.c->select(actor.isDangle==false)->collect(level)->sum() / level

context Goal
inv: c->select(actor.isDangle==false)->size()==1

context Metrics
inv:
noh.value == model.actor->select(isHuman && isDangle==false)->size()
ahr.value ==
  model.can->select(actor.isHuman && actor.isDangle==false)
  ->collect(rate)->sum() / noh.value
ahs.value ==
  model.want->select(actor.isHuman && actor.isDangle==false)
  ->collect(rate)->sum() / noh.value
```

Fig. 2. Meta-model of a SD model and its Metrics

as PROGRESS [10] and AGG [12]. Since we should deal with the attribute values attached to nodes in a graph, we adopt the definition of the AGG system in this paper. An example of AGG can be found in Figure 3 explained in the next section.

3 Metrics and Transformation

We define a meta-model (grammar) of both a SD model and its metrics in Figure 2 to facilitate effective model transformation on a SD model. An instance of a SD model is shown at the bottom of Figure 1. Currently, we use two metrics called average human responsibility (AHR) and average human satisfaction (AHS) with the help of a complementary metric called number of human actors (NumOfHuman) as shown in the figure. The formal definitions of these metrics are shown as the OCL invariants in Figure 2.

We will show how to calculate these metrics by using a SD model in Figure 1 and the intermediate values in Table 1. In the table, the values of Can.rate sum up for each non-dangling actor respectively. For example, 5/5 and 7/9 are summed up for actor:2 because actor:2 can achieve two goals of goal:1 and goal:5 and Can.rate in can-relations between actor:2 and each goal takes 5/5 and 7/9 respectively. The value of Can.rate

Table 1. Intermediate values for calculating AHR and AHS in a SD model in Figure 1. NumOfHuman.value is 4 in this model.

	sum. of Can.rate	sum. of Want.rate
actor:1	0	$\frac{5}{5}$
actor:2	$\frac{5}{5}+\frac{7}{9}$	$\frac{10}{10}+\frac{5}{10}+\frac{3}{10}$
actor:3	$\frac{10}{10}$	0
actor:5	$\frac{10}{3}$	$\frac{9}{7}$
row total	$\frac{5}{5}+\frac{7}{9}+\frac{10}{10}+\frac{10}{3}$	$\frac{5}{5}+\frac{10}{10}+\frac{5}{10}+\frac{3}{10}+\frac{9}{7}$
row total / 4	1.52 (= AHR.value)	1.02 (= AHS.value)

shows relative weight of responsibility with respect to expectation of a depender. For example in Figure 1, actor:5 has to achieve goal:4, and actor:2 wants to achieve the goal. In this case, Can.rate between actor:5 and goal:4 takes 10/3 (= 3.3). We may regard actor:5 takes a really heavy responsibility about goal:4 because this value means actor:2's expectation is about three times as actor:5's ability. On the other hand, actor:3 takes a reasonable responsibility about goal:2 because Can.rate between actor:3 and goal:2 is 10/10 (= 1.0). The row total of "sum. of Can.rate" shows the total of such responsibility. We finally divide the value of row total by the number of non-dangling human actors, and we can get the AHR.value, 1.52 in this case.

How to get AHS.value is almost the symmetrical way above. The value of Want.rate shows relative weight of satisfaction with respect to ability of a dependee. In our extended SD model, we permit a goal to have more than one dependee such as actor:6 and actor:2 of goal:5 in Figure 1. However, we only use a non-dangling actor in such a case, and our OCL (context Goal) guarantees there is only one non-dangling actor.

Because metrics AHR and AHS can be calculated from any SD model in conformance with the meta-model in Figure 2, we can observe changes of these metrics during any model transformation. We regard a model transformation is good when AHR decreases because systems make the responsibility of human to be decreased. We also regard a model transformation is good when AHS increases because systems have to increase satisfaction of human. Currently, we only have two metrics but we may append any metrics that are useful to evaluate model transformation.

Figure 3 shows an example of a model transformation using AGG. The as-is model in the figure is a part of the SD model in Figure 1. Three patterns of graphs such as NAC, LHS and RHS in AGG correspond to a model transformation rule. When a part of graph is matched to LHS and the part is not matched to NAC, the part is transformed into RHS. Because each node in a graph pattern may contain variables and variables in RHS can be defined based on variables in LHS, metrics and its changes can be naturally represented in a model transformation rule written in AGG. In this example, AHR varies from 0.388 (=0.7777/2) to 0, and AHS varies from 0.64 (=1.28/2) to 0.71. This model transformation thus causes the reduction of the human responsibility and the gain of the human satisfaction.

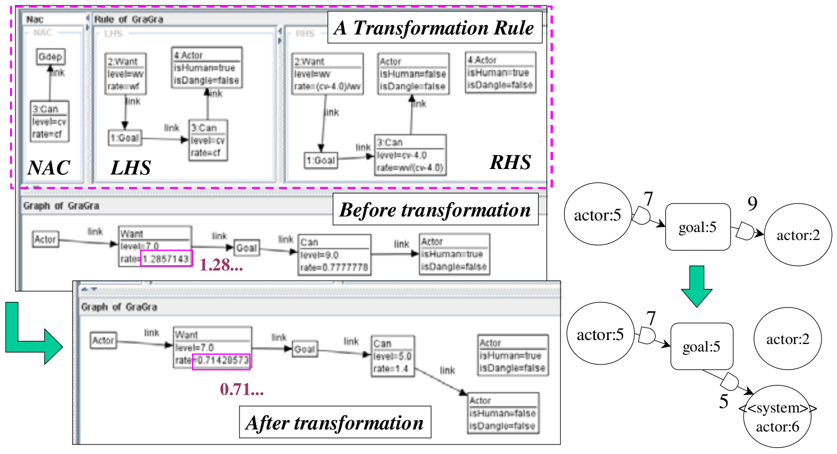


Fig. 3. Example of a model transformation using AGG. An as-is model in this figure is a part of the SD model in Figure 1.

4 Related Work

Activity-based Costing (ABC) [1] and Balanced Score Card (BSC) [6] are famous methodologies to explore better to-be models of business, and they focus on several attributes such as costs, performance, time, knowledge and so on. Such kinds of attributes can be introduced in our SD meta-model, or our meta-models can be explicitly related to other meta-models such as business process with such attributes. These methodologies are useful to explore problems of an as-is model, but to-be models are not explicitly specified. In our method, the model transformation technique explicitly specifies such to-be models.

We can find several researches for transforming as-is business process to to-be one [9, 8, 4], and some of them use metrics for facilitating better transformation. Our research mainly focuses on strategic dependencies that are earlier than processes with respect to clarifying requirements. As mentioned in the previous section, the meta-modeling technique enables us to make explicit relationships between early requirements (SD model) and other concerns such as process, architecture and implementation. This explicit relationships and the separation of concerns help us to integrated and rational decision for business improvement. Because transformation among such different concerns is already proposed [2], it is not so difficult to define such relationships.

Original i* [14] has a lot of vocabularies such as a goal, a soft-goal, a task, a resource, an actor, an agent, a role, a position and so on. We however use only a goal and an actor because these two are the fundamental elements of a SD model. There are a lot of extensions of i* [5, 7, 11] and they have more vocabularies than original one. Our method in this paper can be extended naturally according to each extension because such extended vocabularies can be formalized by attributes attached to goals, actors, can and want-relations. There are some researches about i* model using metrics (summary can be found in [3]), but there are few ones focusing on changes of the metrics.

5 Conclusion

In this paper, we proposed a method to improve an as-is model of business by using a model transformation technique and metrics on the model. We use i* SD model for representing business models and AGG for model transformation. Currently, we only focus on strategic dependencies among actors in business, but we have to take into account more information such as business process, architecture, implementation and so on. Our method can easily take into account such additional information because it uses meta-modeling technique useful for make explicit relationships among different aspects of the business. We want to extend our current meta-model including metrics in such a way.

Acknowledgement This research is supported by Takahashi Industrial and Economic Research Foundation, Japan, 02-001-A24-1009 and MEXT/JSPS KAKENHI Grant Number 23500042.

References

1. Robin Cooper, Robert S. Kaplan, and Lawrence S. Maisel. *Implementing Activity-Based Cost Management: Moving from Analysis to Action*. Inst of Management Accountants, Oct. 1993.
2. Ken Decreus, Monique Snoeck, and Geert Poels. Practical challenges for methods transforming i* goal models into business process models. In *RE*, pages 15–23, 2009.
3. Xavier Franch and Gemma Grau. Towards a catalogue of patterns for defining metrics over i*models. In *CAiSE*, pages 197–212, 2008.
4. Kaori Fujiwara, Bala Ramachandran, Akio Koide, and Jay Benayon. Business process transformation wizard: a bridge between business analysts and business process transformation technology. In *IEEE SCC*, pages 83–90, 2007.
5. Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Modeling security requirements through ownership, permission and delegation. In *RE*, pages 167–176, 2005.
6. Robert S. Kaplan and David P. Norton. *The Balanced Scorecard: Translating Strategy into Action*. Harvard Business School Press, Sep. 1996. `http://www.balancedscorecard.org`.
7. Haralambos Mouratidis and Paolo Giorgini. Secure tropos: a security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):285–309, 2007.
8. Marion Murzek, Gerhard Kramler, and Elke Michlmayr. Structural patterns for the transformation of business process models. In *EDOC Workshops*, page 18, 2006.
9. Mariska Netjes, Hajo A. Reijers, and Wil M. P. van der Aalst. On the formal generation of process redesigns. In *Business Process Management Workshops*, pages 224–235, 2008.
10. Andy Schürr. Developing graphical (software engineering) tools with progres. In *ICSE*, pages 618–619, 1997.
11. Alistair G. Sutcliffe. Trust: From cognition to conceptual models and design. In *CAiSE*, pages 3–17, 2006.
12. Gabriele Taentzer. Agg: A graph transformation environment for modeling and validation of software. In *AGTIVE*, pages 446–453, 2003. `http://user.cs.tu-berlin.de/~gragra/agg/`.
13. Eric Yu, Jaelson Castro, and Anna Perini. Strategic Actors Modeling with i*, RE 2008 - Tutorial, Aug. 2008.
14. Eric S. K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *RE*, pages 226–235, 1997.