Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2021) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2021)

Tools for logging and analyzing goal dependency modeling

Haruhiko Kaiya[a,*], Kouta Hayashi[a], Yu Sato[a]

[a]*Kanagawa University, Hiratsuka 259-1293, Japan*

Abstract

Modeling goal dependencies among people and systems contributes to exploring the systems valuable to the people. Therefore, guiding such modeling processes is useful for the system development. Because there is no predefined process model of good goal dependency modeling, we have to clarify the characteristics of actual modeling processes for a start. In this paper, we introduce the tools for logging and analyzing goal dependency modeling. We also report a case study to use the tools so as to find common characteristics of the modeling processes. We expect such tools help us to find how to guide a modeling process. The logging tool is developed by expanding an existing modeling tool called astah. Each developer can thus easily record the logs with his/her familiar modeling tool. The analyzing tools finally visualize each modeling process in several ways by using Graphvis. We can observe the development process, and analyze the common characteristics with the help of these tools. Although we cannot find the unknown and unique characteristics, we can confirm some reasonable characteristics. For example, the ratio of transitions between goals and other elements is larger than the other ratio.

Keywords: Modeling tool; Logging; Process Mining; Goal Modeling

1. Introduction

In many manufacturing fields, people believe good development processes contribute to resulting good products. In the case of routine of the office work, we can define its ideal process model in advance. We can thus check the conformance of each process instance against the model. However, most processes in an information system development are usually creative and each of them cannot have its ideal process model in advance without analyzing the actual processes. We have been studied a requirements engineering method using the notation of goal dependencies [8]. Because we of course want the goal dependency models to be high quality, we want to improve their development processes. As the first step based on this research objective, we developed the tools for logging and analyzing the development processes of goal dependency modeling. We also performed a preliminary case study to use the tools.

* Corresponding author. Tel.: +81-463-59-4111.
E-mail address: kaiya@acm.org

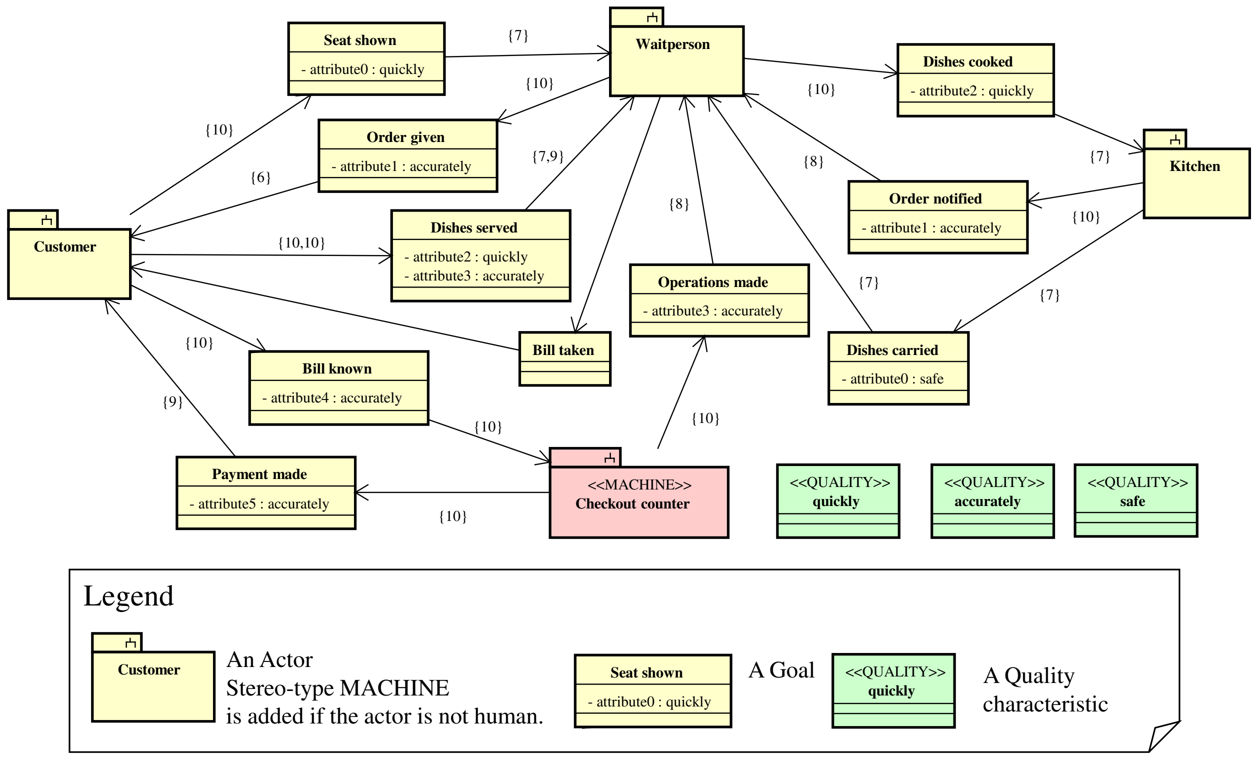


Fig. 1. An example of the goal dependency model

The rest of this paper is organized as follows. In the next section, we introduce a notation of our goal dependency modeling. The notation is an extension of istar. The notation can be represented in a UML class diagram. We then explain the logging tools in detail in section 3. The tool is developed by expanding an existing modeling tool. Section 4 shows analyzing tools and the example of their usage. In section 5, we show a case study. In the case study, we asked several students to use the tools, and analyzed the results to find common characteristics of modeling processes. We briefly review related works in section 6. Section 7 shows the summary of our current results and the future issues.

2. Notation of Goal Dependency Modeling

The tools in this paper record and analyze the logs of a goal dependency modeling. Its notation is explained in this section. The notation is based on istar [13]. Because the notation can be represented by using UML class diagrams [7], most software developers can describe its models by using their familiar modeling tools.

By using an example of a goal dependency model in Figure 1, we briefly explain the characteristics of the notation. In this figure, a typical restaurant in Japan is modeled. We expect information systems can improve the quality of the restaurant. Exploring such systems is one of the initial problems in requirements engineering. This problem is used in the case study in section 5.

In this notation, a goal is represented in a class without stereo-type “QUALITY”. An actor is either a person or a system who wants someone to achieve a goal or who will achieve it. An actor is represented in a subsystem. For example at the left and top in the figure, we can find two actors “Customer” and “Waitperson”. At the restaurant, customers initially want a waitperson to take them to a seat. Such issue is represented as a goal “Seat shown” in the figure, and an arrow from “Customer” to the goal specifies that the “Customer” wants to achieve the goal. Another arrow from the goal to “Waitperson” specifies that the “Waitperson” is expected to achieve the goal. When an actor is not a person, a stereo-type “MACHINE” is attached as shown at the bottom in the figure so that we can clearly identify the contributions of artificial elements such as hardware or software systems.

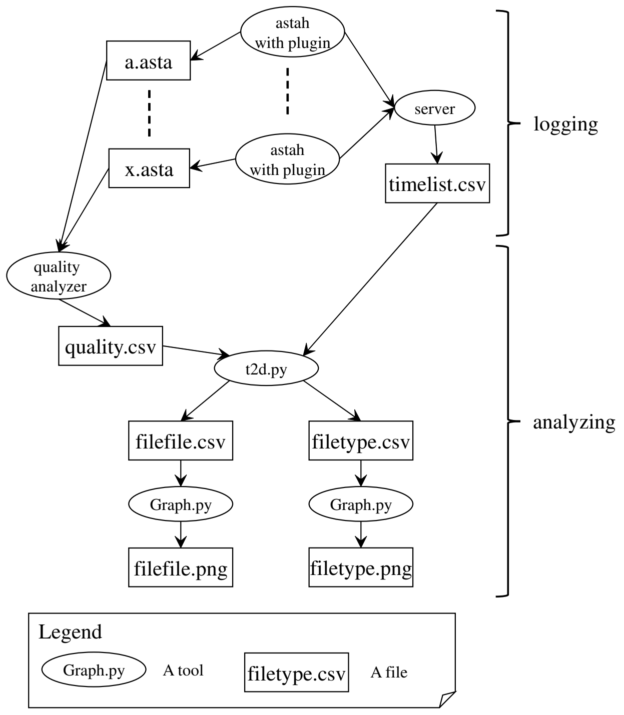


Fig. 2. Tools and their data flow

To specify an extent of expecting and achieving a goal, quality attributes can be specified in the notation. A quality attribute is represented in a class with a stereo-type "QUALITY" and exemplified at the bottom right in the figure, e.g. "quickly", "accurately" and "safe". Several quality attributes may be attached to each goal. For example, a goal "Seat shown" has one "quickly", and another "Dishes served" has two attributes "quickly" and "accurately". When a goal has several attributes, the same number of values should be attached to an arrow from/to the goal. The values attached to an arrow to a goal show the extent of expectation when the goal is achieved. Each value takes from 1 to 10 in this notation, and 1 shows the minimal expectation and 10 shows maximal expectation. According to the figure, a "Customer" wants to achieve a goal "Dishes served" quickly and accurately. The model in the figure shows the extents of quickness and accuracy are maximal because an arrow between "Customer" and "Dishes served" has values 10 and 10. The values attached to an arrow from a goal to an actor show the extents of ability achieving the goal by the actor. Unfortunately, the goal "Dishes served" seems to be achieved not so quickly but almost accurately according to the model because the values from "Dishes served" to "Waitperson" are 7 and 9. The precise notation in each actor allows us to specify how to decompose delegated goals to each actor. However, we do not explain it because we do not use it in the case study in section 5.

3. Tools for logging modeling activities

Figure 2 depicts the data flows among tools for logging and analyzing the modeling activities. Ovals in the figure show tools, and rectangles show files generated or used by a tool.

To record the logs of modeling processes, we extended an existing modeling editor astah [3]. Each editor generates a model file such as "a.asta" or "x.asta" as shown in Figure 2. More than one models are sometimes described during a modeling process. For example, as-is and to-be models are edited simultaneously. Therefore, an extended astah editor simply monitors the editing actions and a server records the actions monitored by several editors into a logging file.

The extended editor can monitor the following kinds of actions:

- start an editor.

- add an element to a diagram such as a class, a subsystem and an association.
- modify an element.
- delete an element.
- select an element.

The type of these actions are sent to the server with the file name, the name of diagram, the identifier of the corresponding element, its label string and its type. The extended editor cannot monitor the selection of a class when the type of an attribute is a class. The extended functionalities above are implemented as plugin-in of astah.

The server then simply records the logs in CSV format file “timelist.csv” as shown in Figure 2. Each line contains the following values:

- a file name containing diagrams edited by astah
- a label string of an edited element
- a type classifier name such as “Class”, “Subsystem” or “Association”
- a diagram name. Note that astah enables its output file to store several diagrams.
- a type classifier of an action such as “add”, “modify” and so on.
- an identifier of an edited element such as “2re-c58fb693f77284b554a56a8466d77d7a”
- a timestamp (an integer value of millisecond)

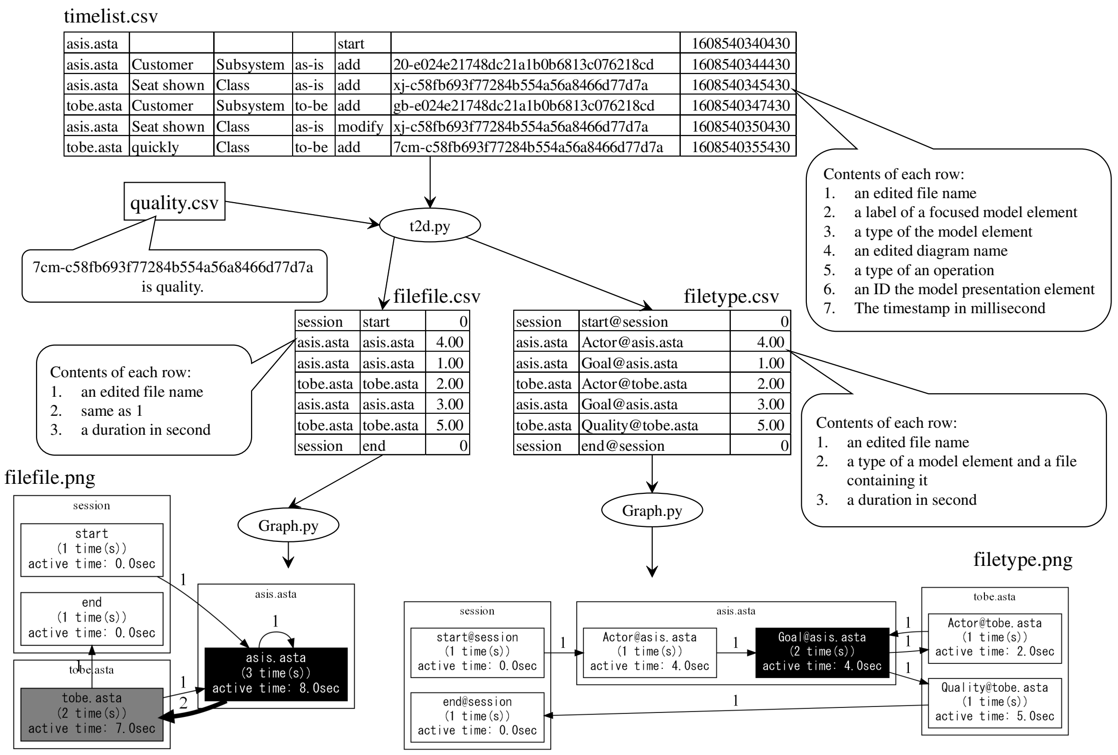
4. Tools for analyzing the log

To analyze the modeling processes, three types of tools are used as shown in Figure 2. In our goal dependency modeling notation explained in the previous section, classes defining quality have a stereo-type QUALITY as exemplified in Figure 1, and classes defining goals do not have such stereo-type. A tool “quality analyzer” distinguishes quality classes from classes in a model file such as “a.asta”. The result is stored in a file “quality.csv” as shown in Figure 2. The tool is implemented in Java by using astah API.

By using two files “quality.csv” and “timelist.csv”, several CSV files suitable for our visualizing tool “Graph.py” are generated by a tool “t2d.py”. These two tools are implemented in Python. As explained in the previous section, a file “timelist.csv” simply stores the list of events with a timestamp. We regard the duration of an event as the difference between the timestamp of the previous event and the timestamp of the event. For example at the top in Figure 3, “timelist.csv” contains six events. Note that the last column of “timelist.csv” is the timestamp in millisecond. The tool “t2d.py” then calculates the duration of each event in second as shown in the last column in “filefile.csv” and “filetype.csv”. Note that the event “start” is excluded. The tool also adds two pseudo-events “start” and “end” to each csv file so that we can identify first and last events in a session. A session consists of a continuous sequence of events. A developer usually describes a model several times during a period. For example, he/she describes a model three times, e.g., in Monday, Wednesday and Friday. A session intuitively corresponds to such a time.

The file “filefile.csv” is used to identify the transitions between files of as-is and to-be models. The tool “Graph.py” is a general visualizing tool of event transitions using Graphviz [6] on the basis of a book [5]. The tool reads the list of three values. The first value specifies a group of events, and the next specifies a type of each event. The last specifies the duration of each event. When the tool read “filefile.csv”, it generates the directed graph about the transitions among files. For example at the bottom left in Figure 3, the file “asis.asta” is firstly edited. After that, the same file is edited once, and the file “tobe.asta” is edited twice. The node of the graph reports the frequency and total duration of an event. For example in the figure, we can know the file “asis.asta” is edited three times, and the total duration is eight seconds.

The file “filetype.csv” is used to identify the transitions among the types of goal dependency model elements such as “Actor”, “Goal”, “Quality” and “Dependency”. As mentioned in section 2, we use class diagrams to describe goal dependency models, and both goal and quality are represented in a class. Only a class corresponding to quality has a stereo-type “QUALITY” in our representation. To distinguish goal and quality, a file “quality.csv” is used. For example of the “timelist.csv” in Figure 3, “t2d.py” can find two classes “Seat shown” and “quickly”, but it cannot categorize each class into goal or quality. Because the file “quality.csv” contains the results of analyzing “asis.asta”


timelist.csv
quality.csv
7cm-c58fb693f77284b554a56a8466d77d7a is quality.
t2d.py
Contents of each row:
1. an edited file name
2. a label of a focused model element
3. a type of the model element
4. an edited diagram name
5. a type of an operation
6. an ID the model presentation element
7. The timestamp in millisecond
filefile.csv
filetype.csv
Contents of each row:
1. an edited file name
2. same as 1
3. a duration in second
Contents of each row:
1. an edited file name
2. a type of a model element and a file containing it
3. a duration in second
Graph.py
Graph.py
filefile.png
filetype.png
session
start (1 time(s)) active time: 0.0sec
end (1 time(s)) active time: 0.0sec
tobe.asta
tobe.asta (2 time(s)) active time: 7.0sec
asis.asta
asis.asta (3 time(s)) active time: 8.0sec
1
1
1
2
session
start@session (1 time(s)) active time: 0.0sec
end@session (1 time(s)) active time: 0.0sec
asis.asta
Actor@asis.asta (1 time(s)) active time: 4.0sec
Goal@asis.asta (2 time(s)) active time: 4.0sec
tobe.asta
Actor@tobe.asta (1 time(s)) active time: 2.0sec
Quality@tobe.asta (1 time(s)) active time: 5.0sec
1
1
1
1
1
1

timelist.csv

asis.asta				start		1608540340430
asis.asta	Customer	Subsystem	as-is	add	20-e024e21748dc21a1b0b6813c076218cd	1608540344430
asis.asta	Seat shown	Class	as-is	add	xj-c58fb693f77284b554a56a8466d77d7a	1608540345430
tobe.asta	Customer	Subsystem	to-be	add	gb-e024e21748dc21a1b0b6813c076218cd	1608540347430
asis.asta	Seat shown	Class	as-is	modify	xj-c58fb693f77284b554a56a8466d77d7a	1608540350430
tobe.asta	quickly	Class	to-be	add	7cm-c58fb693f77284b554a56a8466d77d7a	1608540355430

filefile.csv

session	start	0
asis.asta	asis.asta	4.00
asis.asta	asis.asta	1.00
tobe.asta	tobe.asta	2.00
asis.asta	asis.asta	3.00
tobe.asta	tobe.asta	5.00
session	end	0

filetype.csv

session	start@session	0
asis.asta	Actor@asis.asta	4.00
asis.asta	Goal@asis.asta	1.00
tobe.asta	Actor@tobe.asta	2.00
asis.asta	Goal@asis.asta	3.00
tobe.asta	Quality@tobe.asta	5.00
session	end@session	0

Fig. 3. An example of converting timelist.csv and quality.csv into visualized transition graphs

- After a waitperson shows a seat for a customer, the customer browses a menu card to choose items. The customer calls a waitperson when the customer has chosen the items.
- The waitperson takes up the order of the customer, and the order is passed to the kitchen by using an order form.
- A waitperson[a] takes the dishes and their invoice to the customer.
- The customer of course eats the dishes.
- The customer with the invoice goes to the checkout counter to make the payment. A waitperson checkouts the order of the customer by using the invoice.
- Cash and the credit cards are acceptable in this restaurant.
- A discount coupon is also acceptable.

Please explore the introduction of ICT systems in this restaurant although the restaurant works well now.

[a]Any waitpersons can serve any customers in Japan. Therefore, the waitperson taking up the order could be different form this waitperson.

Fig. 4. A document of the tasks in a restaurant

and “tobe.asta”, “t2d.py” can convert “timelist.csv” into “filetype.csv” as shown in this example. In the same way of “filefile.csv”, “filetype.csv” can be visualized as shown in this example.

	asis	asis	tobe	tobe
Students	freq	min	freq	min
A	296	92.7	442	48.1
B	291	16.3	450	26.3
C	389	255.0	382	61.8
D	833	67.3	747	47.1
E	411	32.4	360	31.6

asis	tobe
average min.	
0.313	0.109
0.056	0.058
0.656	0.162
0.081	0.063
0.079	0.088

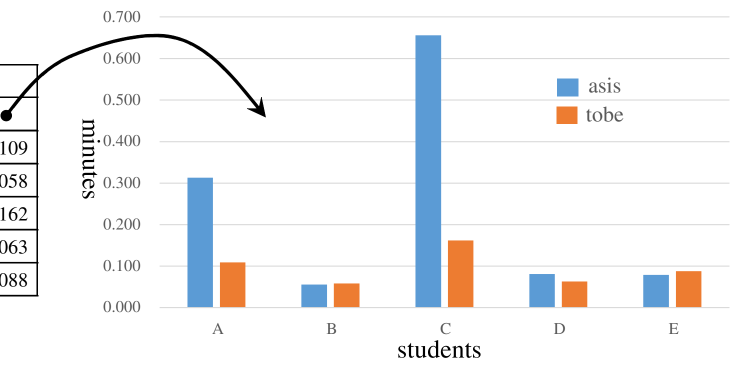


Fig. 5. Frequency and duration of events in as-is or to-be model

5. Case Study

5.1. *Objective and Settings*

The objective of this case study is to know the characteristics of goal dependency modeling processes. We have no explicit hypotheses except there are some common characteristics among several different developers. We asked five students to develop as-is and to-be models of goal dependencies exemplified in Section 2. Although they are third year bachelor students, they had already had the experiences to describe goal dependency models by using UML class diagrams. Firstly, a document of the tasks in a restaurant in Figure 4 and four actors in Figure 1 were given to them. The document contains seven descriptions each of which explains a task performed in a restaurant. Each student had to submit models and logs within a week. They may describe the models anywhere and any times.

Usually in as-is/to-be analysis, a developer first describes an as-is model. He/she then makes a copy of the as-is model so that he/she can efficiently describe a to-be model by modifying the copy. We forbade students describing such a way so that we could monitor the process in describing a to-be model.

5.2. *Results and Discussion*

Table 1. Size of models and the number of sessions

	asis				tobe				
Students	Actor	Goal	Quality	Dependency	Actor	Goal	Quality	Dependency	session
A	4	10	4	18	8	14	4	28	3
B	4	10	3	20	8	14	3	28	1
C	4	8	3	16	6	9	3	18	4
D	5	10	6	20	5	11	6	22	3
E	4	11	4	22	5	10	4	11	2

Table 1 shows the size of models and the number of sessions. The data in this table can be obtained from a file “filetype.csv” in Figure 2, but the table itself is described manually. A session is a continuous duration of a modeling process. If the number of the session is 3, the student had described his/her models in Monday afternoon, Wednesday night and Friday morning for example. Because initial four actors were given, the sizes of the models among students are not so different from each other. Although each student could introduce new quality attributes into the to-be model, no one did it as shown in the table.

We first focus on the frequency and the duration in as-is and to-be models respectively, i.e. we focus on the values in the boxes of “filefile.png”. Figure 5 shows the results. The left table in the figure shows the frequency and the duration of events in as-is or to-be modeling. For example, Student B performed some actions in 291 times, and his/her total spending time is 16.3 minutes while he/she describe an as-is model. These two values can be derived from a black

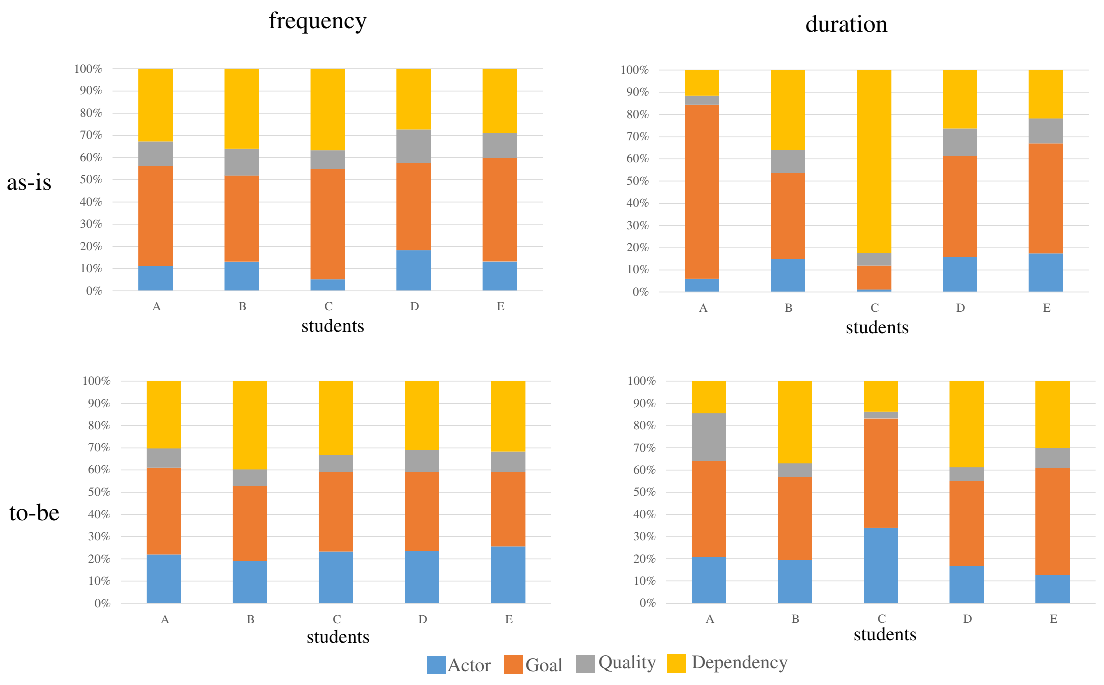


Fig. 6. Frequency and duration Ratio of model element types

box of "filefile.png" in Figure 3 although the file in the figure is not generated from the data of Student B. The middle table in the figure shows the average duration of events. For example of an as-is modeling by Student B, 16.3 divided by 291 simply gives 0.056. According to the bar chart at the right of the figure, the average duration of as-is modeling is almost larger than one of to-be modeling. At the as-is modeling, a developer has to understanding the business activities to be modeled. Therefore, it takes additional hours to describe the as-is model. If a developer has little idea to introduce systems into as-is situation, it also takes much hours to describe the to-be model. However, we have already known many restaurants introducing ICT technologies. Therefore, students could efficiently describe to-be models as shown in the results.

We second focus on the frequency and the duration with respect to the type of the elements, i.e. we focus on the values in the boxes of "filetype.png". Figure 6 shows the frequency and duration ratio of model elements. A left-top bar chart shows the frequency ratio for each student in an as-is model. For example, about 10% of Students A's events is applied to Actor, and about 45% is applied to Goal. Left-bottom, right-top and right-bottom bar charts show the frequency ratio in a to-be model, the duration ratio in an as-is model and the duration ratio in a to-be model respectively. Left two bar charts can be derived from the values with a prefix "time(s)" of "filetype.png" in Figure 3 although the file is not generated from an actual student. Right two bar charts can be also derived from the values with a postfix "active time". According to the bar charts, the ratio of Actor in a to-be model is almost larger than the ratio in an as-is model. In an as-is model, Actors were given. However in a to-be model, each student had to explore additional Actors by him/herself. This seems to be one of reasons of ratio's increase. Figure 7 shows the average duration of actions for each type. For example at the top left of the figure, Student A took about 0.2 minutes to operate Actor elements of an as-is model in average. The average ratio of Actor in a to-be model seems to be smaller than one in an as-is model. Most students spent much time to understand the Actor when they developed an as-is model, but they spent much time to understand the relationships between Actors and others. We think that is one of the reasons of this result.

We third focus on the transitions between models. In this case study, all students could edit the as-is and to-be models simultaneously. We thus expected transitions between different models could be observed. For example, an as-is model is reedited after a to-be model is described. Table 2 shows such results. The results can be derived from the values attached to arrows of "filefile.png" in Figure 3 although the file is not generated from the data of a

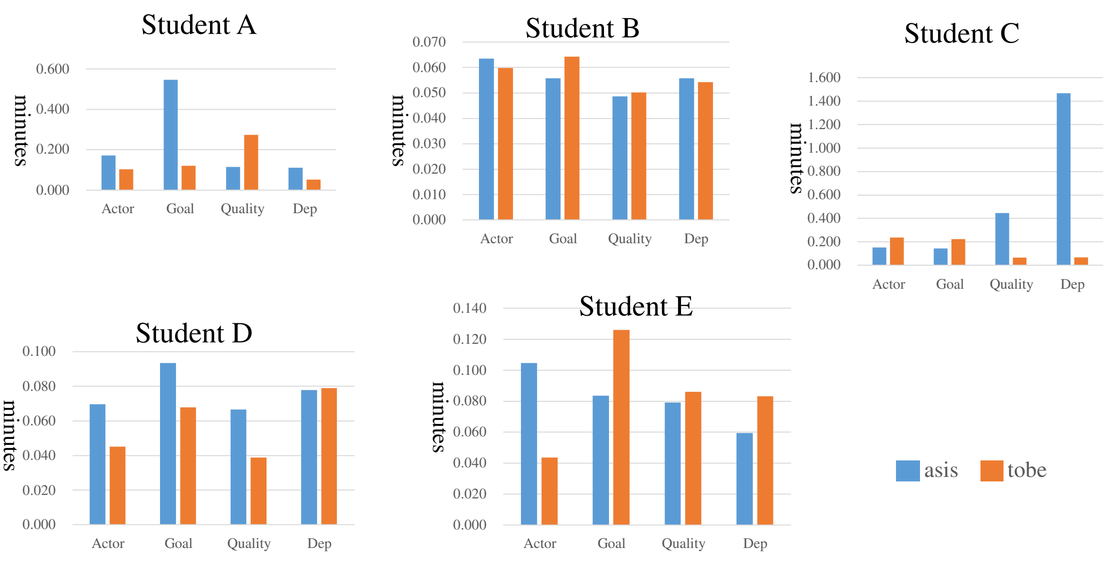


Fig. 7. Average duration of model element types

Table 2. The number of transitions within/between as-is and to-be models

Students	asis-asis	tobe-tobe	asis-tobe	tobe-asis
A	295	440	0	0
B	289	448	2	1
C	383	378	3	3
D	827	743	3	4
E	410	359	0	0

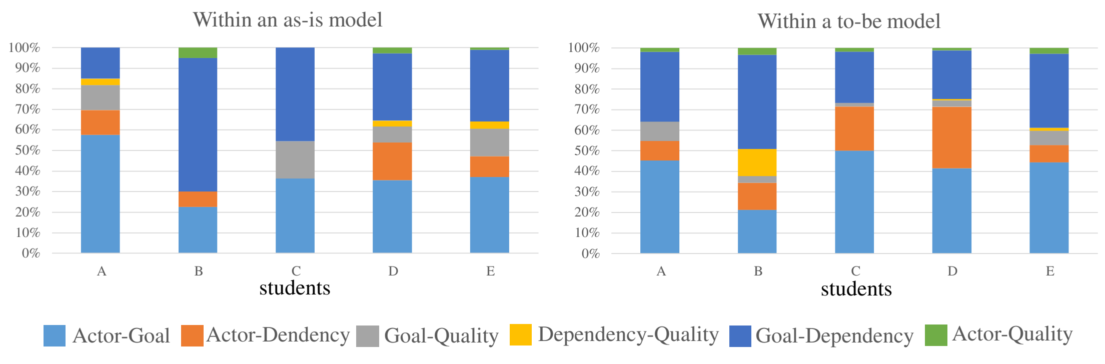


Fig. 8. Ratio of transitions between different element types

student. Against our expectation, most students edited the as-is model and the to-be model almost respectively. A few transitions between different models can be observed in several students. From this results, we assume students can easily understand the as-is business without considering ICT introduction into it.

We finally focus on the transitions among different element types. Figure 8 shows the results. In the results, we do not distinguish the directions of the transition. For example, transitions from goals to actors and those from actors to goals are categorized into the same type “Actor-Goal”. We also omit the self-transitions from this results. There are no common characteristics among the as-is models of students as well as the to-be models. The characteristics depend on each student. In most cases, the ratios of Actor-Goal and Goal-Dependency take the large portion. Because the students describes the goal dependency models, this result is not strange. Except Student B, the ratio of Goal-Quality in the as-is model is larger than one in the to-be model. From this result, we regard the quality attributes related to a goal are investigated at the as-is modeling, and the results are carried forward at the to-be modeling.

Although this case study is not an experiment, it is a kind of empirical studies. We thus discuss its threats to the validity. One of the serious issues is a construct validity about the duration on an event. In our definition, the duration of an event is calculated on the basis of the difference between the timestamp of the previous event and one of the event. We cannot say a student has investigated a model element during all the duration. In addition, a student can investigate models and their elements anytime when he/she does not use the modeling tools. We cannot easily mitigate or avoid this threat. A threat to external validity of course exists because we have the data of only five students. Goal dependency modeling is not so common in the industrial software development. Therefore, we assume the data of practitioners are not so different from those of the students. In this case study, we prohibited all students from copying and modifying the as-is model when each of them described his/her to-be model. This could cause a threat to internal validity because the frequency and the duration of events about elements in a to-be model are affected by this factor.

6. Related Work

A modeling process is not a task in a fixed from but a task requiring creativity. Therefore, it is not easy for us to define its process-oriented model. There are a lot of researches of business process mining [1], and most of them focus on processes that can be defined in the process-oriented models. We can regard the modeling process can be defined by the decision-oriented model, and processes defined by such model can be analyzed by using intentional mining [10]. However, techniques and examples of intentional mining are very few.

We briefly review the application of process mining techniques to software engineering activities. In [2], the authors tried to discover the data models from event logs. In [9], the authors tried to find the segments of source codes of legacy systems to be maintained by using their own process mining tool. In [4], a validation and verification method of process models was proposed based on the extended UML notation. As a matter of course, the studies did not focus on the creative aspect in software development activities. In [12], logs of a software modeling activity are recorded and the logs are used to find the difficulty points. Metrics are used to identify the patterns in the logs in the paper. Defining metrics of logs is a reasonable way to identify characteristics in a modeling activity.

7. Conclusion

In this paper, we introduce the tools for logging and analyzing goal dependency modeling. The goal dependency models are useful to explore information systems introducing an as-is business or life activity. The as-is/to-be analysis is thus usually used in goal dependency modeling. To extend an existing modeling editor “astah” by using its plugin mechanism, developers can record the logs of their modeling activities without additional efforts. The logs are sorted and analyzed by the tools written in Java and Python. The visualized transition graphs of modeling processes can be obtained with the help of Graphvis. To confirm the usage to find the common characteristics of goal dependency modeling, we performed a case study. Five students participated in the case study, and we could successfully record and analyze their logs. Although we could not find the unknown and unique characteristics of the modeling activities, we could confirm some reasonable characteristics. For example, the ratio of transitions between goals and other elements is larger than the other ratios. We finally point out some future issues. We want to use general and extensible analyzing tool such as ProM [11] because we want to find the characteristics that our analyzing tool cannot find. From the engineering point of view, we want to find the usage of analytic results of the logs such as education or the improvement of the development efficiency. We have to distinguish good processes from bad ones when we improve the processes. To do this, we have to find correlation between the quality of products and one of processes.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers 21K11837, 21K11895 and 18K11249.

References

[1] van der Aalst, W., 2016. Process Mining, Data Science in Action. 2nd ed., Springer.

[2] Bano, D., Weske, M., 2020. Discovering data models from event logs, in: Dobbie, G., Frank, U., Kappel, G., Liddle, S.W., Mayr, H.C. (Eds.), Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings, Springer. pp. 62–76. URL: https://doi.org/10.1007/978-3-030-62522-1_5, doi:10.1007/978-3-030-62522-1_5.

[3] Change Vision, Inc., last accessed Mar. 2021. astah professional. URL: https://astah.net/products/astah-professional/.

[4] Estañol, M., Sancho, M., Teniente, E., 2015. Verification and validation of UML artifact-centric business process models, in: Zdravkovic, J., Kirikova, M., Johannesson, P. (Eds.), Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings, Springer. pp. 434–449. URL: https://doi.org/10.1007/978-3-319-19069-3_27, doi:10.1007/978-3-319-19069-3_27.

[5] Ferreira, D.R., 2017. A Primer on Process Mining: Practical Skills with Python and Graphviz. Springer Briefs in Information Systems, Springer, Cham. doi:10.1007/978-3-319-56427-2.

[6] Graphviz forum, last accessed May 2021. Graphviz - graph visualization software. URL: https://graphviz.org/.

[7] Kaiya, H., Haga, K., 2017. A CASE tool for goal dependency model with attributes based on an existing UML editor, in: Zanni-Merk, C., Frydman, C.S., Toro, C., Hicks, Y., Howlett, R.J., Jain, L.C. (Eds.), Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference KES-2017, Marseille, France, 6-8 September 2017, Elsevier. pp. 1196–1205. URL: https://doi.org/10.1016/j.procs.2017.08.033, doi:10.1016/j.procs.2017.08.033.

[8] Kaiya, H., Ogata, S., Hayashi, S., Saeki, M., 2016. Early requirements analysis for a socio-technical system based on goal dependencies, in: New Trends in Software Methodologies, Tools and Techniques - Proceedings of the Fifteenth SoMeT_16, Larnaca, Cyprus, 12-14 September 2016, pp. 125–138. URL: http://dx.doi.org/10.3233/978-1-61499-674-3-125, doi:10.3233/978-1-61499-674-3-125.

[9] Kato, K., Kanai, T., Uehara, S., 2011. Source code partitioning using process mining, in: Rinderle-Ma, S., Toumani, F., Wolf, K. (Eds.), Business Process Management - 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011. Proceedings, Springer. pp. 38–49. URL: https://doi.org/10.1007/978-3-642-23059-2_6, doi:10.1007/978-3-642-23059-2_6.

[10] Khodabandelou, G., Hug, C., Deneckère, R., Salinesi, C., 2013. Process mining versus intention mining, in: Nurcan, S., Proper, H.A., Soffer, P., Krogstie, J., Schmidt, R., Halpin, T.A., Bider, I. (Eds.), Enterprise, Business-Process and Information Systems Modeling - 14th International Conference, BPMDS 2013, 18th International Conference, EMMSAD 2013, Held at CAiSE 2013, Valencia, Spain, June 17-18, 2013. Proceedings, Springer. pp. 466–480. URL: https://doi.org/10.1007/978-3-642-38484-4_33, doi:10.1007/978-3-642-38484-4_33.

[11] ProM Project, last accessed Mar. 2021. Prom tools. URL: https://www.promtools.org/doku.php.

[12] Tanaka, T., Mori, K., Hashiura, H., Hazeyama, A., Komiya, S., 2014. Proposals of a method detectiong learners' difficult points in object modeling exercises and a tool to support the method, in: IIAI 3rd International Congress on Advanced Applied Informatics, IIAI-AAI 2014, Kokura Kita-ku, Japan, August 31 - Sept. 4, 2014, IEEE Computer Society. pp. 650–655. URL: https://doi.org/10.1109/IIAI-AAI.2014.136, doi:10.1109/IIAI-AAI.2014.136.

[13] Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J., 2010. Social Modeling for Requirements Engineering. The MIT Press.