Chapter 2
Modeling and Analysis of Social Activity Process

Can Wang and Longbing Cao

Abstract Behavior modeling has been increasingly recognized as a crucial means for disclosing interior driving forces and impact in social activity processes. Traditional behavior modeling in behavior and social sciences that mainly relies on qualitative methods is not aimed at deep and quantitative analysis of social activities. However, with the booming needs of understanding customer behaviors and social networks etc., there is a shortage of formal, systematic and unified behavior modeling and analysis methodologies and techniques. This paper proposes a novel and unified general framework, called Social Activity Process Modeling and Analysis System (SAPMAS). Our approach is to model social behaviors and analyze social activity processes by using model checking. More specifically, we construct behavior models from sub-models of actor, action, environment and relationship, followed by the translation from concrete properties to formal temporal logic formulae, finally obtain analyzing results with model checker SPIN. Online shopping process is illustrated to explain this whole framework.

2.1 Introduction

Behavior refers to the action or reaction of any material under given circumstances and environment.[1] Human behavior has been increasingly highlighted for social activities in many areas such as social computing [20], intrusion detection, fraud detection [8], event analysis [21], and group decision making, etc. In both natural and social sciences and applications, multiple behaviors from either one or multiple

[1] http://dictionary.reference.com

The work of C. Wang is sponsored by Australian Research Discovery Grant (DP0988016).

C. Wang (✉) · L. Cao
QCIS Centre, Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia
e-mail: cawang@it.uts.edu.au
url: http://datamining.it.uts.edu.au/group/

L. Cao
e-mail: lbcao@it.uts.edu.au

L. Cao, P.S. Yu (eds.), *Behavior Computing*,
DOI 10.1007/978-1-4471-2969-1_2,

actors often interact with one another. Such behavior interactions may form interior driving forces that impact underlying social activities or situations, and may even cause challenging problems. Take the online shopping process as an example, the customer and the merchant communicate with each other to guarantee the success of an online transaction through the inspection of a trusted third party. Similar behavior communications are widespread in many applications, such as interactions in social communities and multi-agent systems.

To the best of our knowledge, along with qualitative research in behavior sciences [16], behavior representation has been a typical topic in the AI community. Major efforts on action reasoning [10] and composition [17], behavior coordination [15] and planning [6], and modeling systems rather than behaviors [18], have been made. For instance, Serrano and Saugar [18] exploited the application-independent software connector to specify multi-agent societies rather than agent behaviors. Gu and Soutchanski [10] discussed reasoning about action based on a modified Situation Calculus. Sardina et al. [17] considered behavior compositions when failure presents. In addition, many works on 'behavior modeling' actually refer to behavior recognition [9] and simulation [19] instead of representation and checking, which is different from our focus here. Limited work can be identified on representation [2] and checking [4] complex behavior structures and interactions.

Modeling complex behaviors and their interactions are challenging. In the existing work on behavior modeling strategies, there are major issues: (1) traditional behavior modeling that mainly relies on qualitative methods from behavior and social sciences [16] often leads to ineffective and limited analysis in understanding social activities deeply and accurately; (2) traditional behavior expressiveness is too weak to reveal the fact that behavior plays the key role of an internal driving force [3] for social activities; (3) the existing behavior modeling approaches for social activities are bottom-up techniques, which have too many styles and forms according to distinct situations; (4) the existing work often overlooks the checking of behavior modeling, which weakens the soundness and robustness of behavior modeling methods. Consequently, a unified quantitative representation and checking approach for behavior modeling and analysis is in great demand for social activity studies. In this paper, we take great advantage of the model checking techniques to build a novel and unified modeling and analysis system for social activity processes.

There are different types of formal verification, from the manual proof of mathematical arguments to interactive computer aided theorem proof, and automated model checking. Manual proofs are time-consuming, error-prone, and often not economically viable. Computer-aided theorem provers still require significant expert knowledge. However, a considerable number of works in recent years have been devoted to studying theoretical aspects and application fields of the famous technique model checking [4], which outperforms the manual proof and test with simulation in terms of nondeterminism and automation. Model checking is a system verification technique that is being applied to the design of Information and Communication Technology Systems, ranging from software controlling storm surge barriers [14], through space craft controllers [11] to integrated circuits. Moreover, model checker

SPIN [12] becomes a general tool for verifying the correctness of distributed software models in a rigorous and mostly automated fashion. Model checking is currently attracting considerable attentions, and was given the ACM Turing Award 2007 in recognition of the paradigm-shifting work on this topic initiated a quarter century ago.

This verification technology provides an algorithmic means of determining whether an abstract model, i.e. a hardware or software design, satisfies a formal specification expressed as a temporal logic formula. Based on this principle, we could develop our novel framework, that is *Social Activity Process Modeling and Analysis System* (SAPMAS). Specifically, when given a social activity process in terms of actor, action, environment and relationship sub-models, together with the negative forms of desired properties to be verified, transformations are conducted to convert them into a graphical model and combination of patterns, respectively. Subsequently, corresponding transition system (TS) and temporal logic formulae (TLF) can be obtained by semantic mappings. Then, model checker SPIN is used to output either "no sequence found" or "targeted activity sequences" with the inputs: TS and TLF. Afterwards, distinguished activity patterns can be extracted from the activity sequences attained. In the final stage, we illustrate the proposed system through one social activity process on analyzing online shopping behavior interactions, which shows the promising potential of our new system SAPMAS for handling similar issues in behavior modeling.

Overall, according to the theories and techniques we have used, the most outstanding advantages of our method lie on the following four aspects to overwhelm traditional approaches:

- All the involved procedures complete automatically with solid theoretical backgrounds for effective analysis.
- Logics can easily express concurrent properties accurately to expose the vital role that behavior plays.
- This uniform system can be constructed for varied processes with little or no relevance, such as social network and multi-agent systems.
- The involvement of checking make our behavior model solid and stable.

The paper is organized as follows. In Sect. 2.2, the concepts of behavior model, including sub-models of actor, action, environment and relationship, are specified. Section 2.3 introduces the behavior property that contains combination of patterns and temporal logic. Section 2.4 proposes the whole framework of SAPMAS. A case study of online shopping process is exemplified in Sect. 2.5. We conclude the paper in Sect. 2.6 with research issues.

2.2 Behavior Model

Within the scope of Behavior Informatics (BI) [3], behaviors refer to those activities that present as actions, operations or events as well as activity sequences conducted

by human beings within certain contexts and environments in either a virtual or physical organization. In the sequel, we will establish both graphical and formal ways to model behaviors. The core of behavior modeling contains concepts that enable representation and reasoning about behaviors. Four dimensions are identified to represent a behavior.

- *Actor Sub-model*: The dimension allows for describing the behavior subjects and objects, for example, organizations, departments, systems and people involved in an activity or activity sequence.
- *Action Sub-model*: The dimension allows for tracing the activities, operations and events, etc. happening in an activity or activity sequence.
- *Environment Sub-model*: The dimension allows for depicting the contexts and circumstances surrounding an activity or activity sequence.
- *Relationship Sub-model*: The dimension allows for representing the connections among activities or activity sequences.

2.2.1 Actor Sub-model

The basic concept in actor sub-model is "actor", which designates an entity that conducts and implements activities. More systematically, actors are considered to be structured, that is to say, they may contain other actors. Also, actors are related to each other as so to guarantee the occurrence of actions, if one of them is the subject, then the other one must be the object. It induces three critical attributes of the actor sub-model as follows:

- *Subject(s)*: The entity that issues the activity or activity sequence;
- *Object(o)*: The entity on which a behavior is imposed;
- *Structure(st)*: The hierarchy to reflect the roles of the entities involved.

As we know, for a behavior, a subject must make an action on an object. In this way, a predicate logic form can be induced as $Action(Subject, Object)$, where subject and object are terms, while action is a predicate indicating the relationships between subject and object. Moreover, a hierarchy can be utilized over actors such that an actor can be defined in terms of other actors. Therefore, we could formally represent actor sub-model as $Action(Structure(Subject, Object))$, where structure is a binary function to reveal the specific structures between multiple subjects and objects. In addition, the corresponding subject and object are equipped with communication points, indicating physical or logical relationships among actors involved. The symbols are shown in Fig. 2.1.

2.2.2 Action Sub-model

The basic concept in action sub-model is "action", for one action, it has several associated attributes to describe the features of the so-called action. We could divide

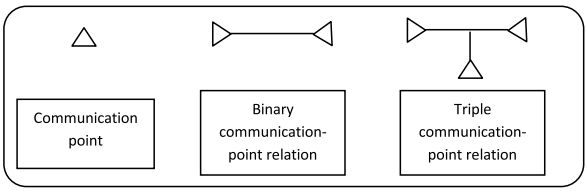

Fig. 2.1 Symbols of the actor sub-model

them into two categories as follows:

- Objective Attribute
  - *Time(t)*: When the behavior occurs;
  - *Place(w)*: Where the behavior happens;
  - *Status(u)*: The stage where a behavior is currently located;
  - *Constraint(c)*: What conditions impact on the behavior.
- Subjective Attribute
  - *Belief(b)*: Information and knowledge of the subject about the world;
  - *Goal(g)*: Targets that the behavior subject would like to accomplish;
  - *Plan(l)*: Sequences of actions that the behavior subject can perform to achieve one or more of its intentions;
  - *Operation(o)*: What the subject has chosen to do or operate;
  - *Effect(e)*: The results or influence led by the execution of a behavior on the object or the context.

In action sub-model, from the perspectives of both predicate logic and temporal logic, we could conduct a formal expression of all these features as follows:

$$Belief(X) \rightarrow \Diamond Goal(X) \rightarrow \Diamond Plan(X) \rightarrow \Diamond Operation(X) \rightarrow \Diamond Effect(X),$$

where X denotes a function as $X = f(Actor, Time, Place, Status, Constraint)$ with five items, eventually $\Diamond$ is one of temporal modalities. Here, the above formula means that if item X has belief, then eventually will set up a goal, this goal will lead to a plan in the future, followed by the actual operation afterwards, then effect will generate finally. It is similar to the well-known BDI (Belief-Desire-Intention) model [22] developed in multi-agent systems, the techniques involved can be adapted in this model as well for our further research.

2.2.3 Environment Sub-model

The basic concept in environment sub-model is "environment", two categories are focused on here. One is related to physical condition, while the other one matters the social relationship. Specifically, they are

- *Context(e)*: The environment in which a behavior is operated;
- *Associate(a)*: Other behavior instances that are associated.

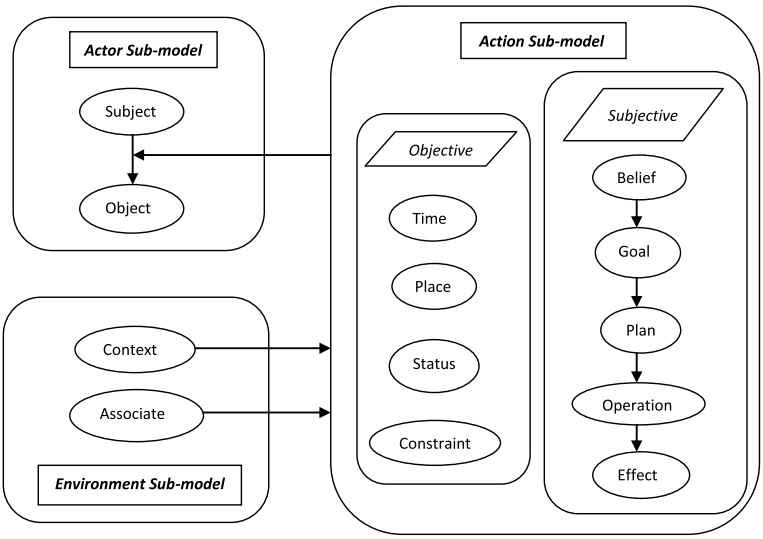

Fig. 2.2 Attributes and links among actor, action, and environment sub-models

In fact, they have been embedded into the above action sub-model. *Context* is essentially the enabling condition, i.e. *constraint* in action sub-model, while *associate* is the relevant *operation* in action sub-model. In logic language, that is $(Context \rightarrow Action) \wedge (Associate \rightarrow Action)$.

Overall, for one behavior, all of the 13 attributes and their links among actor, action and environment sub-models are depicted in the following Fig. 2.2.

2.2.4 Relationship Sub-model

The basic concept in relationship sub-model is "relationship", all of the above three sub-models focus on only one behavior, but for multiple behaviors, we should concentrate on the couplings among them.

As we know, an action can only happen when its enabling condition is satisfied. To some extent, this kind of enabling condition is a constraint which can be formulated in terms of other actions having occurred yet or not. There are several kinds of relationships [7] shown in Fig. 2.3.

In this figure, initiative actions are actions without causality conditions to start the activity sequences. The enabling relationship requests that behavior A must perform before behavior B, while the disenabling relationship means that if behavior A happens, behavior B cannot take place. The or-split relationship represents the situation that one of the latter actions can be chosen after the former one has been done, while the and-split relationship expresses that all of the latter actions will be conducted once the former action has occurred. Similarly, or-join means after selecting

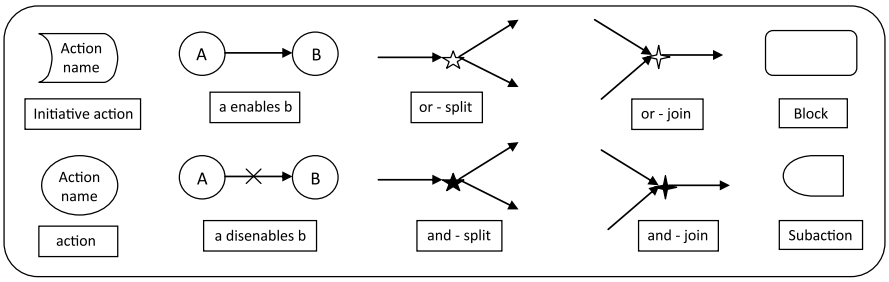

Fig. 2.3 Symbols of the relationship sub-model

Table 2.1 Logic representations of relationship sub-model

Relationship	*enable*	*disenable*	*or-split*	*and-split*	*or-join*	*and-join*
Logic Form	$a \to b$	$\neg a \to b$	$a \to (b \vee c)$	$a \to (b \wedge c)$	$(a \vee b) \to c$	$(a \wedge b) \to c$

one of the former actions to be done, the latter one follows, whereas and-join depicts the scenario that the latter action can only be taken when all of the former ones have completed. These are the basic elements to describe the multiple types of relationships in social activity processes, for some special cases, we may add some symbols or operators if possible. By using them, an instance graphical model is constructed as exemplified in Sect. 2.5. The corresponding logic forms are formalized in the simplest case with merely actions a, b, c as shown in Table 2.1.

However, extensive modeling of all relevant actions and their enabling conditions will lead to a huge, disorderly and unsystematic graphical model. Thus, we need to develop methods to express all of these information in a systematic and hierarchical way. Here, two means are used to tackle the complexity of models. One is to divide the complex graph into blocks according to different stages, while the other is to make decomposition based on distinct actors, which will be illustrated in Sect. 2.5.

2.3 Behavior Property

In order to analyze and mine some special behaviors in an existing system, such as frequent sequence, exceptional sequence and hidden sequence, we must firstly define the properties of those patterns to be mined accurately. State differently, if we aim to discover some frequent patterns, we should give a specific definition of the term "support", then frequent pattern is an activity sequence with support greater than or equal to a given minimum threshold support, while exceptional behavior is concerned on the condition that both "Intentional Interestingness" and "Exceptional Interestingness" are well defined [3], what hidden means must be confirmed before mining hidden sequences. In this section, we take advantage of temporal logic to express properties of the desired behavior sequences.

Temporal logic [4] is basically an extension of traditional propositional logic with operators that refer to the behavior of systems over time. They provide a very intuitive but mathematically precise notation for expressing properties about the relationship between distinct states. The underlying nature of time in temporal logics can be either linear or branching. In the linear view, it is Linear Temporal Logic (LTL), while Computation Tree Logic (CTL) is logic that is based on a branching-time view. At this stage, we mainly focus on LTL, afterwards CTL can be considered for extension. In our consideration, all the processes related to social activities, as well as any distributed system, can be divided into six basic patterns as follows.

- *Tracing*: Different actions with sequential order, i.e. $\{a_1, a_2, \ldots, a_n\}$.
- *Consequence*: Different actions have causalities in occurrence, i.e. $\{a_i \rightarrow a_j\}$.
- *Synchronization*: Actions occur at the same time, i.e. $\{a_1 \leftrightarrow, \ldots, \leftrightarrow a_n\}$.
- *Combination*: Different actions occur in concurrency, i.e. $\{a_1 \| a_2 \|, \ldots, \| a_n\}$.
- *Exclusion*: Different actions occur mutually exclusively, i.e. $\{a_1 \oplus, \ldots, \oplus a_n\}$.
- *Precedence*: Some actions have required precedence, i.e. $\{a_i \Rightarrow a_j\}$.

Note that the *Consequence* means that the occurrence of action a_i will lead to action a_j, while the *Precedence* represents that the occurrence of action a_j must require the happening of action a_i. The above items are the six fundamental patterns in social activities, while in application quantifiers and temporal modalities are necessary to express a variety of properties. Specifically, the involved quantifiers are an existential quantifier $\exists$ and a universal quantifier $\forall$, and the temporal modalities are basic operators such as next $\bigcirc$, until $\bigcup$, eventually $\diamond$ and always $\Box$.

The sequential or parallel combinations of the above basic patterns, quantifiers and temporal modalities have a great power to express diverse properties. The sequential combination, intuitively, describes that the actions occur in a successive order for a property. The parallel combination, however, represents that all the actions involved in a property take place simultaneously. Symbols $\times$ and $\otimes$ are used to denote sequential and parallel combinations, respectively. In addition, nested combination is of great necessity to reveal the inner and outer relationships of actions. For extension, we could consider the fuzzy or probabilistic combinations of those relationships.

2.4 Behavior Analysis

In order to analyze the expressiveness, robustness and stability of our behavior model, we take great advantage of model checking to test the behavior model with the given properties. In this section, we interpret how to use model checking appropriately in behavior analysis.

Model checking [4] is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for that model. Specifically, two essential prerequisites for model checking are a system under consideration and required properties. On having them at hand, the

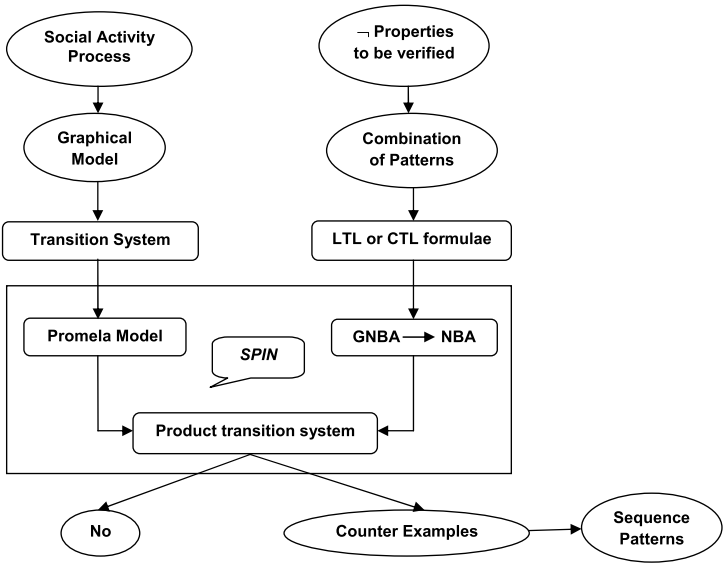

Fig. 2.4 The steps performed in SAPMAS

model checker examines all relevant system states to check whether they satisfy the desired property. If a state is encountered that violates the property, the model checker provides a counterexample that indicates how the model could reach the undesired state. Stated in details, the system will be represented or transformed as a transition system (TS), which are basically direct graphs where nodes denote states and edges model transitions (state changes), to describe the behavior of systems. The requirement will be formalized as a property specification in the form of temporal logic formula, then the negation of temporal-formula property is accepted by a Generalized Nondeterministic Büchi Automaton (GNBA), afterwards transformation will be conducted to generate a Nondeterministic Büchi Automaton (NBA) to replace the GNBA. When a TS and a NBA are ready, a persistence checking by nested depth-first search is led to verify whether the property holds for this system, then relevant results will be obtained: either "Satisfied" or "No" with a counter example.

Here, we mainly take advantage of model checking to analyze behaviors. At this stage, we only give a general framework of the whole process, more details will be discussed for our future work. We have introduced the behavior model and behavior properties, which will be mapped to a transition system (TS) and temporal logic formulae (TLF), respectively. Below, Fig. 2.4 illustrates the steps performed in Social Activity Process Modeling and Analysis System (SAPMAS).

Specifically, at first, a social activity process and the properties that are the features of the desired patterns must be given. However, the negative forms of the properties are considered rather than the original ones, since the counter examples obtained against ¬properties are just the desired activity sequences in accor-

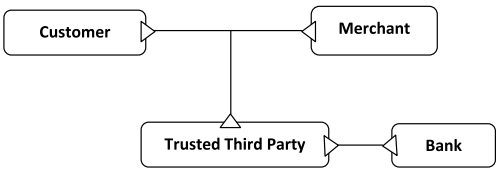

Fig. 2.5 The graphical actor sub-model of online shopping

dance with those properties. Then the particular patterns can be extracted based on the attainable counter examples. So after getting the social activity process and ¬properties ready, we could use the symbols proposed above to transform them into a graphical model and combinations of patterns, respectively. In the next stage, TS and TLF are further obtained. Subsequently, TS is then translated to Promela using the approach discussed in [13], note that the language Promela, which is short for "process metalanguage", is the input language for the prominent model checker SPIN [12]. Simultaneously, a GNBA can be constructed to accept the negative forms of LTL or CTL formulae, and then converted to NBA. Afterwards, a product of TS is made up from the combination of a Promela Model and a NBA. These three steps are achieved within a famous model checker SPIN. Thus, two input elements of SPIN are corresponding TS and TLF, then accordingly, the outputs of SPIN are two alternative answers, one is "no" which means there is no activity or action sequence possessing the desired properties, while in contrast, counter examples will be given out in accordance with the obtainment of the targeted sequence patterns.

2.5 Case Study

In this section, we use the increasingly popular e-Business process, i.e., online shopping process [1, 5] to illustrate the whole procedure of the system SAPMAS.

For *actor* sub-model, messages are communicated among a customer, a merchant and a trusted third party (TTP), along with the bank. Figure 2.5 depicts this typical actor sub-model, showing the parties involved in the online shopping process. There are four actors in total, i.e., a customer, a merchant, a TTP and the bank. It includes triple communication-point relation among the customer, the merchant and the TTP. The legend has been given in Fig. 2.1.

With regard to *action* sub-model, for the action "Send product", *subject* is the merchant, *object* is the customer, this action occurs at the *time* when the merchant has accepted the purchase order (PO) and the *place* is online, the *status* is in the main stage of online shopping process, it has the *constraint* of the acceptance of PO. This action has the *belief* that the customer will buy the product finally, and aims that the product will not be corrupted during transit. To achieve this *goal*, the merchant draws up a *plan* to send the copy of the encrypted product together with a signed cryptographic checksum. Intuitively, the *operation* is just to send the product, and this operation leads to the *effect* of requesting the decrypting key from the customer. Moreover, for *environment* sub-model, the *context* is that both the product ID has

been validated by the customer and the PO has been accepted by the merchant, while the *associate* is all the operations with direct link to the action "Send product", for instance, Accept PO, Receive product and Forward PO.

In the sequel, we will show how the transaction process takes place in a graphical model for the *relationship* sub-model. The whole procedure can be explained as follows: First, the customer browses the product catalog online located at TTP and chooses a favorable product. After that, the customer downloads the encrypted product together with the product identifier. Then if the identifier of the encrypted product file corresponds to the identifier in the product identifier file exactly, the transaction proceeds, otherwise advice is sent to the TTP and the customer waits for correct encrypted product ID. Subsequently, the customer prepares a cryptographic checksum and PO, which are sent to the merchant. Once receiving the PO, the merchant examines its contents. If the merchant is satisfied with the PO, the merchant digitally signs the cryptographic checksum of the endorsed PO and forwards to the TTP as well as a single use decrypting key for the product. Meanwhile, the merchant then sends a copy of the encrypted product to the customer, together with a signed cryptographic checksum. Next, the customer validates whether the first and second copies of the product are identical so as to confirm the ordered product. Afterwards, the customer forwards to the TTP the PO and a signed payment token, together with its cryptographic checksum to ask for decrypting key from the TTP. To verify the transaction, the TTP first compares the digest included in the PO from the customer with the digest of that from the merchant. If the two do not match, the TTP aborts the transaction. If they do match, the TTP proceeds by validating the payment token with the customer's financial institution. If the token is not validated, the TTP aborts the transaction and advises the merchant simultaneously. Otherwise, the TTP sends the decrypting key and the payment token, both digitally signed with TTP private key, to the customer and the merchant, respectively. Finally, after a somewhat complex process, the customer will receive the decrypted product if everything is all right, or else the transaction will be aborted by the TTP.

In stage-based Fig. 2.6, we divide the whole graphical model into three parts, namely *Preliminary Stage*, *Main Stage* and *Final Stage*. The *Preliminary Stage* is the phase for the customer to choose his or her favorable product, while the critical activities and communications happen in the *Main Stage*, then the TTP makes the last validation to determine the transaction in the *Final Stage*. This figure reveals the distinct phases of the whole procedure.

In the actor-based one in Fig. 2.7, the whole graph model has been divided into four sections according to the four actors involved. As mentioned before, the participating actors are the *Customer*, the *Merchant*, the *TTP* and the *Bank*. Note that all the actions involved in the communications of two actors have been cut into two sub-actions with symmetrical shapes. It shows that this kind of action only can happen with all of its related sub-actions enabled simultaneously. This figure reveals the different roles of distinct actors in a certain process or a series of activities.

Now, we have got the online shopping process visualized to a graphical model as shown in Figs. 2.6 and 2.7. Here, we will analyze the behavior properties. For the

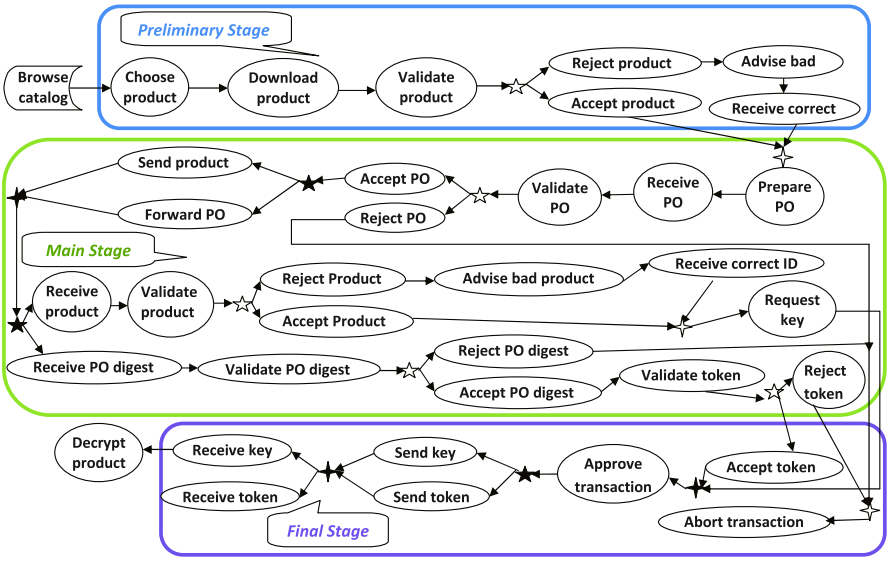

Fig. 2.6 The graphical action sub-model of online shopping based on stages

simple property stated as "*the actions choose product, download encrypted product and validate product ID must happen one after another in order*" can be written as $\{CP, \bigcirc DC, \bigcirc VP\}$, where the elements are the abbreviated forms of the corresponding actions, they are the so-called "atomic proposition" in logic language, and the same denotation applies for the following representations. Further, for the more complicate case that "*after accepting the PO, the merchant will send product to the customer and forward PO to the TTP*", accordingly, can be represented as $\{AP\} \times \bigcirc\{SP \| FPO\}$. Moreover, the property described as "*customer's rejecting product and accepting product cannot happen simultaneously while the former one leads to advise bad ID and the latter one goes directly to the stage of requesting key from the TTP*", similarly, can be depicted as $\{\{RP \rightarrow \bigcirc AID\} \oplus \{AP, \bigcirc RK\}\}$. Besides, formula $\{\{\forall RWOC\} \rightarrow \{\bigcirc AT\}\}$ means that "*all the rejected actions without proper correction will lead to undesirable transaction abortion*". Note that all of the above expressions of properties are validate for the graphical model. But those following properties presented as $\{ST, AP\}$ and $\{\{\forall AA\} \rightarrow \{\Diamond DP\}\}$, which mean "*accepting product after sending token*" and "*all the accepted actions will lead to receiving decrypted product eventually in the end*" respectively, will not hold. Note that expressions *RWOC* and *AA* denote the sets of corresponding atomic propositions. All of the mentioned examples are rather simple instances just to illustrate how to express properties in logic language.

For behavior analysis, two properties chosen are "*customer's rejecting product and accepting product cannot happen simultaneously while the former one leads to advise bad ID and the latter one goes directly to the stage of requesting key from the TTP*" and "*all the accepted actions will lead to receiving decrypted product eventually in the end*", represented as $\{\{RP \rightarrow \bigcirc AID\} \oplus \{AP, \bigcirc RK\}\}$ and

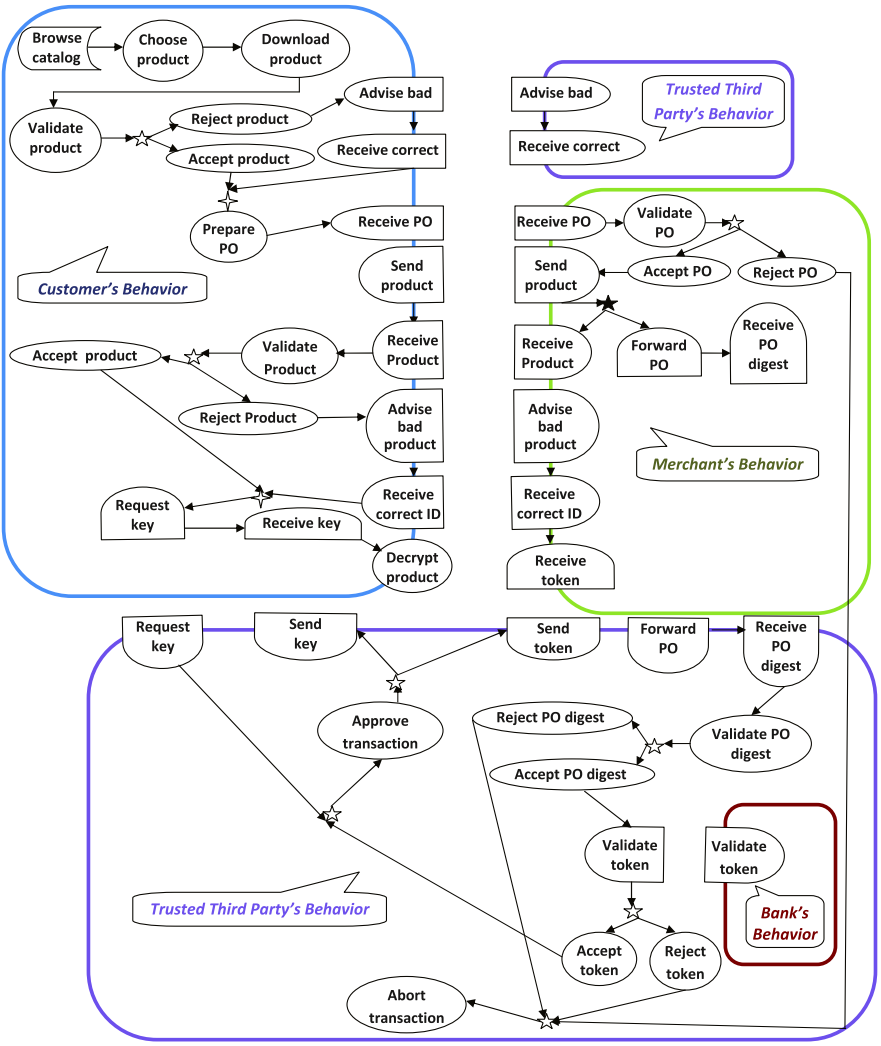

Fig. 2.7 The graphical action sub-model of online shopping based on stages

$\{\{\forall AA\} \rightarrow \{\Diamond DP\}\}$, respectively. After that, accurate transition system semantics of the graphical model and transformation rules are given to form corresponding TS and TLF. Then by using model checker SPIN, we could easily get the result that all the activity sequences following the online shopping process are the desired sequences for the former property, while for the latter one, the answer is no as there exists no activity sequence satisfying this property. Although this is rather a simple and naive example for analysis, it has shown the basic procedures of our modeling and analyzing system SAPMAS.

2.6 Conclusion

In this paper, we build a general framework, i.e., Social Activity Process Modeling and Analysis System (SAPMAS), for modeling and analyzing social activities. Different from existing behavior representation systems, SAPMAS utilizes the current advanced technique model checking, which is based on solid mathematical and computational backgrounds. It provides a graphical model to capture behavioral elements and properties within a social activity process, and summarizes the combinations of patterns to categorize the properties to be verified. The graphical model and combinational patterns are further transformed to a transition system and the logic form respectively to verify and refine social behavior models. SAPMAS outputs the desired activity sequence patterns after verification by SPIN. We exemplify the success use of SAPMAS in modeling online shopping process for our case study.

As a new research topic in behavior informatics, SAPMAS consists of many open issues that are worthy of systematic investigation as well as case studies. These issues include the following aspects:

- Combine quantitative and qualitative properties for advanced mining.
- Construct accurate transition system semantics for the graphical model.
- Establish transformation rules between combined formulae and TLF.
- Adapt the model checker SPIN to improve the functionality and applicability.
- Build extraction rules from counter examples to desired sequence patterns.
- Extend existing transition system to probabilistic or fuzzy transition system.
- Consider state explosion problems existing in model checking techniques.

These are all the unresolved and challenging issues for this new framework and system, and they are also our research issues in the future work for completing and enriching our newly proposed system SAPMAS with real world applications.

References

1. Bonnie, B.A., James, V.H., Paul, B.L., Scott, L.S.: Model checking for design and assurance of e-business processes. Decis. Support Syst. **39**, 333–344 (2004)
2. Brachman, R.J., Levesque, H.J.: Knowledge Representation and Reasoning. Elsevier, Amsterdam (2004)
3. Cao, L.: In-depth behavior understanding and use: the behavior informatics approach. Inf. Sci. **180**, 3067–3085 (2010)
4. Christel, B., Joost, P.K.: Principles of Model Checking. MIT Press, Cambridge (2008)
5. Chung, S.Y., Park, C.: Online shopping behavior model: a literature review and proposed model. In: Proceedings of the 11th International Conference on Advanced Communication Technology, pp. 2276–2282 (2009)
6. Edelkamp, S., Kissmann, P.: Optimal symbolic planning with action costs and preferences. In: Proceedings of IJCAI-09, pp. 1690–1695 (2009)
7. Eertink, H., Janssen, W.P.M., Oude Luttighuis, P.H.W.M., Teeuw, W., Vissers, C.A.: A business process design language. In: Proceedings World Congress on Formal Methods, Toulouse. Springer LNCS (1999)
8. Fast, A., Friedland, L., Maier, M., Taylor, B., Jensen, D., Goldberg, H., Komoroske, J.: Relational data preprocessing techniques for improved securities fraud detection. In: KDD (2007)

9. Gabaldon, A.: Activity recognition with intended actions. In: Proceedings of IJCAI-09, pp. 1696–1701 (2009)
10. Gu, Y., Soutchanski, M.: Decidable reasoning in a modified situation calculus. In: Proceedings of IJCAI-07, pp. 1891–1897 (2007)
11. Havelund, K., Lowry, M., Penix, J.: Formal analysis of a space craft controller using spin. In: Holzman, G., Najm, E., Serhrouchni, A. (eds.) Proceedings of the 4th International SPIN Workshop, pp. 147–167 (1998)
12. Holzmann, G.J.: The Spin Model Checker: Primer and Reference Manual. Lucent Technologies (2004)
13. Janssen, W., Mateescu, R., Mauw, S., Springintveld, J.: Verifying business processes using SPIN. In: Holzman, G., Najm, E., Serhrouchni, A. (eds.) Proceedings of the 4th International SPIN Workshop, pp. 21–36 (1998)
14. Kars, P.: The application of Promela and Spin in the BOS project. In: Proceedings of Second Spin Workshop (1996)
15. Koenig, S., Keskinocak, P., Tovey, C.: Progress on agent coordination with cooperative auctions. In: Proceedings of AAAI-10, pp. 1713–1717 (2010)
16. Pierce, W.D., Cheney, C.D.: Behavior Analysis and Learning. Lawrence Erlbaum Associates, 3rd edn. (2004)
17. Sardina, S., Patrizi, F., de Giacomo, G.: Behavior composition in the presence of failure. In: Proceedings of KR-08, pp. 640–650 (2008)
18. Serrano, J.M., Saugar, S.: An architectural perspective on multiagent societies. In: Proceedings of AOSE-10 at AAMAS-10, pp. 85–90 (2010)
19. Subramanian, K.: Task space behavior learning for humanoid robots using Gaussian mixture models. In: Proceedings of AAAI-10, pp. 1961 (2010)
20. Wang, F.Y., Carley, K.M., Zeng, D., Mao, W.J.: Social computing: from social informatics to social intelligence. IEEE Intell. Syst. **22**(2), 79–83 (2007)
21. Weiss, G., Hirsh, H.: Learning to predict rare events in event sequences. In: KDD-98, pp. 359–363 (1998)
22. Wooldridge, M.: Reasoning About Rational Agents. MIT Press, Cambridge (2000)

http://www.springer.com/978-1-4471-2968-4

Behavior Computing
Modeling, Analysis, Mining and Decision
Cao, L.; Yu, P.S. (Eds.)
2012, XVI, 376 p., Hardcover
ISBN: 978-1-4471-2968-4