Simulation in Produktion und Logistik
Entscheidungsunterstützung von der Planung bis zur Steuerung
Wilhelm Dangelmaier, Christoph Laroque & Alexander Klaas (Hrsg.)
Paderborn, HNI-Verlagsschriftenreihe 2013

A Petri net-based simulation model for the flexible modelling and analysis of building construction processes

Petri-Netz-basiertes Simulationsmodell zur flexiblen Modellierung und Analyse von Bauprozessen im Hochbau

Kais Samkari, Volkhard Franz, Universität Kassel, Kassel (Germany),
ksamkari@uni-kassel.de, vfranz@uni-kassel.de

Abstract: This paper suggests the combination of a component-type-based modelling approach with a Petri net modelling approach to analyse and simulate the shell works of a building construction project. This approach preserves the flexibility in modelling the construction processes at the level of a building component type. In addition, it benefits from the Petri net-based analysis methods to examine the static and dynamic behaviour of the system. After describing the models, the paper maps the process model onto the Petri net model. Then, a method for generating a Petri net that represents the project network from the process data of the project is described in detail. Finally, an application to a practical example is examined and discussed.

1 Introduction and background

The use of a simulation approach in building construction management concerns the efficient analyses of construction processes before and during construction. Despite significant advances in the application of simulation in the field of construction planning, an attempt to find an economical execution plan of a large-scale building construction project using simulation is still not an inexpensive endeavour. In order to plan using simulation there are two aspects of the simulation approach that should not be underestimated. On the one hand, the approach should provide a high flexibility to model and change the processes of the construction project being investigated. On the other hand, it should examine in detail the static and dynamic behaviour of the modelled construction processes.

Several studies have revealed that a component-type-based process modelling approach can achieve a high level of flexibility in modelling construction processes, as it reduces the effort of specifying the technological and spatial dependency relationships among construction activities (Aalami and Fischer 1998, Huhnt and

Enge 2006, Kugler 2012). One question that needs to be asked, however, is how the model's analysis properties such as *reachability* and *liveness* are checked.

Several studies investigating the use of Petri nets to analyse and simulate the dynamic behaviour of complex construction processes have been carried out (Franz 1989, van der Aalst 1996, Sawhney 1997, to name a few). These studies have demonstrated that the use of a Petri net is very reasonable, since it can be effectively used in modelling the structure and behaviour of complex stochastic systems (such as a construction project) at an abstract level. Further, Petri nets have a firm mathematical foundation that has resulted in an abundance of analytical methods that can be used to analyse many properties and problems associated with concurrent systems (Murata 1989). Modelling difficulties arise, however, when an attempt is made to manipulate the structure of a Petri net by a non-specialist.

This paper will focus on a modelling approach that combines a component-type-based process model and a Petri net model. While the component-type-based process model is the interface to the user, the Petri net model is the interface to the simulation environment. Section 2 describes the process model and the Petri net model, and section 3 introduces the mapping concept. Section 4 describes a method to generate the Petri net automatically. Section 5 examines the results of applying the method to a building construction project. The paper ends with a conclusion about the advantages of the proposed approach.

2 The process model and the Petri net model

2.1 The process model

This approach utilizes the component-type-based process model adopted from the CAD-integrated Simulation Modelling tool (CiSmo) (Kugler 2012). The CiSmo process model is inspired by the construction method model templates (CMMT) proposed by Aalami and Fischer (1998) and the process modelling technique addressed by Huhnt and Enge (2006).

The process model is divided into three main levels: construction activity, production process and construction method.

2.1.1 Construction activity

The process model consists mainly of a set of construction activities, each identified by a building component type, e.g., *foundation*, *column* and *beam*, and a driving resource type, e.g., *steel fitter* and *concrete pump*. While the building component type specifies the activities according to a set of CAD-object types, the driving resource type represents the resource type, the duration of which determines the duration of the activity.

The following attributes can be defined on the level of a construction activity: the production rate of the driving resource (e.g., in min/m2), the rate of the required materials (e.g., in m3/m2), and the needed construction machines (as a constant) and equipment (e.g., in unit/m3). It is important to note that the production rates of the driving resource can be defined as a deterministic or stochastic function of time.

2.1.2 Production process

As the number of activities in any given case is relatively large, it is necessary to group them into production processes according to construction trade and building component type, e.g., *concrete works of foundation*. An execution result and a set of technological prerequisites can be defined on the level of a production process.

An execution result is used to represent the state of the corresponding building component type after the process is executed. The execution result of a process can be either newly defined, or selected from a set of states belonging to the building component type attached to the process.

A technological prerequisite is a relation between two processes, namely a dependent and an independent process. The process that owns the prerequisite is the dependent process. A prerequisite is created first by relating a process either to the execution result of another construction process or to a newly defined state of a specific building component type, and then by specifying the type of the dependency relationship, i.e., *start – start*, *start – finish*, *finish – start* and *finish – finish*. The dependency relationships can be modified by time lags. When the new state of a building component type is defined, it is added to the set of states that belongs to that building component type. Thus, the newly entered value becomes a possible value while determining the execution result of the other processes. This enables the user to describe the dependency relationships among the production processes precisely.

Additionally, prerequisites and processes themselves have workflow instructions. By using the spatial terms: *this*, *previous*, *next* and *all*, the user can determine the spatial order of the process execution relative to the floors and working sections of the building.

2.1.3 Construction method

Since several production processes can handle different stages of the construction works of the same building component type, those processes can be considered together as a construction method.

In conclusion, what is interesting in the described modelling approach is that it does not require that the processes and their activities be defined in the system in their order of execution. Conversely, the user has a degree of flexibility in establishing a concept of the project execution by means of building component types in a very active way. Also through this approach, the construction knowledge of the finished projects could be used as an additional source in setting up new projects. Thus, structured data could partially replace any lack of experience and knowledge in the new projects.

2.2 The Petri net model

The standard model of a Petri net (PN) is defined by the tuple (P, T, W, M), where P: is a finite set of places, T: is a finite set of transitions, W: is a finite set of arcs and M: is a finite set of tokens representing the marking of the places.

To incorporate the domain-related semantics of the building construction field into the standard model of the PN, the transitions are annotated with the stereotype *Transition-ST*. The tags in the stereotype correspond to the attributes of the product

and process model elements, e.g., production rate and building component type. This ensures that the required data of the building product and the construction processes are not lost during the model transformation. Additional stereotypes such as *Split-ST* and *Sync-ST* will also be introduced later, which allow a preservation of the defined execution semantic of the processes.

A fundamental aspect of the building construction processes is concurrency that can easily be expressed by a conflict-free structure of a PN (also known as a marked graph) (Murata 1989). Each place in such a structure has exactly one incoming arc and one outgoing arc. Thus, the execution of a marked graph is considered deterministic, since firing any transition does not disable any other transition.

3 The mapping concept

A construction activity to be executed in a particular spatial location (i.e., specific floor and working section) of the building is represented by a transition, while the data of the activity are represented by values assigned to the attributes of the transition's stereotype. Technological prerequisites and workflow instructions in a particular spatial location are represented by a place with an incoming arc and outgoing arc that bind the appropriate transitions together.

In the proposed approach, the resource sharing among the construction activities is not modelled into the structure of the PN. In shell construction, there are basic types of construction resources that nearly every construction activity requires by its execution, e.g., a *crane* and a *steel fitter*. Further, it is a fact that some shell works on a construction site develop through successive steps depending on the technological and spatial dependency relationships among them. Considering these two facts, modelling the resource sharing of the activities into the PN as resource places is considered inefficient, since it increases the complexity of the PN by means of the number of the arcs. Additionally, it unnecessarily increases the number of the checks for enabled transitions whenever a resource is freed, though most of the extra checks will not be satisfied most of the time.

The resource requirements of an activity are modelled into the guard function of the corresponding transitions. A guard function restricts the enabling of a transition to the availability of the required resources. It is the responsibility of the simulation engine to evaluate the guard functions and to determine the enabled transitions during the runtime of a simulation run. Meanwhile, the simulation engine distributes the available resources over the enabled transitions based on a set of distribution strategies.

The timing concept is included into the model by associating a firing delay with each token as it is consumed by a transition. The firing delay specifies the time that a token has to reside in a transition, before the token is released again. This delay depends on the production rate, the quantity of the building component in the spatial location of the transition and the number of the assigned units of the driving resource.

4 Generating the Petri net

Generating the project Petri net consists of spreading the production processes and their construction activities over the floors and working sections of the building. The generation of the PN requires five steps, each of which is explained in more detail in subsections 4.1 – 4.5. After an empty PN of the project is created, a sub PN for each production process is generated and added to the project PN. Then, the generated sub PNs are associated with each other, or with the project start sub PN. Next, the net is simplified, so that dead associations and trivial transitions are eliminated from the PN. Finally, the end transitions of the processes, which are not predecessors of other transitions, are associated with the project end transition.

4.1 Step 1, create a sub PN for the project start and end

The PN created for the project consists of two dummy sub PNs representing the start of the project and the end of the project. The transitions of the sub PNs are assigned the *Start-Project-ST* and *End-Project-ST* stereotype, respectively. The initial marking of a simulation run consists of a token in the project start place, while the simulation run ends with a token residing in the project end place.

4.2 Step 2, create a sub PN for each production process

In order to spread the processes and their construction activities over the floors and working sections of the building, a set of transitions corresponding to the first activity in the process is created first. The following example describes this step in more detail. The workflow instructions of the process determine how many transitions have to be created (Fig. 1). Each transition receives the *Transition-ST* stereotype, a reference to the activity and a spatial location of execution. Next, the spatial terms of the workflow instructions that reflect the execution preference among the transitions are interpreted relative to the floors and sections of the building. Accordingly, the transitions are sorted and the sequences among them are created (step 2 and 3 in Fig. 2). Before the transitions corresponding to the rest of the activities of the process are created and connected to each other, a simplification step takes place. This step evaluates, for each transition, whether a quantity of the building component type exists in the transition's spatial location of execution. If the quantity does not exist, the transition is eliminated. Finally, the start and end transitions of the process are marked for a further purpose in the following steps.

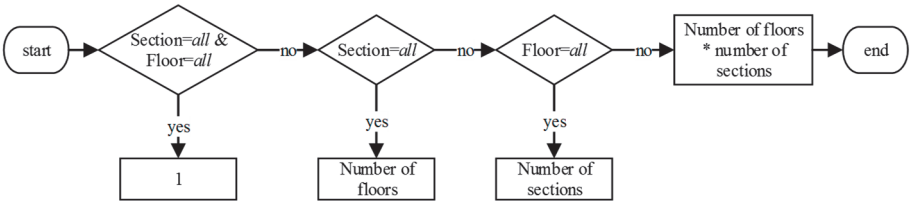

Figure 1: *A workflow diagram showing how to determine the required number of transitions according to the workflow instructions of the process itself*

Figure 2 shows an example in which a production process is spread over two floors and two working sections. The spatial terms of its workflow instructions are set to: *this* floor, *next* section. On the one hand, this means that there is no workflow preference at the level of the floors, and thus the execution can start on each floor at the same time. On the other hand, these instructions mean that the execution can start in a section of a floor after the execution in a successor section of the same floor is started. Therefore, only the transitions in each floor need to be bound together using a *start – start* dependency relationship. This is done by splitting the transition of the second section of each floor into two transitions. Each newly created transition, namely 1,2' and 2,2', receives the *Split-ST* stereotype. While the new transition gets the technological prerequisites and the resource requirements from its original transition, the original transition preserves the production rate and the quantity of the building component in its spatial location. Thus, the firing delay of the original transition's tokens could still be calculated.

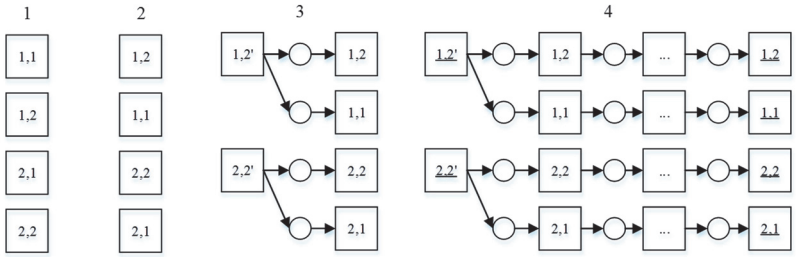

Figure 2: *Spreading a production process over two floors and two working sections. The workflow instructions of the process are set to: this floor, next working section. The label of a transition indicates the number of the floor and the number of the working section*

4.3 Step 3, associate the sub PNs with each other

This step concerns associating the generated sub PNs with each other, or with the project start sub PN according to the technological prerequisites of each production process. The transitions of the dependent process are called dependent transitions and similarly, the transitions of the independent process are called independent transitions. Consequently, this step associates each dependent transition in the dependent process with an appropriate independent transition in the independent process of a prerequisite.

When a process has no prerequisites, then its start transitions are associated with the project start transition using a *finish – start* dependency relationship. Otherwise, the algorithm enumerates the technological prerequisites and retrieves the independent process of each. The transitions of the dependent process can be either the start transitions (in the case of a *start – start* or a *finish – start* dependency relationship) or the end transitions of the process (in the case of a *start – finish* or a *finish – finish* dependency relationship). Next, for each dependent transition, the spatial location of its corresponding independent transition is determined. This depends on the spatial terms of the workflow instructions of the prerequisite in hand. After that, the algorithm distinguishes between four cases in order to retrieve the independent

transition. If the independent process is not found and the quantity of the building component in the spatial location is zero, then the PN is not affected. However, if the quantity is not zero, the result is the creation of a new transition that is assigned a *Hangover-ST* stereotype. Further, when a spatial location is not available, the project start transition (in the case of a *start – start* or a *start – finish* dependency relationship) or the project end transition (in the case of a *finish – start* or a *finish – finish* dependency relationship) is considered the independent transition.

The general case is to find and return the independent transition from the list of transitions of the independent process. While searching for the independent transition in this list, a transition with the specified spatial location might not be found. Consider the case when process B must wait until process A is executed in all the sections of the same floor, i.e., *this* floor, *all* sections. At the same time, the execution of process A in the sections of each floor is configured to be concurrent, i.e., *this* floor, *this* section. The independent transition in this case is a new transition that synchronizes the execution of process A in all the sections of each floor (Fig. 3, left side). The resulting transition receives a *Sync-ST* stereotype, a spatial location and references to the synchronized transitions. Consider also the opposite case when process B must wait until process A is executed only in the same section of the same floor, i.e., *this* floor, *this* section, while the execution of process A in the sections of each floor is configured to happen at the same time, i.e., *this* floor, *all* sections. In this case, the algorithm returns the transition that most closely matches the specified spatial location (Fig. 3, right side).

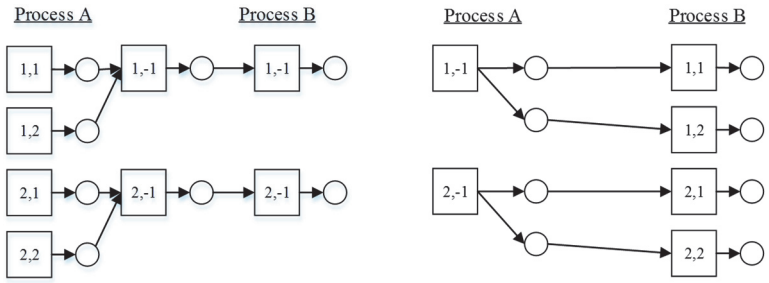

Figure 3: *A synchronization transition is created for each floor for case one (left), while, in case two (right) the most closely transitions are returned*

Finally, the dependent transition is associated with the retrieved independent transition according to the type of the dependency relations of the current prerequisite (Fig. 4). For example, a *start – finish* dependency relationship between transition A and transition B requires splitting each into two transitions, namely A' and B'. While transition A' receives a *Split-ST* stereotype, transition B' receives the *Fwd-Split-ST* stereotype, and is assigned only a reference to its original transition B. In addition, a place with one incoming arc and one outgoing arc is created. The place's incoming arc is added to the set of the post-arcs of transition A', and the place's outgoing arc is added to the set of the pre-arcs of newly created transition B'.

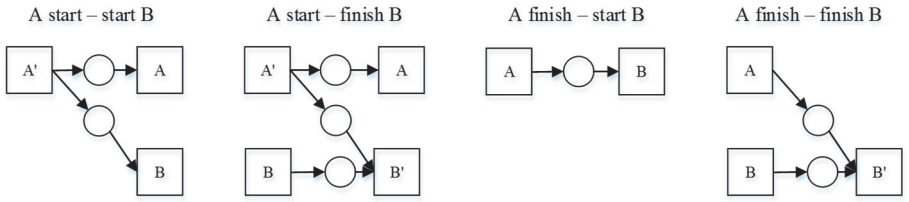

Figure 4: *Four Petri net models representing the standard dependency relationships*

The dependency relationships can be modified by time lags. This requires the creation of a transition that is assigned an *Interval-ST* stereotype and a firing delay corresponding to the interval given in the process data (Fig. 5).

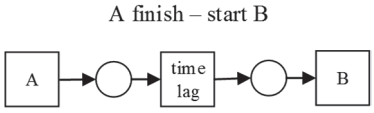

Figure 5: *Example of how to model a time lag of a dependency relationship, here a finish – start relationship*

4.4 Step 4, simplify the structure of the PN

The fourth step is to simplify the structure of the PN so that dead associations and trivial transitions are eliminated from the PN. By definition, a dead association is a place associating a source transition with a target transition reachable through a chain of places starting from the same source transition. In this case, the place and its incoming and outgoing arcs can be spared.

A trivial transition is a transition that either is assigned a *Split-ST* or a *Sync-ST* stereotype (but not a *Transition-ST*), and has only one outgoing arc; or is assigned a *Fwd-Split-ST* stereotype and has only one incoming arc. In the case of a transition with a *Split-ST* or a *Sync-ST* stereotype, the set of pre-arcs are moved and added to the set of pre-arcs of the following transition. Then, the transition itself and its following place are eliminated. In the case of a transition with a *Fwd-Split-ST* stereotype, the set of post-arcs are moved and added to the set of post-arcs of the previous transition. Then, the transition itself and its previous place are eliminated.

4.5 Step 5, associate sinks with the project end transition

In the last step, the end transitions of the processes are checked. Each end transition that does not have post-arcs to places connecting it to other transitions, i.e., a sink transition, is associated with the project end transition using a *finish – start* dependency relationship.

5 Discussing the results

The implementation of the proposed method is a part of the simulation environment MOSAICA (Samkari et al. 2012b). MOSAICA is loosely coupled from CiSmo system, which is integrated into the CAD-environment AutoCAD Architecture. The

CiSmo plugin automatically collects the product data of the building model of a construction project and saves them to a XML file. The process data are put into the system using CiSmo's GUIs and are saved to a MySQL database. MOSAICA allows read access to the product and process data of CiSmo using a specific programming module that is compatible with the data structure of CiSmo. Similarly, it is possible to develop modules that allow read/write access to a variety of data formats of product and process data of other building modelling environments and/or planning and management information systems. In addition, MOSAICA allows the visualization of a Petri net created for a project using an open source graph visualization software called *Graphviz*. The Petri net diagram allows a visual check for inconsistencies in the structure of the net. In addition, a verification and validation (V&V) method was developed and integrated into MOSAICA (Samkari and Franz 2012a). The V&V method checks, for example, whether a firing sequence exists from the project start transition to the project end transition and whether the execution of the net is deadlock-free. In addition, it checks whether all building component types are assigned a construction activity and will be constructed. Consequently, this results in an increased accuracy of the simulation results.

The implemented method is applied to a four-story nursing home. Each floor of the building is divided into four working sections. The building 3D-CAD-model contains building components of 24 different building component types. The component types are associated with 60 construction activities representing the shell works of the project. The source of the process data is the real execution plan of the project. The generated Petri net contains 599 transitions: 462 *Transition-ST* transitions, 118 *Split-ST* transitions, 14 *Interval-ST* transitions, 3 *Sync-ST* transitions, 1 *Start-Project-ST* transition and 1 *End-Project-ST* transition.

The application of the *liveness* property to the Petri net detected dependency relationships to undefined production processes as well as flow deadlocks resulting from errors in the specification of the workflow instructions of a dependency relationship of a process. Also, the application of the *reachability* property to the Petri net detected building component types that are not assigned to production processes.

This test case revealed some issues that make the application of the proposed modelling approach less practical than it could be. For example, the generated Petri net is visually difficult to read and understand since it contains a large number of nodes and arcs. Thus, a tool to specify which parts of the net to visualize is extremely conceivable. Also, the generated net is strongly dependent on MOSAICA framework which makes the net not directly executable in other Petri net simulators. An attempt to transform the generated net into the format of other Petri net simulators would support the portability of the proposed modelling approach.

In order to validate the approach, the results of the execution of the Petri net using MOSAICA are compared to the results of the execution of the same product and process data under the same experiment conditions in another simulation environment (Kugler 2012). The comparison shows the results to be identical in terms of the plan of the construction activities and their execution durations on the floors and in the working sections of the building.

6 Conclusion

This paper presents a reliable method to generate a marked graph Petri net from the product and process data of a construction project. By first generating a sub Petri net for each production process and then by associating the sub nets according to the technological and spatial dependency relationships, a project network describing the execution plan on the floors and in the working sections of the building is created. The focus of this paper is on an approach combining the component-type-based process model of CiSmo with a discrete time stochastic Petri net model. The approach ensures a high flexibility in modelling construction processes and provides a formal method in analysing the static and dynamic behaviour of the processes. Moreover, it relieves a non-expert user of the task of dealing with the principles of the Petri net theory. The application of the introduced method to a four-story nursing home confirmed the validity of the approach.

References

Aalami, F.; Fischer, M.: Joint product and process model elaboration based on construction method models. In: Björk, B.C.; Jädbeck, A. (Ed.): Proceedings of CIB W078 conference, The life-cycle of construction IT innovations – Technology transfer from research to practice, Stockholm (Sweden), 05.-08. June 1998.

Franz, V.: Planung und Steuerung komplexer Bauprozesse durch Simulation mit modifizierten höheren Petri-Netzen. Kassel, Gesamthochschule Kassel 1989.

Huhnt, W.; Enge, F.: Can algorithms support the specification of construction schedules?. Journal of Information Technology in Construction 11 (2006) Special Issue Process Modelling, Process Management and Collaboration, pp. 547-564.

Kugler, M.: CAD-integrierte Modellierung von agentenbasierten Simulationsmodellen für die Bauablaufsimulation im Hochbau. Kassel, kassel university press 2012.

Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77 (1989) 4, pp. 541-580.

Samkari, K.; Franz, V. (2012a) A Petri net based method for the early verification & validation of a simulation study in construction management. In: Laroque, C.; Himmelspach, J.; Pasupathy, R.; Rose, O.; Uhrmacher, A.M. (Ed.): Proceedings of the 2012 Winter Simulation Conference (WSC), Berlin (Germany), 09.-12. December 2012, pp. 344:1-344:2.

Samkari, K.; Kugler, M.; Kordi, B.; Franz, V. (2012b) Colored Petri-net and Multi-Agents: A Combination for a Time-efficient Evaluation of a Simulation Study in Construction Management. In: Issa, R.R.; Flood, I. (Ed.): Proceedings of the 2012 ASCE International Conference on Computing in Civil Engineering, Florida (USA), 17.-20. June 2012, S. 153-160.

Sawhney, A.: Petri Net Based Simulation of Construction Schedules. In: Andradóttir, S.; Healy, K.J.; Withers, D.H.; Nelson, B.L. (Ed.): Proceedings of the 1997 Winter Simulation Conference (WSC), Atlanta (USA), 07.-10. December 1997, pp. 1111-1118.

Van der Aalst, W.M.P.: Petri net based scheduling. Operations-Research-Spectrum 18 (1996) 4, pp. 219-229.