A Petri Net Table Model Applied to Classic and Agile Project Management

Maxi Weichenhain[1] and Wolfgang Fengler[1]

[1] Computer Architecture and Embedded Systems Group,
Technische Universität Ilmenau, Germany
maxi.weichenhain@gmx.de; ra-fgltr@tu-ilmenau.de

Abstract. The world of project management (PM) is diverse and extensive, as is the resulting tool selection on the market. Despite the wide variety of tools available, many PM activities are carried out using self-developed spreadsheets that have been tailored to meet the needs of the project. The authors have evaluated experiences from several completed projects and recognizes the need for a systematic model based on a formalised and tool-independent table structure. The goal of this work is to use a powerful Petri net (PN) model for mapping and simulation, taking advantage of the model's benefits. Thus, an effective and flexible methodology for PM can be realised. In order to use the concept of PN for this purpose, appropriated extensions and necessary interpretations from the area of high-level timed Petri nets must be used. Moreover, PM has to be brought to the same description level as PN. The concept of this proposed method is to enter all data, initial plan inputs and intermediate status updates into a defined table structure. A PN is then automatically generated in the background or updated to reflect the current status of the project. Simulations should be performed on the PN models, and their results should be put back into the table. The advantages of the PN concept can thus be fully exploited and become useful for a project manager, even if he or she does not have any PN knowledge. This article presents an overview of the proposed formalised methodology, its suitability to PM, and its benefits. The method itself can be applied to a wide range of projects. Focus is placed on how to transfer certain areas of PM into the introduced PN constructs and interfaces. Examples are used to illustrate how to transform PM activities and their related information into the selected PNs.

Keywords: Petri Net, Project Management, Atomicity, Level of Abstraction.

1 Introduction

Projects are deeply rooted in history. From an activity without extensive methods, instruments or specifications, a structured approach has grown over the past years. This approach is called project management (PM). The use of PM methods has reached a high degree of coverage, as evidenced by the wide variety of tools and utilities available on the market. Project modelling, a sub-task of PM, is a cornerstone for the successful implementation of a project.

The authors practical experiences are consistent with previous study results, which claim that spreadsheets are not only frequently used as a substitute for a PM tool but are also often used as an accompanying documentation and reporting tool in addition to customised or industry solutions. The reasons for spreadsheet use can be diverse. [1, 2]

The intention is not to replace conventional tools, but to support and formalize the use of spreadsheets, and to extend it by a number of simulation and evaluation possibilities. Using spreadsheets facility is generally done without a systematic method, and a new project often requires extensive revisions. Particularly in the case of agile project methods, which are increasingly being used in a wide range of project forms as well as resource planning activities, the complexity and unwieldiness of these tables increases dramatically.

This article briefly introduces the basic idea and structure of the PM-Petri-net (PN) model, including a differentiation from existing approaches in the literature. Based on selected areas from the PM, this PM-PN method is presented as an example and to clarify the principles of this method. The method's functionality is illustrated by an excerpt from a Petri net created with a selected tool.

2 The Project Management Petri Net Model

The type of project examined in this paper takes place in a dynamic environment that is subject to a wide range of uncertainties, especially with regard to resources and time. Revisions to the plan are constantly required but are often reduced to significant deviations, resulting in potential sources of error and quality losses in the PM. Another problem is the poor adaptability of agile methods and their resulting changes. [3]

2.1 Classification

In order to meet the requirements of today's agile PM, there exists a need for new methods of PM. This requires paying adequate attention to the dynamic and complex nature of today's projects. In addition to formal mathematical structure, PNs provide a broad, customisable, and extended range of functions well-suited to PM. Furthermore, they can be implemented alongside the project and scaled as the scope of the project changes. Fundamental adjustments and shifts in planning or resource allocation can be time-consuming, even without considering the limits of the selected tool's adaptability. Expanding a table, on the other hand, is easy, as is the basic idea of the use of modular structured constructs. [4, 5]

Therefore, the first step in creating a new method for PM is to systematise the use of tables. In the present contribution, previously defined table structures are used, which are described in Section 4.2. The second step involves transforming the tables into PNs. For this step it is prerequisite to write down the project planning and actual project accompanying statuses for the inverse transformation of simulation results to forecast further courses of the project. The basic concept that is applied will be described in Section 4.

2.2 Concept

The PM and the PN concept originate from quite different worlds. To model PM, its complex tasks have to be abstracted. To model PNs, a decomposition of the entire model into subsystems, including the interactions of their components, has to take place. A merge of these two concepts can only be affected if a common level of detailing and abstraction can be found. First of all, the requirements for mapping PM to a PN must be identified and a suitable common abstraction level must be defined. This level of abstraction will serve as the basis to define dimensions for the smallest described objects and their components. [7]

The objects of modelling are called the project objects with their operations and relationships to each other. To ensure a formal and consistent presentation, it is also necessary to classify objects' properties and derive their requirements. On this basis, a formal table structure and thus a method for the transfer and adaptation of PM can be defined. Since the proposed PM-PN model is a dynamic consideration, the cycle of this formal modelling must also be presented as it will represent the basic idea of the developed method.

In the presented approach, the so-called constructs are identified on the basis of project objects. These constructs map the requirements in the context of PM and communicate via interfaces and can be used in a modular and arbitrary manner. The starting situation is a mixed classic and agile PM approach. For the modular assembly of the identified constructs, the consideration of the predecessor-successor handling is decisive. Constructs have been defined to satisfy the needs, which is expressed in parallelism and several predecessor-successor relationships, as well as constructs with simple relationship requirements. This stated, these constructs can be combined for the classic and agile approach. For example, the mapping of sprints (more in Section 4.1), a technical term from agile PM, requires that they are not processed in parallel but sequentially.

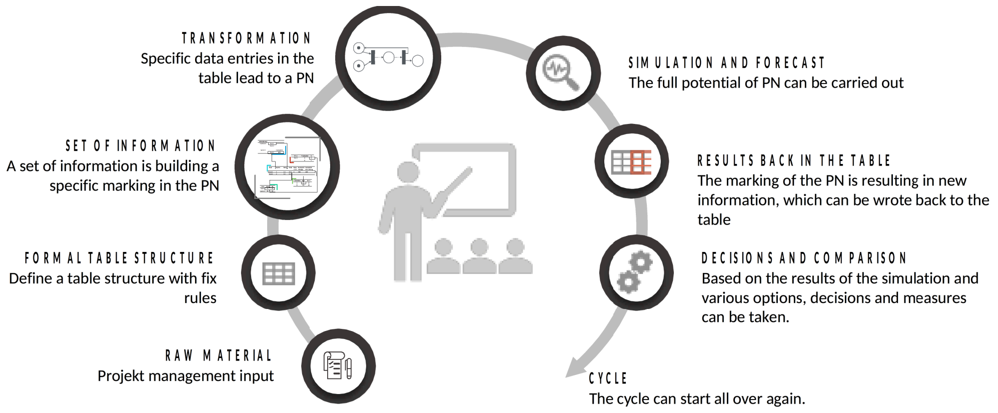

TRANSFORMATION
Specific data entries in the table lead to a PN
SIMULATION AND FORECAST
The full potential of PN can be carried out
SET OF INFORMATION
A set of information is building a specific marking in the PN
RESULTS BACK IN THE TABLE
The marking of the PN is resulting in new information, which can be wrote back to the table
FORMAL TABLE STRUCTURE
Define a table structure with fix rules
DECISIONS AND COMPARISON
Based on the results of the simulation and various options, decisions and measures can be taken.
RAW MATERIAL
Projekt management input
CYCLE
The cycle can start all over again.

Fig. 1. The cycle from data entry to PN to benefit from new capabilities

In order to close the cycle (demonstrated in figure 1) of data entry, which includes modelling by the use of PN and its simulation or forecasting possibilities, the results need to be written back to the table. Among other things, as an example, the correct runtime of a finished sprint and their assigned tasks can be checked. If all preconditions to complete a sprint are met, the marking is written back to the table by the PN and can be compared with the plan values.

2.3 Description of the Used Petri Net Model

The behaviour of the PN used in this work is defined such that a place is an event or a certain state in the environment of an activity and the transition of this activity (activity transition) itself are associated with a state transition. Where applies:

A transition fires if all pre- and post-conditions are met. This depends on the selected interpretation and extension of the PN and will be described later. The firing ability regarding the post-conditions is generally given. This is a consequence of the structure of the transformed PNs. When the preconditions are set and the transition fires, the assigned activity is started and delayed in accordance with the selected time model. The change of the marking of the pre-conditions takes place with the start of an activity, and the change of the marking of the post-conditions takes place with the end of that activity.

Post-conditions of transitions can also be prerequisites for further transitions. Transitions without an activity execution can also occur, for example to control the starting of tasks. As identified in various research [8, 9], variants of PNs are extended by additional concepts. In the context of this article, the extensions with high-level timed PNs, especially coloured PNs with structured tokens based on the predicate/ transition-PNs will be used, as well elements of hierarchical place subnets with timed and timeless transitions.

These 'structured tokens', as already defined for another use case [10], allow tokens to carry various kinds of information, such as machine conditions, properties of resources, or project related information and documents. The firing rule defined in the previous use case [10] allows suitable structured multiplicities of arcs to direct operation on the structures of tokens and logical links that control the execution of activities depending on the existence and content of the tokens. Structured tokens describe the existence and current data structure of events and states in the environment of the activity.

Fundamental investigations on the subject of the present contribution are also possible with the coloured PN tool Chromos [10] (for example, transformation and simulation), by representing all possible occurring assignments of the structured tokens with a set of logical colours. This tool was used for the presented results. For real applications, this method is not practical due to the number of coloured tokens needed and the number of assignments. It will then be necessary to have a special tailored PN tool. However, Chromos will be used to demonstrate the simulation and composition of these constructs, according to the chosen principle that inputs and results are represented in the defined table structure. The graphical user interface had a subordinate meaning.

A construct in the sense already used, represents a hierarchically subordinated place subnet in the used PN. A 'place subnet' in this context is a subnet in which arcs from the surrounding construct only end in places, and arcs to the surrounding network only begin at places. Restrictions on the inner structure of the subnet generally do not exist, but this is dependent on the construct. This allows a consistent linkage of constructs across the places defined as an interface through the mechanism of overlaying places.

3 Application of Petri Nets in the Field of Project Management

In the literature, PN approaches have often been used to tackle PM problems. They all share a similar origin. The focus has always been on the classic methods of project management. In this analysis, the trend of agile project management methods is included. Previous research has often listed following problems in the field of project management: growing scopes, uncertainties, requirements, and complexity. The importance of planning, monitoring, controlling, and real-time analysis in parallel to the project activities is growing. This creates the need for dynamic project planning and in particular for a dynamic creation of the project schedule and resource control. One research [11] took a look at conventional management tools like program evaluation and review technique (PERT) and critical path method (CPM), or even improved tools such as decision CPM (DCPM), graphical evaluation and review technique (GERT) and venture evaluation and review technique (VERT) and determined, that these methods cannot satisfactorily meet all requirements with regard to: "non-autoomatic rescheduling of activities, non-suitability to resolve conflicts arising from resource priorities, incapability of representing resource interdependencies, no provision of information to analyse reasons for the tardy progress of activities, and no help in the studies of partial allocation, mutual exclusivity and substitution of resources."[11]
The classic tools for PM no longer meet these increased demands, and there exists a need for a realistic method to model project. It has always been mentioned that the properties of PNs are useful for realistic and dynamic modelling (e. g. dynamic modelling, concurrent, non-deterministic, or formal-mathematical). Research on PM approaches that are based on PN concepts have existed since the 1980s. Until the year 2000, the aim to optimise the project time by optimising the use of resources or to optimise the progress management itself was a selected topic. A project going through several states and along this states attempts have often been made to supplement the historical methods using PNs, or in rare cases, to integrate them as a substitute for individual task areas. [12, 13]

In recent years, resource optimisation and resource management have been a frequent and diverse area of research. An attempt has been made to model, simulate, and optimise the resource allocation and the critical path with PNs. For example, the PN extensions such as the time Petri net (TPN), stochastic Petri net (SPN) and coloured Petri net (CPN) have been chosen as a remedy as formed by the transformation from another methodology, such as the PERT, the CPM, the work breakdown structure (WBS), and the business process model and notation (BPMN). Alternative paths and simulation have often been revisited as a PM enrichment. [12, 14, 15]

One work [13] was devoted to the suitability of PNs for use in the area of PM. Several authors [3] has studied a comprehensive framework for modelling, scheduling, and simulating projects with (1) a newly developed PN extension, (2) PN patterns, and (3) a graph search algorithm. As a starting point, he has transferred a BPMN to a PN in order to gain analysis and insights for the re-planning of the project. No formal approach was found to transfer PM into a table structure in order to use it for a PN and to provide a modular system. What is the aim of the approach presented here. [4]

4 Models, Transformations and Simulations

When modelling projects with PN concepts, the adaptation of state elements and events must be transferred from the PN to the PM. For this purpose, the term project object is introduced, and a closer look will be done in connection to discrete models.

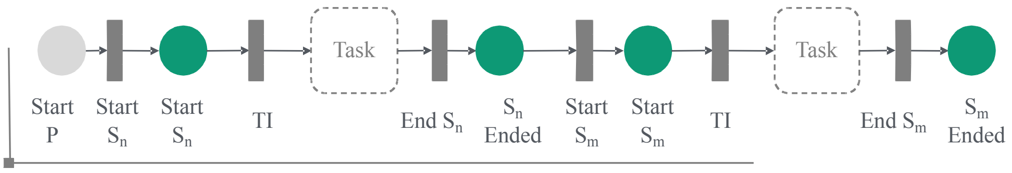
Based on this approach, the table structure is presented as an example in this section with the goal of clarifying the proposed concept. The following subsection begins by defining the idea behind the modular constructs. Afterwards the automatic generation on the variance of planned and actual data shall be considered as an example.

4.1 Common Level of Abstraction

The project objects are defined as activities, goals, resources (all kinds of resources including machines, employees, organisational units, and capabilities), time, and relationships. The project objects are related to one another. The activities either serve as an input, consumption, or output to other project objects. Activities are understood as processes that in turn can be combined to form activity chains. Each project object is characterised by its properties and their function behaviour as well as the operations involved in the interaction of their relationships with one another.

In the present case of the complex objects and their relationships as well as their defined interface to communicate with one another, a division into several components is recommended. The term 'construct' is introduced for this purpose, which is a cluster of PN elements (referred to in section 2.3 as a subnet). The formation of the constructs is based on the introduced project objects' time, resources, and activities and their predecessor-successor dependencies. The basic principle of a construct is based on object-oriented programming in order to generate the required PN with class descriptions. Each construct must provide an interface. This interface allows other constructs to access predefined data of the individual components or to retrieve results. [6]

With agile project planning methods, such as the framework of Scrum [16], the foundation relies on flexibility and continuous adaptation to feedback instead of detailed project planning. In the context of this article, a general explanation of agile project planning will be foregone and only the sprint will be discussed.



➢Simple concatenation - two interfaces are available per sprint: Place *Start Sprint* and place *Sprint Ended*
➢*Legend: Project (P), Sprint (S), Task Initiation (TI)*

Fig. 2. Concatenation of sprint constructs [6]

In figure 2 you find an example to concatenate the interfaces for the sprint construct. Sprints are defined as time-boxed periods, where all identified requirements are managed using a type of to-do list and are implemented incrementally. In other words, a sprint is a structuring of several individual tasks that are realised within a defined period of time. These tasks can run in parallel within their allocated sprints.

4.2 Table Structure

A great challenge in creating a PM-PN model lies in the selection and recovery of the required data. [17] From experience, spreadsheets maintained for this purpose are laden with formulas and complexity, which often leads to the inclusion of information that is not easily accessible or recognisable. As an example, a changed sprint composition has to be maintained in several tables and often also leads to structural changes for individual tables. The formalisation of the tables, which refers to the mechanism for filling in the tables and the data entry, simplifies the process of accommodating a changed sprint composition. Each piece of information has a clearly defined location in the table and is therefore limited to specific table areas. This structure is beneficial for making changes, since the changes are taken into account by the PN whenever accessing elements. As a result, only the information contained in the table is transferred to the PN. An optimised table form is used in which the table size is reduced and limited to purely data entry.

The representation of the project and its monitoring, analysis, and simulation does not take place through underlying formulas or logic. Instead it will be done through the use of PN constructs, PN elements, and their interactions.

The aim is to standardize the data and allow models to be created automatically. A prerequisite for this is that all data, deadlines and correlations must be included in the table. The structure of the table must be synchronized with the program for PN transfer. There are several possible approaches and structure options. In the model presented, a meaningful normalisation and no double data storage is aimed. The tables to guide the project planning are divided into several tables. The data of the duration, date format, quantity and the formation of IDs have to be defined and rules have to be set. The content-related link is realised by a uniform identifier (ID). The ID is subject to a scheme that allows information to be accessed when using structured tokens. For example, when considering the reusable resource ‘employee’, each activity (or partial information of the ID structure) is stored in the structured token and memorised by the construct of reusable resources for later tasks. This information out of the structured token could be used to determine whether the resource, that is, the employee, has already gained experience related to this kind of task. Another example, concerning defining rules for data entry, is the determination that the duration of the task must always be consistent with the time model. This constraint is enforced by allowing only multiple units of the chosen time format. [6] Table 1 represents only a section of the aspects highlighted before.

The method used to record the assignment of tasks to the respective sprints and to collect start and finish dates is demonstrated in the table above. The data acquisition must correspond to the requirement of a modularly expandable table format. This is necessary to automatically generate the PN and its underlying constructs, which also requires a standardised format for the data acquisition. All required data has to be recorded in the pre-formatted tables.

Table 1. Excerpt from a sample spreadsheet for maintaining the data of sprints

ID	Process	P ID	Sp	Duration D	Start D	End D	Duration Dev	Start Dev	End Dev
S _I_01	Excel as I	-	1	6	22.01	26.01	13	29.01	02.02
R_F_01	R Filter Function	S _I_01	1	14	22.01	26.01	29	29.01	02.02
R_D_01	R Layout	S _I_01	1	8	29.01	02.02	16	05.03	09.03
S_I_02	Connection to the S	-	2	16	23.02	02.03	15	04.03	15.03
TF_I_02	Data transfer TF	S_I_01, R_D_01	1	15	15.02	10.03	28	11.03	16.03

Legend: Design (D), Development (Dev), Target Figure (TF), Interface (I), Predecessor (P), Report (R), Sprint (Sp), System (S)

4.3 Execution Using the Example of the Petri Net Construct Sprint

The sprint construct takes all requirements into account (such as the mandate that sprints do not run in parallel) and contains tasks that needs to be processed. Figure 3 pictures these requirements in the PN for the first sprint of a project with *n* tasks assigned. In the illustrated, simplified example, all tasks run in parallel, which is not always the case. The interface to the time construct has been omitted for improved clarity, but it will be included in the exemplary illustration (Figure 4) of a single task.

The first sprint construct can start as soon as the project is started and there is a token at the place *Start Project* that marks the beginning of the sprint processing. For the coordination of the individual tasks, a *Coordinator* place is used to initiate each task. To do this, it is necessary for the coordinator to be provided with the number of separate tasks assigned to the sprint. A corresponding number of tokens are required in this place. This number of tokens will be provided to the coordinator by firing the transition *Start Sprint* and the corresponding multiplicity of the arc. The coordinator's only aim is to initiate all tasks. The order and interdependencies of the tasks are realised by the transition *Start Task* (see Fig. 4*)* and their relation to other tasks with the use of a test arc to the interfaces of the predecessor task (see Fig. 3). The test arc is necessary to enable multiple successors for a task. All transitions can fire to initiate the processing of their respective tasks. Again, to initiate correctly, each task must provide an interface to process this signal. This interface is realised for each task by the place *Initiate Task$_{\#No.}$* and is highlighted in light grey in Figure 3. Since the transitions for initiating the task are not inhibited by any pre- or post-conditions, the capacity of the *Initiate Task* places are set to 1 and the firing rule is based on maximum firing. This prevents that no task can start more than once and no other task is prevented from starting. To prevent this condition from being violated by other components of the developed PN model, the initialisation transitions cannot be part of the sprint's interface. After the coordinator has initiated the individual tasks, the processing of these tasks can begin in parallel or sequentially in accordance with the developed construct. The completion of a sprint must also be considered. The light grey places represent the interfaces for processing a task and can be found in Figure 4 as an interface per task. [6]

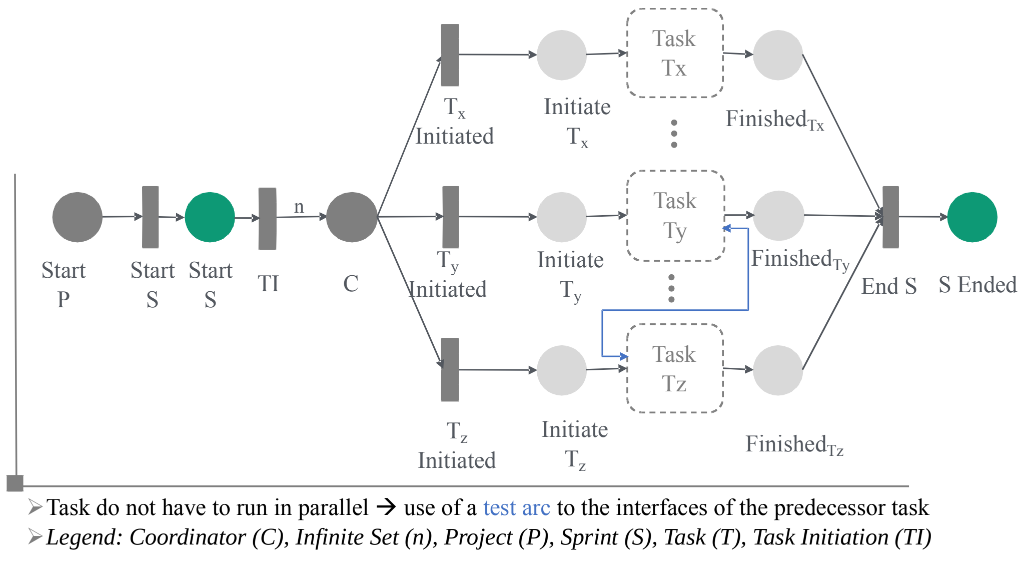

Task Tx
Tx Initiated
Initiate Tx
Finished$_{Tx}$
Task Ty
Start P
Start S
Start S
n
TI
C
Ty Initiated
Initiate Ty
Finished$_{Ty}$
End S
S Ended
Task Tz
Tz Initiated
Initiate Tz
Finished$_{Tz}$
➢Task do not have to run in parallel → use of a test arc to the interfaces of the predecessor task
➢Legend: Coordinator (C), Infinite Set (n), Project (P), Sprint (S), Task (T), Task Initiation (TI)

Fig. 3. Setup sprint construct [6]

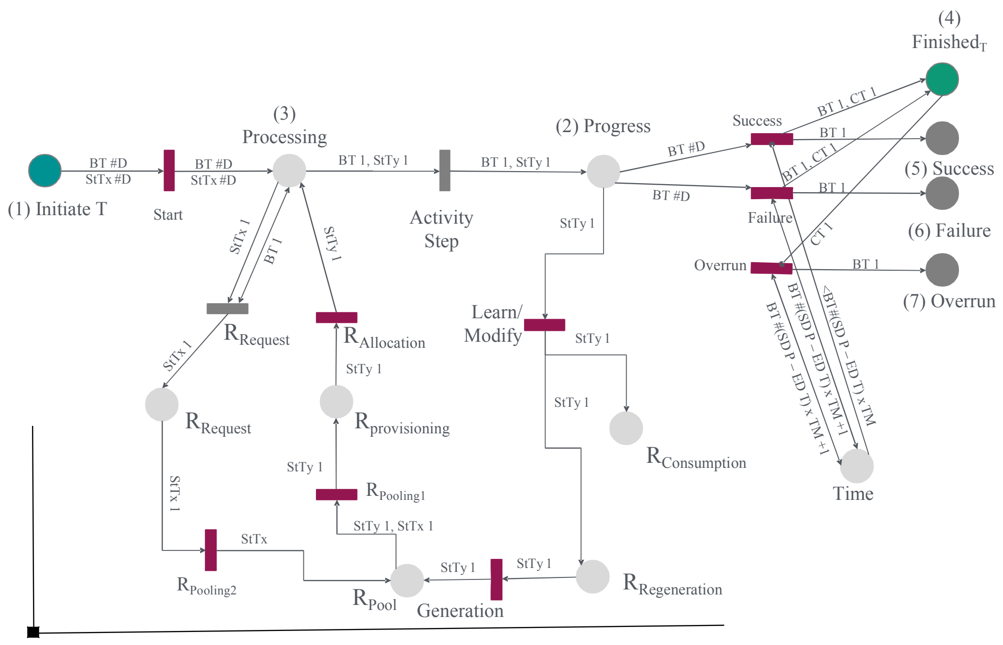
Resource allocation takes place in the construct of the tasks rather than the sprint. The use of structured tokens is illustrated for a task construct in Figure 4 as an example.

Due to different processing times, start dates, or operational delays (employee assignment, vacation, et cetera), individual tasks are terminated at different times. A sprint is only considered to be complete when all of its subtasks have been completed. As a result, the individual tasks for completing a sprint must be brought together again. The implementation of these requirements in the illustrated PN is demonstrated in the right half of Figure 3. It becomes clear that the interface of a task must include an inquiry regarding its status (completed/not finished). This is solved in the pictured PN by the inclusion of the place *Finished* for each task. The merging of individual tasks is realised by a transition *Sprint Ended*. As soon as there is a token for each task in its *Finished* place, the *Sprint Ended* transition fires and the sprint is ended. This merging by means of an additional transition is necessary because only one token is needed for further processing after a sprint. The transition *End Sprint* reduces the number of tokens to one, regardless of the number of assigned tasks. The place *End Sprint* indicates whether the sprint has ended. [6]

4.4 Execution Using the Example of the Petri Net Construct Task

The goal of a sprint in agile PM is to complete functional intermediates. To accomplish this goal, several tasks are performed in a sprint. In the selected mixed project method, the tasks are self-contained and all requirements from the parallel running requirement analysis have already been identified. No dependencies among the tasks themselves should occur in theory. However, this can never be ruled out in practice and therefore it is assumed that there are also predecessor-successor relationships between the tasks set up in a sprint, as opposed to the sprint model in Figure 3. The execution of tasks can take place in parallel within a sprint. As shown in Figure 4, each task is divided into a resource allocation, time allocation, and the actual processing of the task. Figure 4 shows a section of an extended scope as realised with the tool. Independent of this, there may be a need to process tasks sequentially, even in the case of unrelated tasks. For example, this sequential processing is necessary if the same team is always responsible for the implementation of the tasks during the sprint. Furthermore, a task is always divided into design and development phases. In the design phase, the requirements are transformed into a technical concept, which will then be implemented during the development phase.

The general structure of these two phases is identical and occurs strictly consecutively, likewise the requirements sets are identical. For simplicity, this wider subdivision is omitted and only the design phase is considered. In Section 4.5 the same construct is presented with respect to the correct determination of the initial marking of black token. The figure 4 represents the construct of the task, its interfaces with the sprint (green) and all required interfaces with other constructs (light grey), as well as timed (grey) and timeless transitions (red). It is demonstrated, how the interaction of the resource allocation can occur in a strongly simplified way. In general, after each step in the model, the resource allocation is provided with the help of structured token. There will be always a communication between the outstanding resource requirements and an already completed resource provision. With the construct presented here, a task can have any number of resources of any kind (data, things, people) assigned to the provided interface.



➢Two Interfaces are available per task to the sprint: Place *Initiate Task* and place *Finished*
➢(#Number) Steps to calculate the correct initial mark in chapter 4.5
➢*Legend: Black Token (BT), Coloured Token (CT), Duration (D), End Date (ED), Project (P), Resource Allocation (RA), Start Date (SD), Structured Token (StT), Task (T), Time Model (TM)*

Fig. 4. Excerpt of the task construct and interaction of resource allocation

The interface to the time construct ensures that the task can start no earlier than the specified start date. It is important that three types of time information (duration, earliest start, and latest end) are taken into account. This enables the possibility of a later start because of the review of deadline for completion, including the specified duration. If this is not the case, the overrunning time will be measured. Likewise, no steady processing is mandatory, and the progress is recorded separately. [6]

4.5 Calculation of the Correct Starting Initial Marking

There are three kinds of data: planned data, actuals and forecast. The combination of these type of data can lead to different PN models and simulation capabilities. A special characteristic of the planned-actual data PN model is the correct determination of the initial marking. This procedure will be clarified using a sample of one place. The order of the calculation of marking is coordinated such that the results build on each other. One special feature is that arc expressions for mapping the time constraints are not influenced by the actual data. This method ensures that the calculation of the overdraft duration remains unchanged in relation to the planned data and that overdrafts will continue to be calculated correctly.

The calculation of the initial marking takes place for all constructs as soon as the selected time is past the start of the project, assuming that all actual data are available at this time. The place *Initiate Task* in Figure 3 is only relevant during the design phase. During the development phase, this step is no longer required as it is part of the task itself and directly follows the already initiated task from the design phase.

The calculation for the *Initiate Task* place is defined such that as soon as the start date of the task is equal to or greater than the current date and the progress is 0% (meaning the task has not yet started), a black token will be located at this place. If these conditions do not apply, the place will be initiated without a token. Table 2 explains the calculation for the other places and their black tokens at a high level. The number (x) in the first column refers to the places in figure 3. If the conditions in the table are not met, the marking is zero for black tokens. Depending on the input of the formalized table structure, the marking will be automatically produced.

Figure 4 also exemplifies the use of test, inhibitor arcs test and inhibitor arcs with weight. An access to the interface of the time construct may only be done in general with a test arc to ensure that the number of tokens in this place remains unchanged. The transition *Overrun* is no longer capable of firing once the task is completed. This is realised by an inhibitor arc to the place *Finished*. Within the scope of this article, the explanation will be limited to this simplified example. Finally, a highly simplified example for the usage of a structured token is presented. In the experimentally implemented PN using Peneca Chromos, the structured token was taken into account in the allocation of resources for the automatic generation. The structured token records the information that all resources (especially supplies) could be linked with the information that a work step has been processed.

Table 1. Conditions and calculations of an initial mark of black tokens for a task.

Place	Condition	Effect on Token
(1)	Start Date = current date & progress = 0	1 Black Token
(2)	Task is not initiated & progress ≠ 100 %	(Duration * Progress) Black Token
(3)	Task is not initiated & progress ≠ 100 %	(Duration - #(Number of Tokens in the Place Progress)) Black Token
(4)	Task done 100% Next phase not yet started	1 Coloured Token 1 Black Token
(5)	Actual End Date is set before or at the planed End Date & Progress = 100%	1 Black Token
(6)	Exceeding the planed end date	1 Black Token
(7)	Exceeding the planed end date	(Actual-End Date)-(Plan-End Date) Black Token

4.6 Automatic Generation and Definition of the Petri Net Based on the Common Denominator of the Table

This section is intended to exemplify the problem of automatic generation based on deviation from planned and actual data. For the sake of simplicity, automatic re-planning is explicitly dispensed with, but the possibility of a reconfiguration by the project planner is preserved. Figure 5 illustrates the approach of the experimental tool support presented for the automatic generation of a PN. The raw data of the concrete project are recorded in tables in Excel format. Based on the independent table format, a program (here developed in C #) is used to extract the data for the modelling tool Peneca

Chromos. This includes producing the required .cne file format, which is used to generate the PN. [6, 18] During the execution of a project, deviations from planning data can occur at any time. The plan-actual model (Figure 6) was considered, which is based both on planning and current data. The delineation of planning and actual data was taken into account in the developed table format.

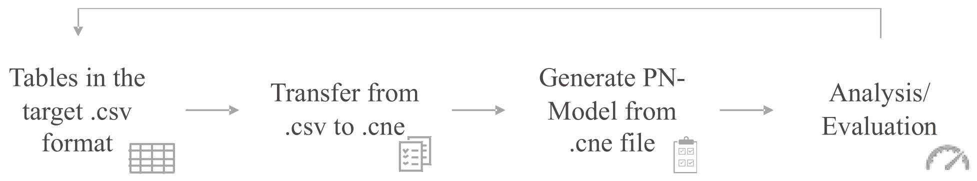

Tables in the target .csv format
Transfer from .csv to .cne
Generate PN-Model from .cne file
Analysis/ Evaluation

Fig. 5. Overview of automatic generation [6]

At the beginning of the program, the paths are added to the project scheduling tables and lists are created to store the data. Based on the current date provided for the start of the project, the parsing of the planning data is carried out using the tables. After the actual data have been taken into account until the key date, the planning data are taken into account without further logic for the sake of simplicity. For the plan-actual model, the calculation of the correct initial marking is a decisive step that must be carried out for all respective constructs.

The designed constructs cover a project setup, with two main framework concepts - the traditional and the agile. There is basically no connection between them and thus an independent parallelism and execution is possible. The advantages of modular concepts are of great importance here. Basically, the interactions defined through interface are the key. The defined table structure can take up tasks with dependencies and their status record, as well as ensure an independent assignment of tasks to implementation blocks (sprints).

All important information must be taken from the tables per construct. Take the example of a sprint, information must be obtained on how many tasks the coordinator has to initiate, which influences the number of arcs. As soon as the network was generated by the data in the tables, the necessary initial markings are set. As a result, achievable markings can be assigned to the table structure and reproducibility is ensured. Each possible marking can be retraceable to the table. In order to do this, the table has to be extended with forecast columns. The network with all intermediate markings can thus be saved in the table. As an example, the required runtime of the sprints and their assigned tasks can be write back to the table. How the individual steps must be programmed depends on the choice of the underlying tool.

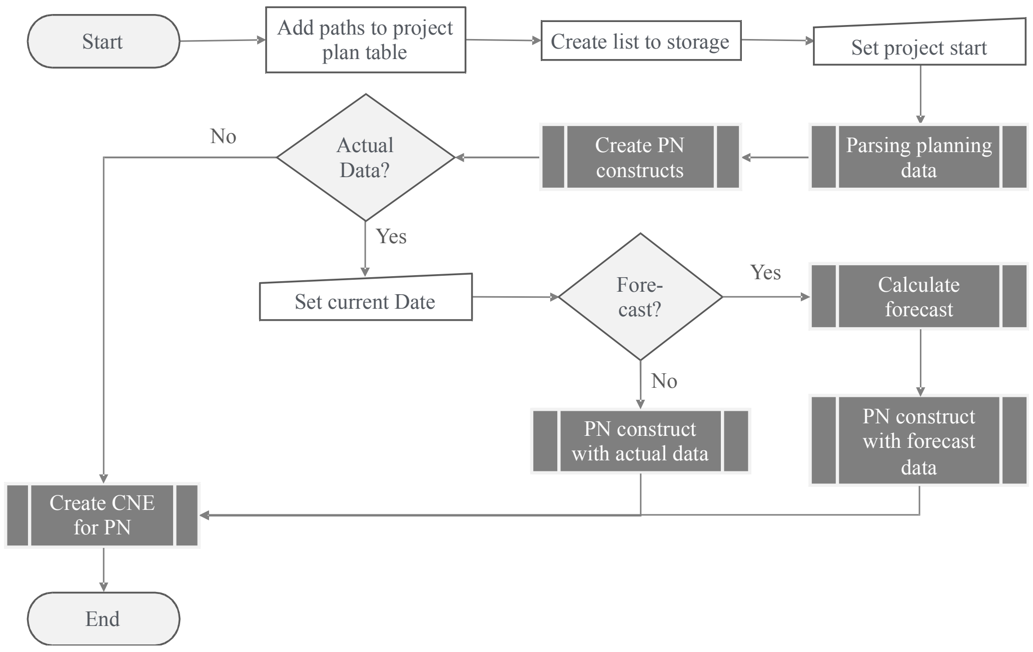


Fig. 6. Rough program overview for creating the plan-actual model [6]

5 Summary and Conclusion

Experience and analysis of PM has led to the systematic model presented in this paper, which also takes into account the agile PM method. An easy-to-handle and extensible table structure as part of the systematic model was only described in principle and assumed as a given input. This table structure enables a user to apply the proposed model for PM without PN knowledge. The paper discusses a method to simulate and analyse project stages and enhance current methods using PN. It is a powerful method in terms of modelling capability and can present advanced features to analyse and can be easily transformed. The advantages are exemplary results from the formal table structure, PN is based on a powerful and easily understandable formalism, all possible system states (dynamic) can be represented, or a regeneration and rescheduling of activities at the time of failures and resource restrictions can be enabled.

PN constructs can be generated from the defined table structure, which includes the mapping of various predecessor-successor relationships and illustration of different dynamics. The resulting PN includes structures, simulation capabilities, and initial and intermediate markers and can support the PM by allowing the results to be fed back and used for project decisions. This also comes with the advantage that changes in the project planning and implementation do not need to be realised in all related tables. Instead, changes can always be taken into account by simulation to predict their impact, starting at the initial state to a defined point due to current intermediate states of the PN. A dynamic and regular comparison of project status and even forecasts can be done quickly. This comparison is made possible by a variety of proposed PN models, where in this article the focus was limited to the planned-actual data PN model with attention given to tasks and their processing in a sprint. The calculation of the correct intermediate state as a new initial marking is of great importance for the PN models and has been briefly exemplified. Currently selected areas of the mentioned cycle has been experimentally implemented with Peneca Chromos. In this case, the automatic generation and the modularity of the PN constructs to be generated were taken into account, as well as the implementation of a simplified concept of structured tokens and were realised for a selected, limited project example. The method could only briefly be explained in the paper. Further on more detailed constructs has to be developed, and an investigation and definition of constructs and their interfaces needs to be a part of it. Future work could include a verification of the entire range of PN extensions from different areas (workflow, stochastics, fuzzy logic, etc.) for possible integration, or case studies for independent practical analysis. An interesting extension would be for simulations and prognoses in the field of resource allocation with a defined set off criteria and the help of fuzzy logic in case of conflicts, or a differentiation of overdrafts and their reasons.

6 References

1. Corcoran, S. S.:Are You Still Using Spreadsheets for Project Management? In: Viewpoint Construction Blog (blog) https://blog.viewpoint.com/viewpoint-team-project-management-software/ (2019).
2. Hastie, S., Wojewoda, S.: Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch. In: https://www.infoq.com/articles/standish-chaos-2015 (2015).
3. Mejía, G., Karen, N., Carlos, M., Sánchez, M. Palacios, J., Amodeo, L.: A Petri Net-Based Framework for Realistic Project Management and Scheduling: An Application in Animation and Videogames. In: Computers & Operations Research 66 (Februar), pp. 190–198, (2016).
4. Thein, C.: Investigation of Possiilies of Project Modeling Based on a Theoretical Petri Net Model – In German: Untersuchung der Möglichkeiten der Projektmodellierung auf Basis eines theoretischen Petri-Netz-Konzeptes. In: Master Thesis, TU Ilmenau, Betreuerin: Maxi Weichenhain, (2017).
5. Kumanan, S., und K. Raja.: Modeling and Simulation of Projects with Petri Nets. In: American Journal of Applied Sciences 5 (12), pp. 1742–49, (2008).
6. Schott, H.: Investigation of Possiilies of Project Modeling Based on a Theoretical Petri Net Model - In German: Untersuchung der Möglichkeit zur Projektmodellierung auf Basis eines theoretischen Petri-Netz-Modells. In: Master Thesis, TU Ilmenau, Betreuerin: Maxi Weichenhain, (2019).
7. Reisig, W., Desel, J.: The concepts of Petri nets. In: Software & Systems Modeling 14 (2), pp. 669-683, (2015).
8. Jensen, K.: Coloured Petri Nets. Springer Berlin Heidelberg, Berlin/Heidelberg (1992).
9. Genrich, H. J., Lautenbach, K.: The Analysis of Distributed Systems by Means of Predicate/Transition-Nets. In: Semantics of Concurrent Computation. Springer-Verlag, pp. 70:123–46. Berlin/Heidelberg (1979).
10. Ali, K., Wolfgang, F., Däne, B.: Extended coloured Petri nets with structured tokens formal method for distributed systems. In: Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium (TMS-DEVS) Bd. 4., Boston, MA, USA (2011).
11. Jeetendra, V. A., Krishnaiah Chetty, O. V., Prashanth Reddy, J.: Petri Nets for Project Management and Resource Levelling. In: The International Journal of Advanced Manufac-

turing Technology 16 (7), pp. 516–20, (2000).
12. Bevilacqua, M., Filippo E. C., Mazzuto G.: Timed Coloured Petri Nets for Modelling and Managing Processes and Projects. In: Procedia CIRP 67, pp. 58–62, (2018).
13. Cohen, Y., Ofer Z.: Modelling and Scheduling Projects Using Petri Nets. In: International Journal of Project Organisation and Management 1 (2), pp. 221, (2008).
14. Niño Mora, K. Y., Mejia Delgadillo, G., Sanchez, M. A., Palacios Bonilla, J., Vargas Flores, H.: Una Platforma Para La Gestión Y Programación De Proyectos Utilizando Redes De Petri. In: Dyna Management 3, (2015).
15. Qu, Y., Wang, M.: A risk emergency management model for software project based on stochastic Petri Nets. In: 2015 7th International Conference on Modelling, Identification and Control (ICMIC), 1–6. Sousse, Tunisia: IEEE (2015).
16. Rossberg, J.: Introduction to Scrum and Agile Concepts. In Agile Project Management with Azure DevOps, 67–123. Berkeley, CA: Apress (2019).
17. Zimmermann, A., Dehnert, J., Freiheit, J.: Modeling and Performance Evaluation of Workflow Systems. In: CiteSeer (2000)
18. Yen-Liang, C., Ping-Yu, H., Yuan-Bin, C.: A Petri Net Approach to Support Resource Assignment in Project Management. In: IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans 38 (3), pp. 564–74, (2008).