COMPARISON OF AGILE AND TRADITIONAL PROJECT MANAGEMENT: SIMULATION OF PROCESS MODELS

[Srovnání agilního a tradičního projektového managementu:
simulace procesních modelů]

Nils Engelhardt[1]

[1]*FOM University of Applied Sciences for Economics and Management gGmbH, Herkulesstraße 32, 45127 Essen, Germany and UCAM - Universidad Católica San Antonio de Murcia, Monasterio de los Jerónimos s/n, 30107, Guadalupe (Murcia), Spain*
Email: nils.engelhardt@gmx.de

Abstract: There are various process models for the successful implementation of product development projects. In addition to traditional project management - mostly represented by the waterfall or V model - methods of agile project management, e.g. Scrum, are increasingly in use. In the literature, the different approaches have already been described in detail. Here, an attempt is made to compare both process models by means of a simulation. This paper describes the implementation in a simulation model by using the software Matlab/Simulink®. The different approaches are modelled in a way that their effects on the project result can be statistically evaluated. A generic project was created with randomly generated parameters of the project properties. The various approaches of project management are then applied in sub-models to this project. The aim of this paper is to discuss the transfer of the characteristics of the project management approaches into the simulation and to show that the differences can be modelled accordingly.

Keywords: agile project management, product development, simulation, traditional project management.

JEL classification: C63, O32

Introduction

There are various process models for the successful implementation of product development projects. In addition to traditional project management (TPM) - mostly represented by the waterfall or V model - methods of agile project management (APM), e.g. Scrum, are increasingly in use. In the literature, the different approaches have already been described in detail (see for example Fernandez & Fernandez (2008), Cooper & Sommer (2016), Salameh (2014), and Wysocki (2009)). Shenhar and Dvir (2007) discussed how different framework conditions or prerequisites influence the success of the respective approaches. Dybå & Dingsøyr (2008) examined the main differences between both approaches (see table 1).

Table 1: Main differences between TPM and APM

	TPM	APM
Assumption	Systems are fully specifiable and predictable	Continuous Design Improvement based on rapid feedback and change
Management	Command and control	Leadership and collaboration
Knowledge Management	Explicit	Tacit
Communication	Formal	Informal
Development model	Life-cycle model	Evolutionary, iterative
Organization	Large, mechanistic	Small, organic
Quality control	Heavy planning, late heavy testing	Continuous Control of requirements, continuous testing

Source: based on Dybå & Dingsøyr (2008).

Boehm & Turner (2004) compared how the traditional and the agile project management approach relate to different characteristics of a project (see table 2).

Table 2: Characteristics of projects and their fit for the project management approach

Characteristics of project	**TPM**	**APM**
Criticality	Extreme	Low
Developers Experience	More Junior	Senior
Product requirements	Stable requirements and specs	Change often during project
Project team size	Large	Small
Company culture	Demands order	Responds to change

Source: based on Boehm & Turner (2004).

An attempt is made to compare both process models by means of a simulation. Instead of using interviews to determine the personal opinions of project stakeholders, an objective comparison should be achieved. In addition, the suitability of the methods is checked on each identical project with the same properties (see chapter 2.1).

1   Methodology: Description of the simulation model
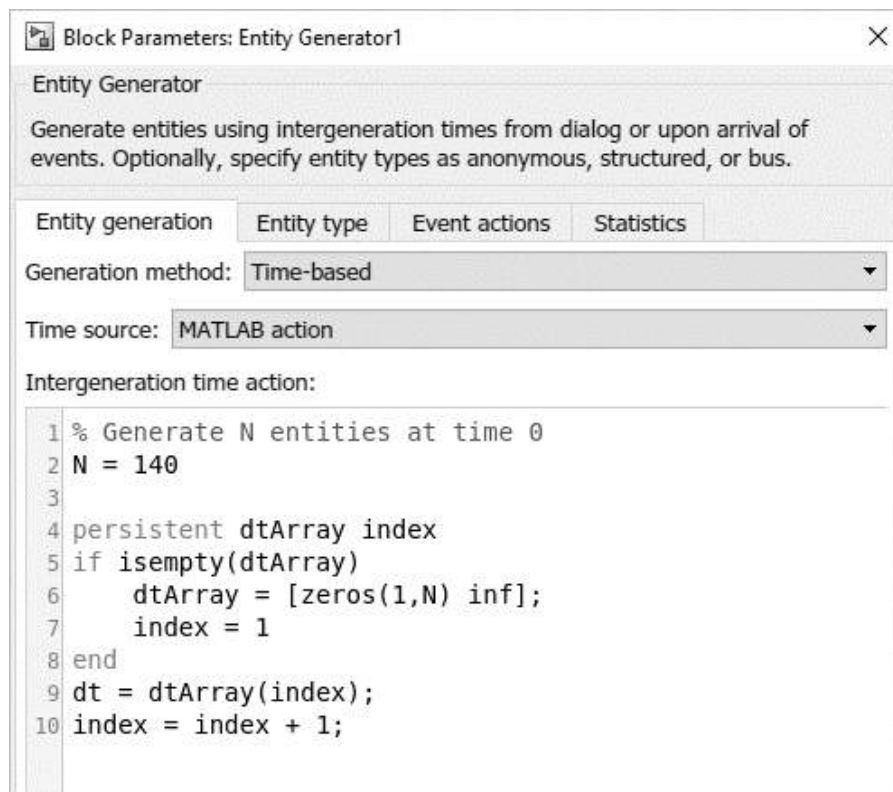
1.1   General description of the model

For the simulation, the program package Matlab/Simulink® Version R2017B was used. After testing several simulation software, it turned out that a discrete element simulation is better suited than modelling a continuous process. For this purpose, the SimEvents® module is available within the program package.

The detailed description of the model, all parameters and properties of the blocks as well as the used functions and calculations are available as HTML and PDF files of more than 450 pages. Therefore, only screenshots are presented in the text for clarity.

In SimEvents, the discrete events are called "entities". In TPM, an entity corresponds to a specific work package, which was determined in the project planning in the Work Breakdown Structure. In APM, a project is structured more from the user's point of view, so requirements or features are planned instead of work packages. The simulation assumes that the entity represents a suitable abstraction for both approaches. This results from the consideration that the total effort, which is determined in the initial estimation, should be the same, as well as the associated uncertainties and resulting errors. The term "entity" is thus an abstraction, which can be applied in its meaning as a "task" for both approaches.
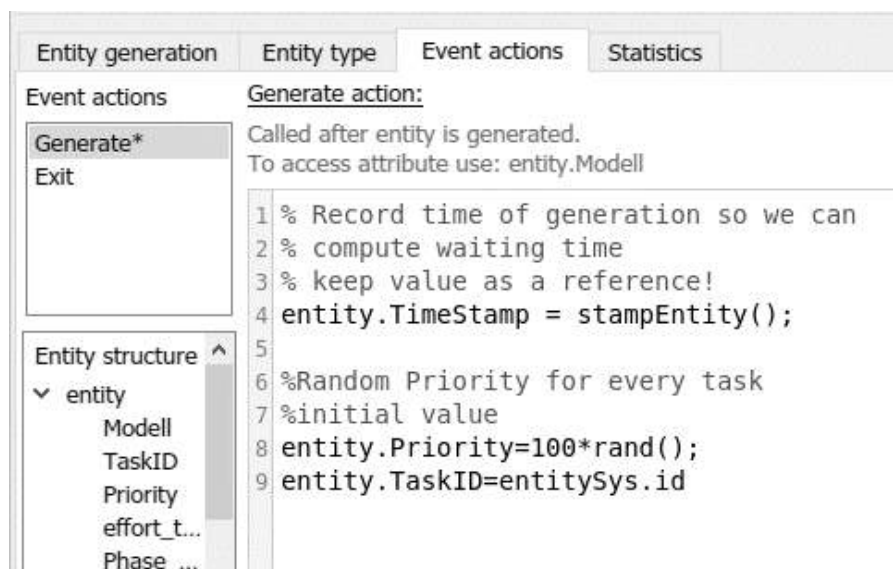
Each entity is assigned parameters with random values. These correspond to the framework conditions of the project which were determined as possible factors influencing the suitability of the approaches. Other parameters (e.g. for planned effort) allow a statistically more secure view through their random values.

To be able to compare the different projects, it is necessary to work with the same input variables (e.g. effort or priority of the individual tasks) and values of the framework conditions. Therefore, in the so-called entity generator, a defined number of tasks are generated (see figure 1), combined in the next step with initial random values of the parameters (in SimEvents called attributes) (see figures 2 and 3) and then replicated for the different sub-models. This is shown in figure 4.

**Figure 1:** Generation of a defined number of entities

Block Parameters: Entity Generator1 ×

Entity Generator

Generate entities using intergeneration times from dialog or upon arrival of events. Optionally, specify entity types as anonymous, structured, or bus.

| Entity generation | Entity type | Event actions | Statistics |

Generation method: Time-based ▼

Time source: MATLAB action ▼

Intergeneration time action:

```
1 % Generate N entities at time 0
2 N = 140
3
4 persistent dtArray index
5 if isempty(dtArray)
6     dtArray = [zeros(1,N) inf];
7     index = 1
8 end
9 dt = dtArray(index);
10 index = index + 1;
```

Source: Screenshot of simulation model
For this project, there are 140 tasks or features defined at the beginning.

Figure 2: Initial identification and prioritization

| Entity generation | Entity type | Event actions | Statistics |

Event actions — Generate action:

Generate*
Exit

Called after entity is generated.
To access attribute use: entity.Modell

Entity structure
∨ entity
   Modell
   TaskID
   Priority
   effort_t...
   Phase_...

```
1 % Record time of generation so we can
2 % compute waiting time
3 % keep value as a reference!
4 entity.TimeStamp = stampEntity();
5
6 %Random Priority for every task
7 %initial value
8 entity.Priority=100*rand();
9 entity.TaskID=entitySys.id
```
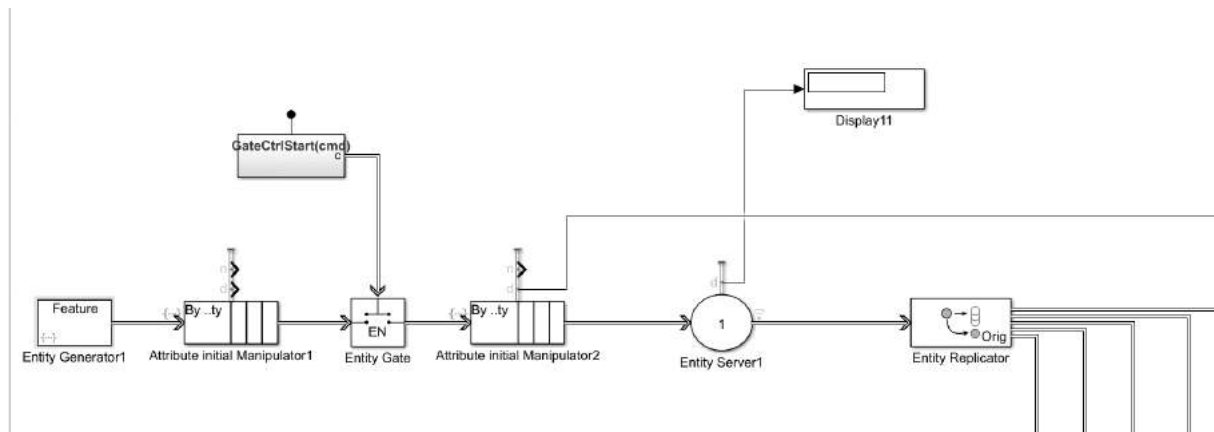
Source: Screenshot of simulation model
All the tasks get a time stamp to calculate the time needed until completion, and an ID to follow the task through all the sub-models and to compare the same task between the models. The priority for each task is randomly assigned at the beginning, it has a value between 0 and 100, with 100 being the highest priority.

Figure 3: Exemplary initial parameterization in the Attribute Manipulator
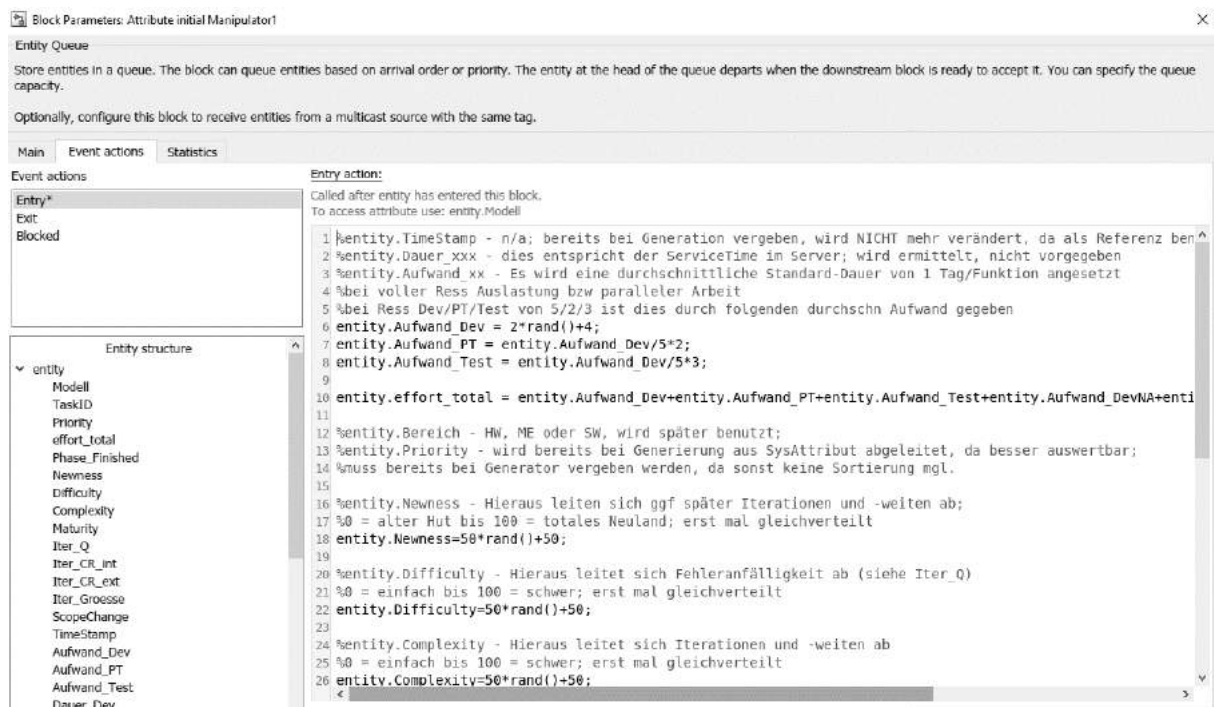
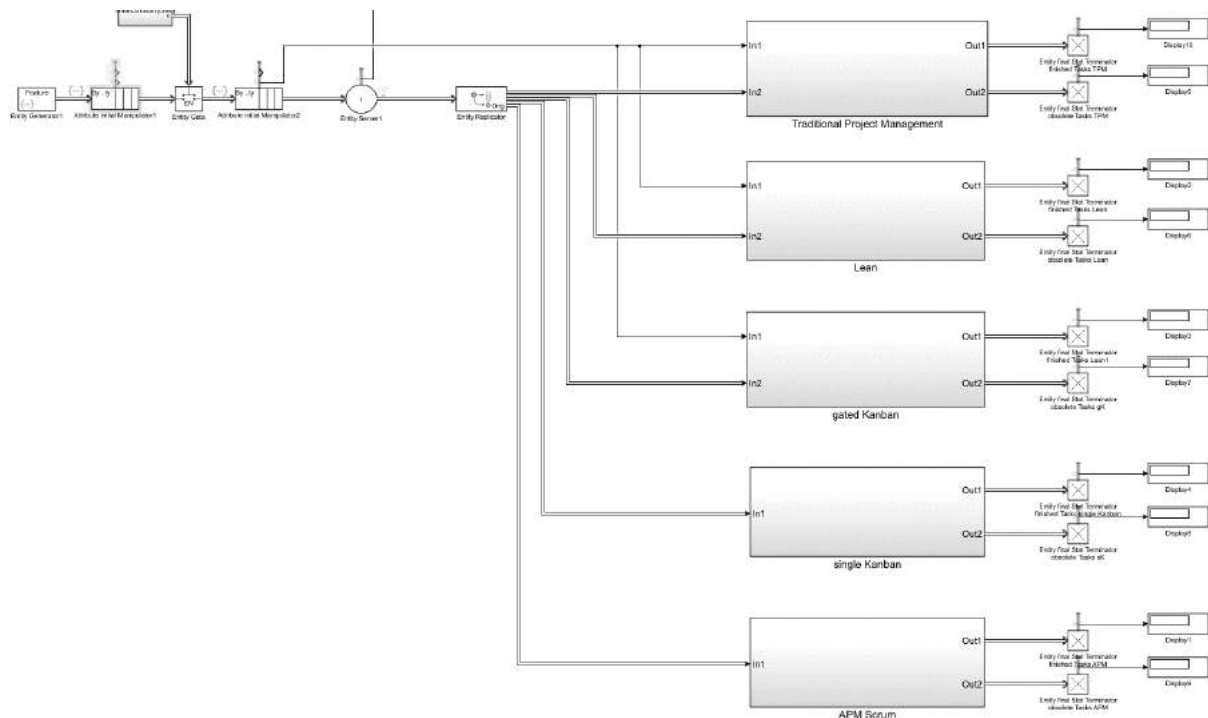*Source:* Screenshot of simulation model

The Entity structure lists all the parameters. Depending on the type of the project, the parameters for the project properties (newness, difficulty, complexity and maturity) are assigned randomly, in this case with values between 50 and 100, which corresponds to a high level of these parameters.

Figure 4: Initial generation, parameterization and replication of the tasks

Source: Screenshot of simulation model

This is the initial flow for all tasks. After generation and manipulation, the tasks are separated in the Entity server. The Entity replicator copies all the same tasks in the 5 sub-models, so all sub-models use the same tasks with the same initial parameters.

The simulation model shows 5 paths (i.e. sub-models), which correspond to the different approaches in varying degrees (see figure 5). These paths are referred to in the simulation as Model 1 to 5. Each task after the branch receives an attribute with the corresponding value for the corresponding model.

Figure 5: Overview of the complete model: Parallel comparison of the models after replicating the entities

Source: Screenshot of simulation model

The overview of the sub-models can be found in table 3:

Table 3: Description of the sub-models

Model	Name	Description
1	Traditional Project Management (TPM)	Model of a traditionally structured multi-phase project. The tasks are processed sequentially. Only when one development step is completely done, the next one is started.
2	Lean	Model is structured in phases similar to the TPM. The difference to model 1 is that within the phase, the tasks are processed in the one-piece flow model and only collected at the end of the respective phase for the review gate.
3	Gated Kanban	The Kanban model limits the number of tasks per development step. The gated model collects the tasks at the end of the review phase, similar to the model 2 "lean".
4	Single Kanban	The model is built in the same way as the gated Kanban model, but in contrast to this, the processed tasks are not collected for a Review Gate but evaluated immediately after execution.
5	Agile Project Management (APM) Scrum	This model depicts the Scrum approach. The respective phases are not determined by the number of tasks, but the time per phase is precisely determined and is not changed.

Source: own illustration.

All paths use their own resource pool, there are resources for developers, prototype builders and testers. All corresponding resource pools have the same contents (resource amount, i.e.,

number of employees) as well as the same settings (utility usability parameters). Thus, both a comparability of the models is ensured, and mutual interference is excluded.

1.2 Structure of the development process and its transfer into the simulation

1.2.1 General structure of the development process
To be able to compare the two approaches, it did not seem sensible to model a particular development project, which was carried out according to one of the two approaches. The transfer to the other model partly contradicts the principles of this model (for example, detailed finely-planned tasks for all project phases are incompatible with the agile model). Therefore, the development process was implemented in a more abstract level. It consists in all sub-models of the steps development, prototyping and testing. After the testing, within each phase a rework consisting of the same steps is assumed (see figure 6).

**Figure 6:** Implementation of the steps development and rework development
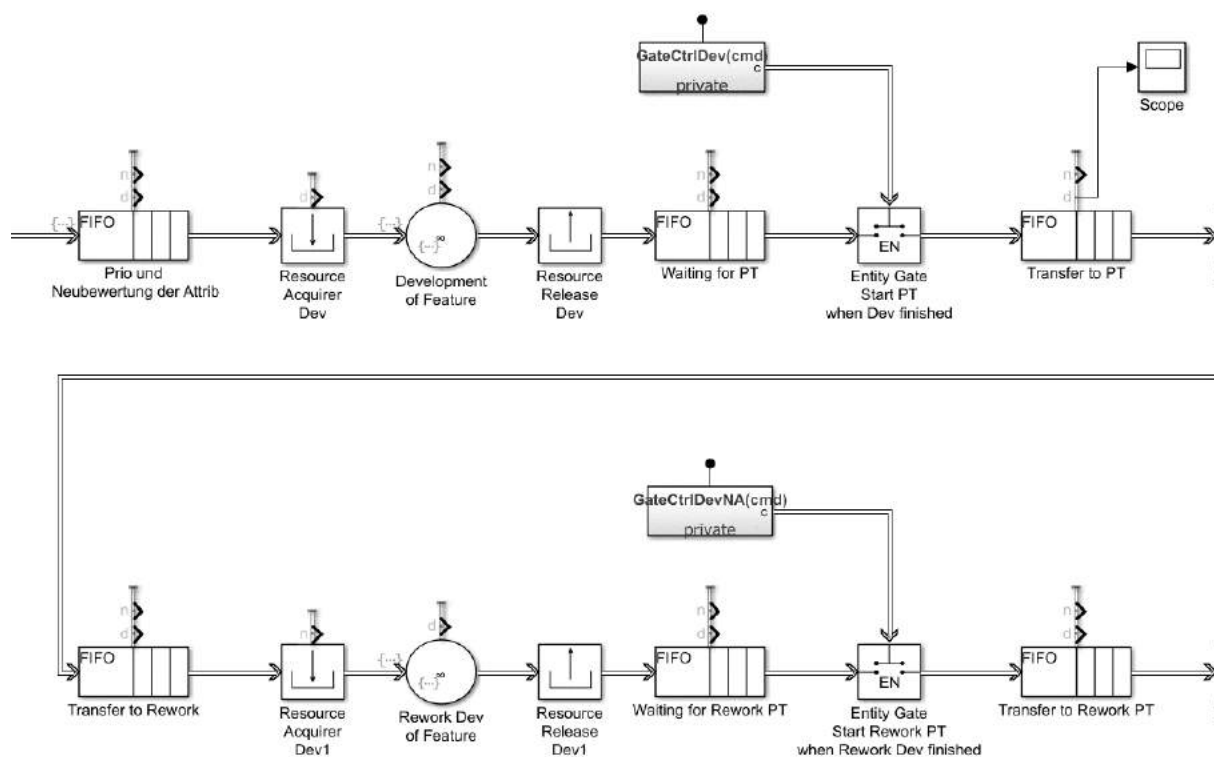
Source: Screenshot of simulation model
This picture shows a part of the general process within a phase or iteration. This is the same in all models. For each processing the corresponding resources are assigned. Between the process steps, there are FIFO buffers and gates, which are used in some of the sub-models.

1.2.2 Review Gates
After a task has been finished, it has to be evaluated. Each task is assessed in a review, whether it has been completely done, whether it is still necessary to rework, or whether it cannot be used. Depending on the sub-model, the tasks are all checked together at the end of a development phase, or individually after completion or at the end of a specified time. However, the evaluation criteria as well as the required parameters and their calculations are modelled using the same function block in all sub-models (see figure 7).

Figure 7: Process flow of a review gate
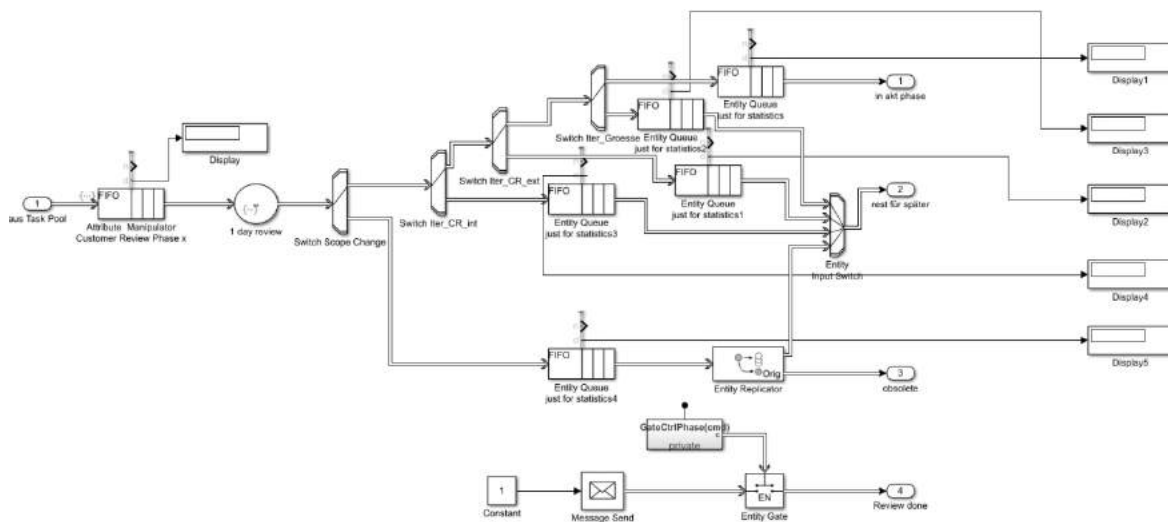
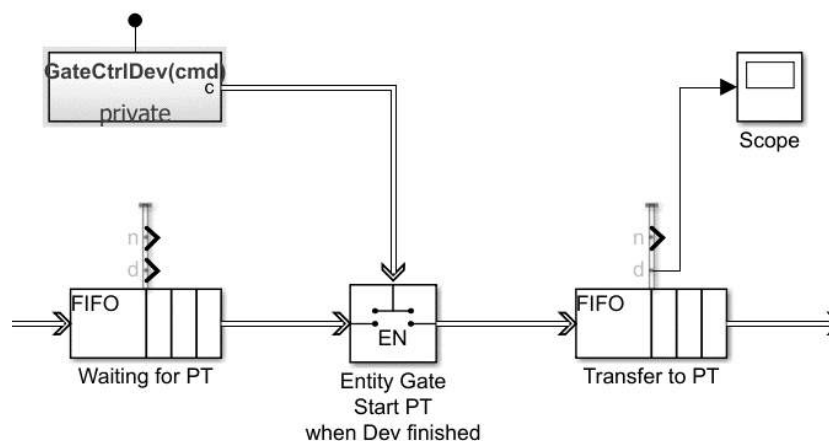*Source:* Screenshot of simulation model

There are several criteria to evaluate the completion of a task. For example, rework may result from internal or external change requests or poor quality. In each switch, the corresponding calculations are used to classify the tasks as done or to return them to the next phase or iteration.

1.3   Description of the implementation of the different approaches

1.3.1   Model 1 - Traditional Project Management

At the beginning of each phase, a specific number of tasks are defined, which are to be processed in this phase. For this, the tasks with the highest priority at this time are selected. During the phase, the procedure is sequential. Each sub-step is run through by all tasks, only after all scheduled tasks have been processed, the next sub-step is started. This is modelled by so-called entity gates (exemplified for the transition from development to prototyping (see figure 8).
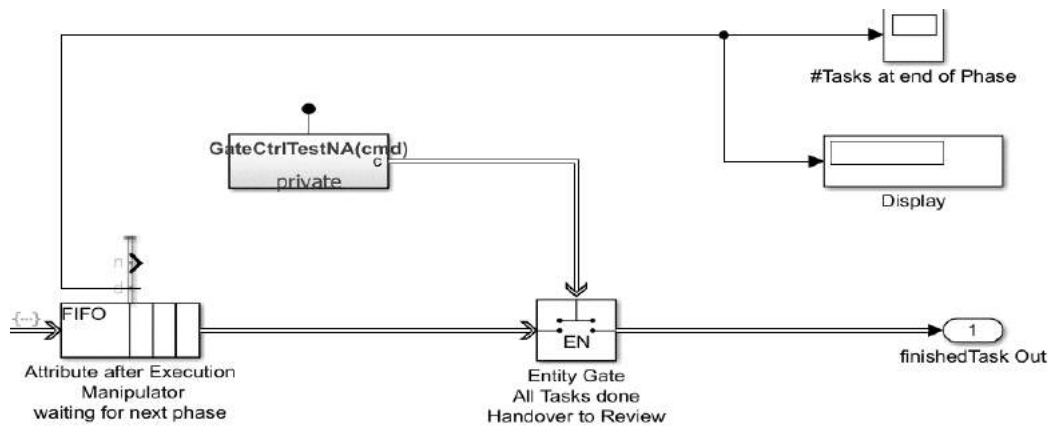
Figure 8: Gate for processing a partial step in the batch

Source: Screenshot of simulation model

At the end of each phase, the tasks are collected to perform a review of all tasks being processed in this phase. At the same time several parameters are adjusted to reflect learning curves or changing environments (see figure 9).

**Figure 9:** End of a phase in the TPM sub-model

Source: Screenshot of simulation model

At the end of each phase, there is a gate to collect all the tasks. The gate opens after completion of all tasks that were entered into the phase at the beginning. The Attribute Manipulator adjusts some parameters to reflect learning curves or changing project environments.

1.3.2 Model 2 – Lean (One Piece Flow)

The main feature of the sub-model "Lean" consists in the application of the concept of the One-Piece-Flow. A new task is taken over into the processing when the resource is free again, so this "pulls" the next task (implementation of the pull principle). Since the respective resources perform both initial processing and rework, the model considers both for the allocation of resources, so a new task is only started when the resource is not tied up in rework (see figure 10).

Figure 10: Implementation of the one-piece flow and pull principle in the lean model using the example of development
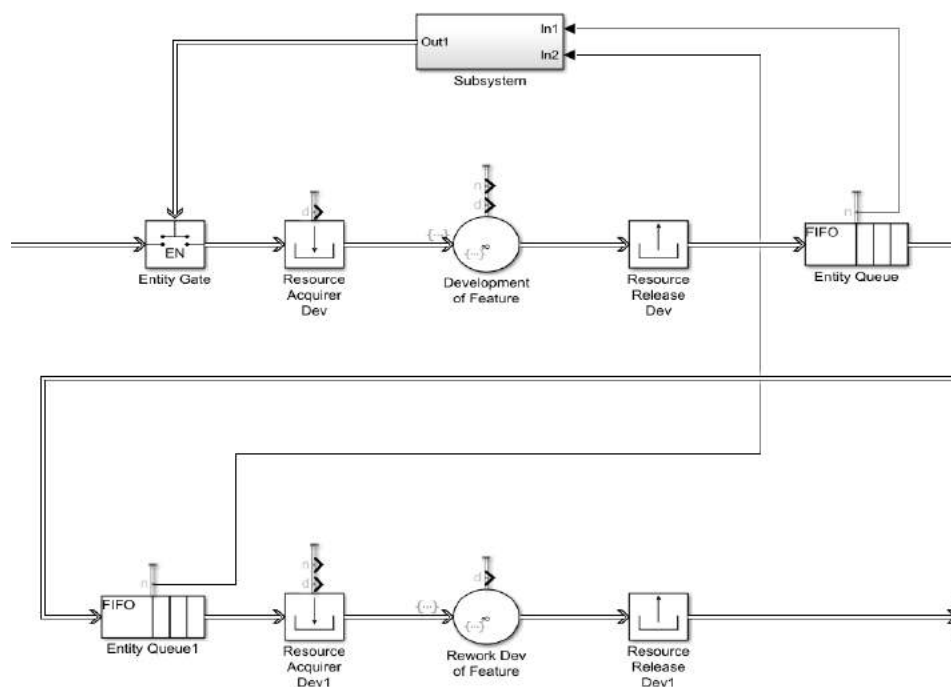
Source: Screenshot of simulation model

The gate in front of the processing step opens, when there is no task waiting for the next step and no resource is allocated for rework of that step. So, the next step pulls a task for processing.

It is assumed in the model that the overall project nevertheless runs in phases (in particular with gate reviews), which correspond to those in TPM. Therefore, the tasks for the review will be collected at the end of the phase. The planning of the tasks for the respective phase is modelled equal to the TPM, so tasks are performed in priority order.

1.3.3 Model 3 – Gated Kanban

The "Gated Kanban" sub-model also assumes a phase model with review gates at the end of each phase. Again, a specific number of tasks at the beginning of the phase is defined depending on their priority. The difference to the two previous models is the implementation of the Kanban principle of a limited "WIP" (work in progress). So only a defined number of tasks are allowed for processing at the same time. The gate at the beginning of the process opens for new tasks only when the number of tasks falls below the WIP limit (see figure 11).

**Figure 11:** Implementation of the WIP limit in the Kanban model, detailed presentation of a phase
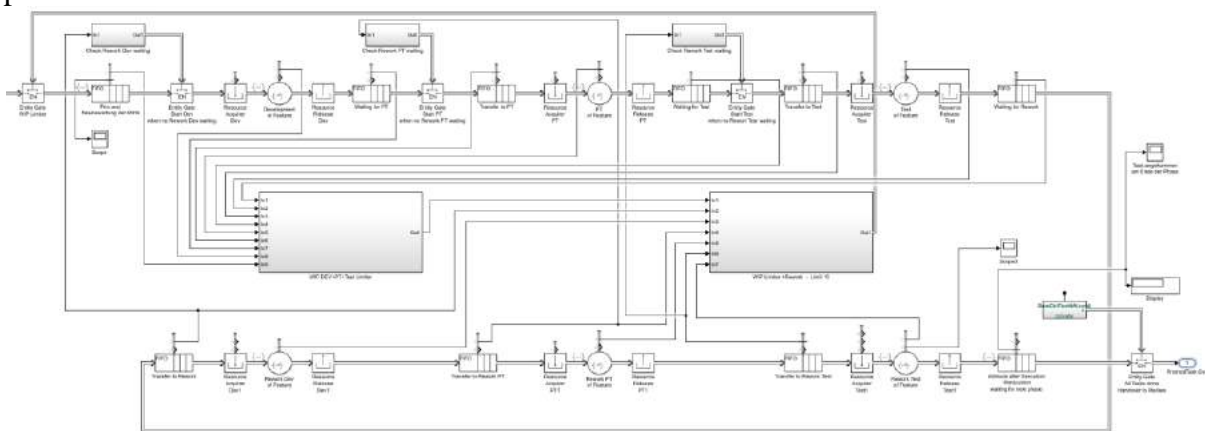
Source: Screenshot of simulation model
The big blocks in the middle add all the task being processed at that moment and trigger the gate at the beginning.

1.3.4 Model 4 – Single Kanban

The "Single Kanban" sub-model takes the idea of the WIP limit a step further. It just ensures that the WIP limit is not exceeded, but no tasks will be collected at the end of a review phase. It is therefore no longer a phase model with gates, but each completed task is immediately reviewed and either evaluated as completed or fed back for processing (see figure 12).

Figure 12: Overview of the Kanban principle without gates at the end of the phase
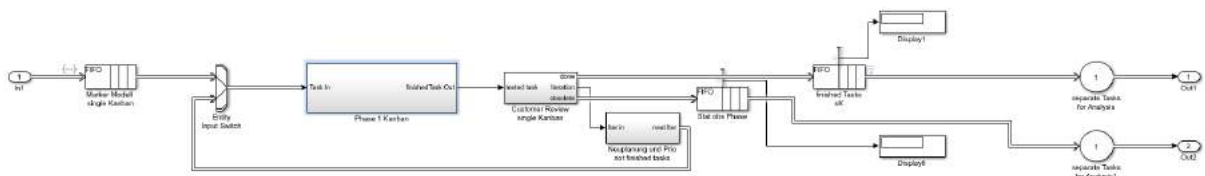

Source: Screenshot of simulation model
Because each task is processed individually, it can be fed back into the task pool (in case rework is needed) in the form of a loop. There it joins in according to its possibly adjusted priority.

This, in consequence of its implementation, is certainly rare or never to be found in practice. However, since in the literature of the APM partly the question of project phases and gates is not taken concrete or for it is referred to hybrid models, as a benchmark helpful.

1.3.5   Model 5 – Agile Project Management (Scrum)

The Scrum approach was chosen to model the APM. It proceeds in much shorter, precisely defined iteration steps than a TPM. The duration of the individual phases is defined, and tasks that were not processed during this time are returned to the task pool (the so-called backlog). This fundamental difference to TPM is illustrated in Owen (2006).

**Figure 13:** Difference between TPM and APM regarding fixed and varying targets
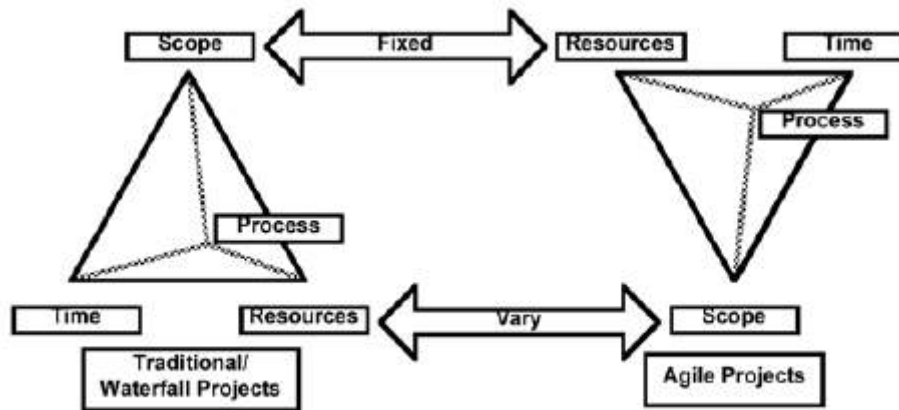
Source: Owen et al. (2006)

In case of deviations of the project, in TPM the scope is fix, so usually time and/or resources are added. In APM, the scope will be adjusted after discussion with the project owner or customer, but time and resource stay fixed.

At the beginning of a new iteration, all tasks in the backlog are scheduled according to their priority for the new so-called "sprint". The duration remains fixed, this is the main difference to the other models. At the end of each phase ("sprint") reviews are also held at the APM, which assess the execution of the tasks (done, rework or change request or obsolete).

The model in its overall view therefore resembles the model "single Kanban" (see figure 14).

Figure 14: Overview of the APM-Scrum model
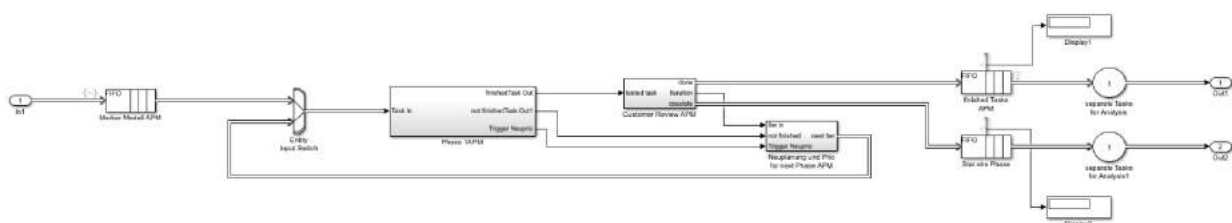
Source: Screenshot of simulation model

Within a phase, it is ensured that the rework is first processed to allow a task to be "done." Afterwards, before the beginning of each new task, it is checked whether the planned time for development, prototyping and testing is shorter than the remaining time of the sprint, i.e. whether completion within the iteration is possible. If not, it will go directly into the backlog. After the set time, all completed tasks are reviewed, the rest moves back to the backlog (see figure 15).

**Figure 15:** Modeling a sprint, excerpt for development
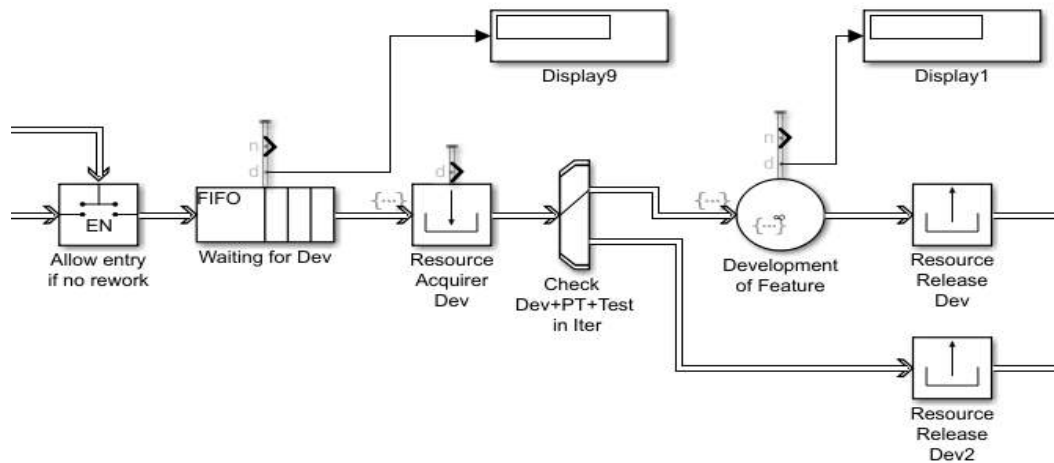
Source: Screenshot of simulation model

The switch is used to calculate, whether a task can be done completely during the remaining time. Only then, the task flows into the development process, otherwise it goes back to the backlog for the next iteration.

The time end is triggered by a timer (see figure 16).

Figure 16: Timer for the end of the sprint
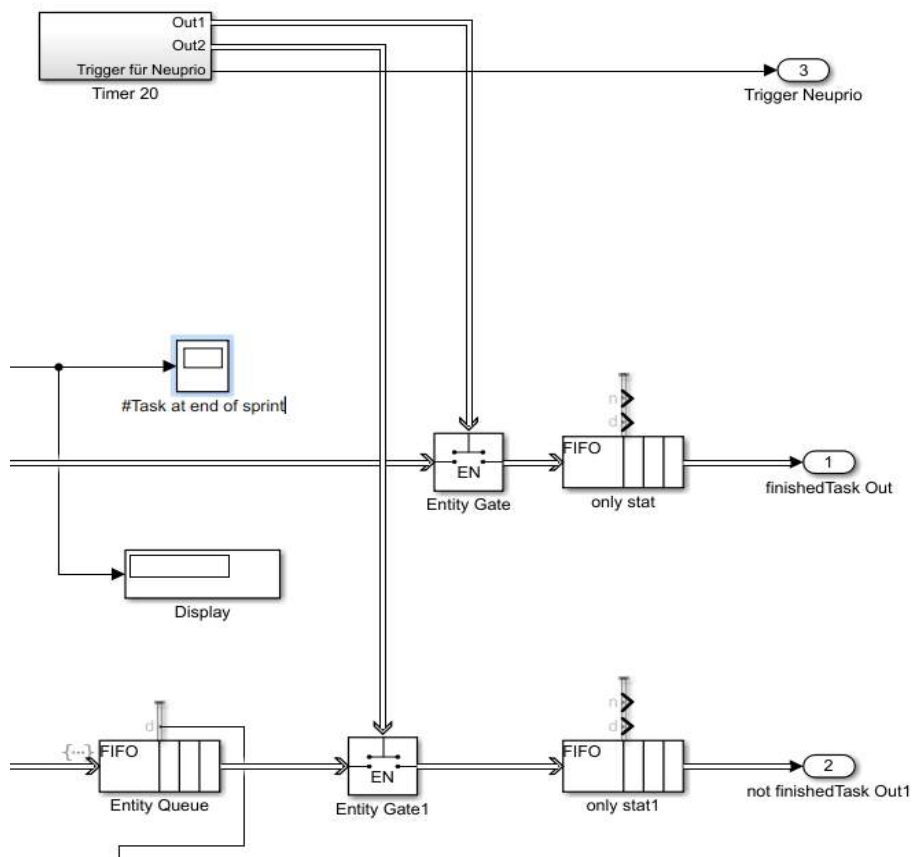
Source: Screenshot of simulation model

After the expiration of the specified time for the iteration (in this example 20 days), the gates open for both the completed and the incomplete tasks.

25

Conclusion and Discussion

The created simulation model has been verified by using a common product development project, with typical values for e.g. development effort and time and rework rates (based on experience of several development projects). Afterwards, the parameters for the project frame conditions were varied in order to determine their influence on the result. At the end of the simulation, the required effort, required rework loops and used time were determined for each individual task. It was shown that the model behaves stably and that a change in project characteristics leads to plausible results.

The evaluation criteria for the success of the project and its transfer into the model are the subject of another paper (Engelhardt (2018)), which is currently still in the review process. Herein, the statistical evaluation of the simulation outputs is also discussed depending on the project characteristics.

The derivation of the simulation parameters from literature is part of the PhD thesis. There it is also shown how a change of the parameters with increasing project duration, e.g. through learning effects or changed framework conditions is modelled. A sensitivity analysis examines how these changes affect the calculation of rework rates in the review gates. It is shown, how the data can be modified to enlarge the model.

Referring to the aim of this paper it could be shown how the characteristics of the different project management approaches are transferred into the simulation and that the corresponding sub-models reflect their special attributes.

References

[1] BOEHM, B. W. and R. TURNER, 2004. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston: Addison-Wesley. ISBN 978-0-321-18612-6.

[2] COOPER, R. G. and A. F. SOMMER, 2016. Agile-Stage-Gate: New Idea-to-Launch Method for Manufactured New Products Is Faster, More Responsive. *Industrial Marketing Management* 59 (November 2016). pp. 167–80. ISSN 0019-8501.

[3] DE WIT, A., 1988. Measurement of Project Success. *International Journal of Project Management* 6, no. 3: pp.164–170. ISSN 0263-7863.

[4] DYBÅ, T. and T. DINGSØYR, 2008. Empirical Studies of Agile Software Development: A Systematic Review. *Information and Software Technology* 50, no. 9–10 (August 2008). pp. 833–859. ISSN 0950-5849.

[5] ENGELHARDT, N., 2018. Comparison of Agile and Traditional Project Management by Simulation: Criteria for Evaluation of the result. In: *Conference Proceedings of Business and Management Sciences: New Challenges in Theory and Practice 25th Anniversary of the Doctoral School of Management and Business Administration*. Gödöllö: Szent István University (currently in review process)

[6] FERNANDEZ, D. J. and J. D. FERNANDEZ, 2008. Agile Project Management - Agilism versus Traditional Approaches. *The Journal of Computer Information Systems*, no. Winter. pp. 10–17. ISSN 2380-2057.

[7] GRAHAM, C., 2014. Agile Project Management. In: A. LESTER, *Project Management, Planning and Control: Managing Engineering, Construction, and Manufacturing Projects to PMI, APM, and BSI Standards*, 6th edition. Oxford: Butterworth-Heinemann, pp. 523-538. ISBN 978-0-08-098324-0.

[8] OWEN, R., L. J. KOSKELA, G. HENRICH, and R. CODINHOTO, 2006. Is Agile Project Management Applicable to Construction? In: R. SACKS and S. BERTELSEN, eds. *Proceedings of the 14th Annual Conference of the International Group for Lean Construction*. Santiago, Chile. pp. 51–66. ISBN 956-310-249-5.

[9] SALAMEH, H., 2014. What, When, Why, and How? A Comparison between Agile Project Management and Traditional Project Management Methods. *International Journal of Business and Management Review*. pp. 52-74. ISSN 2052-6407.

[10] SHENHAR, A. and D. DVIR, 2007. *Reinventing Project Management: The Diamond Approach to Successful Growth and Innovation.* Boston: Harvard Business School Press. ISBN 978-1-59139-800-4.

[11] WYSOCKI, R. K., 2009. *Effective Project Management: Traditional, Agile, Extreme.* 5th edition, Indianapolis: Wiley Publishing, Inc. ISBN 978-0-470-42367-7.