Paper presented at the 35th International Conference of the System Dynamics Society, July 17-21, 2017, Cambridge, MA, USA

Linking Simulator Functionality with Learning: An Extension of the Taxonomy of Computer Simulations to Support Learning

Roland Maximilian Happach[1] and William Schoenberg[2]*

[1]Institute for Diversity Studies in Engineering, University of Stuttgart
Pfaffenwaldring 9, 70569 Stuttgart, Germany.

[2]isee systems inc., Lebanon, NH, USA.

[*]corresponding author: maximilian.happach@ids.uni-stuttgart.de

Abstract

This paper links learning outcomes in form of knowledge creation and transfer (declarative, procedural, structural) to the functionality of simulators. We categorize simulators first on the number of player interacting simultaneously to generate a single run of the simulation model. There are three distinct simulator categories –analysis tools, management flight simulators and multiplayer simulators. The outcome of this research helps simulator builders to connect generic learning outcome to the functionalities of simulators and provides guidelines for briefing and debriefing.

Word count: 8.969 (without Abstract, Tables, Figures, Acknowledgements and References)

Introduction

System dynamics (SD)-based simulators are used for education, training and experimentation (Davidsen 2000; Größler 2004; Davidsen and Spector 2015). The field of education and training using SD-based simulators measures the effect of simulators on the development of mental models, the dissemination of insights and the development of knowledge (Davidsen 2000). The second field focuses on experimentation using SD-based simulators, research investigates mental models that drive human decision making (Davidsen 2000). While the second field continuously produces new research (see e.g. Moxnes 1998; Moxnes and Jensen 2009; Gary and Wood 2011; Gary et al. 2012), the research on education and training using SD-based simulators falls short. There are a considerable number of articles published to promote simulation for education and training (Machuca 2000; Salas et al. 2009; Sterman 2014), but only a few articles investigate the effectiveness of SD-based simulators on learning and training. It is somehow surprising that within the field of system dynamics, in which simulation is a well-accepted instrument (Rouwette et al. 2004), the effectiveness of simulators for education seems to be accepted and assumed without rigorous examination. This problem was already pointed out by Größler (2004) and it seems that most research about education with SD-based simulators is motivated

by him. For instance, Größler et al. (2000) derive a general experimental design for research on teaching using SD-based simulators, and they conduct experiments that empirically shows that users increased performance via the use of a simulator. However, participants of the experiment lacked the ability to transfer knowledge to new novel tasks.

Maier and Größler (2000) point out that the system dynamics community is using a variety of terms for simulators, i.e. management flight simulator, microworlds, decision support system, etc. Further, they identify the absence of a precise definition as barrier for research on the effectiveness of SD-based simulators support for learning. By presenting different criteria for categorization and concluding with a taxonomy, Maier and Größler (2000) try to clear the way for more comparative evaluation studies on the effectiveness of simulators. However, it seems that subsequent research is failing to achieve Maier and Größler's stated goal of supporting analysis on the effectiveness of simulators.

This paper resumes the work of Maier and Größler (2000) and presents an extension by connecting the functionality of SD-based simulators with their learning outcomes. By doing so, we call attention to the missing foundation between simulator design and learning outcomes, while simultaneously providing a starting point to close the gap in the research on the effectiveness of SD-based simulators on education and training.

We want to emphasize that this paper discusses 'teaching using SD-based simulators', we do not focus on learning through model construction, or how to teach system dynamics, but rather on how to teach using a SD-based simulation tool. Richardson (2014a, 2014b, 2014c) describes an approach to model-based teaching of system dynamics and Schaffernicht and Groesser (2016) link learning outcomes with the teaching of system dynamics. Our paper differs since it discusses the communication and transfer of knowledge via the use of a SD-based simulator. We urge scholars within the field of system dynamics to resume the work on a taxonomy for simulators, and to do more research on the effectiveness of simulators in the context of learning. Further, we aim to teach those who develop SD-based simulators how to design their simulators to support their desired learning objectives both in an educational and consulting context.

This paper is structured as follows. In the next section, we present a practitioner's view on the taxonomy by Maier and Größler (2000) and present the most common game-oriented simulator functionalities. In the third section, we present a perspective on learning outcomes and knowledge. The fourth section presents the link between simulator functionality and learning outcomes. In the last two sections, we discuss our findings and implications, and present a conclusion.

Categories of SD-based simulators

Terminology

Over the last decade simulation and with it simulators, have become increasingly important in teaching and training (e.g. Langton et al. 1980; Bakken et al. 1992; Alessi 2000; Schunk 2012). Especially with

the emergence of cognitive and constructivist learning theories, simulation has become an integral part of teaching (Alessi 2000; Spector 2000; Schunk 2012). Additionally, the advancements in technology promotes the use of simulation. The number of SD-based simulators has steadily increased, especially with the advent of easy-to-use tools provided by vendors in the field. For instance, over the past 5 years across Forio Simulate, Forio Epicenter and the isee Exchange there have been 6,017 interactive simulators published. Yet, there is little information about guidelines or best practices on how to effectively build and use simulators for education and training.

Maier and Größler (2000) present a first step by suggesting to the field to stop using different terms to describe the same content. However, their taxonomy has not been accepted and the terms, microworld, management flight simulator and business simulator are still in use as terms which describe the same content (Größler et al. 2000; Morecroft and Wolstenholme 2007; Stouten et al. 2012, 2012; Sterman 2014). The use of these terms including newer ones like learning environment (Davidsen 2000), dashboard (Pruyt and Kwakkel 2012) etc. have become an integral part of the field's lexicon. Maier and Größler (2000) highlight this confusion caused by many different terms and they provide an extensive list of criteria for categorization, shown in Table 1. While the terminology does not provide clear definitions, the criteria does. Therefore, we present yet another terminology to connect with existent literature but use the criteria of Table 1, especially the category "functionality".

In this paper, we focus on game-oriented simulations as our main area of application. Thus, those simulators which make use of a fully completed SD model. In this way, we use the term 'simulator' to refer to a SD simulation model combined with a graphical user interface. This definition overlaps with Maier and Größler (2000) who suggest that an interface is an integral part of a simulation for learning support. However, they define a simulator as single user tool and use a different term – planning game – to refer to multi player. We argue that the terms single player/multi player are intuitively understandable.

In the next two sections, we present the experience of the authors and of their discussions and experiences with others in the field of building SD-based simulators and tools to build simulators for education and training. The description presents the existent state of most popular SD-based simulators.

Single player simulators

The distinction between single and multiplayer simulators is the top-level classification feature of our research. Within the category of single player simulators, we distinguish between two categories: analysis tools and management flight simulators. In analysis tools insight is distilled from reviewing the results of many simulation runs and comparing results across runs. Analysis tools are recognized most easily via their use of a single action to simulate the entire model and their heavy use of comparative graphs. In some cases, analysis tools will compile and display for the user cross run statistics. Analysis tools do not make use of information hiding, and therefore contain no 'twists' or big reveals. Analysis tools rely on learner-based discovery, and all policies are available for exploration, and assumptions are made clear and explicit. In certain instances, authors will also expose model assumptions for

experimentation as well. Published examples of this kind of simulator include: HealthBound, PRISM, C-Learn and T21 North America. Typically, analysis tools require models which contain a high level of detail complexity, and models must be very robust to changes in assumptions and decisions since users will tend to explore a much fuller spectrum of the state space of the model.

Underlying model	Human–computer interface	Functionality
Real-world domain	Chance of intervention while simulating	Number of users possible
Business	Discrete periods	Single person
Other	Simulation in one run	Multi person
Generality of model in regard to domain	Mode of users' input	Degree of integration
Special area of real world domain	Policy-oriented	Stand-alone simulation
Whole domain	Decision-oriented	Integration in computer-based environment
Structure	Mode of display	Main area of application
Feedback-oriented	Text	Modeling-oriented
Process-oriented (mostly without feedback)	Multimedia	Gaming-oriented
Behavior	Mode of Interaction	Use of teachers/ facilitators/coaches
Deterministic	Keyboard	Totally self-controlled learning
Stochastic	Mouse	Support by teacher/ facilitator/coach
Progress of time in simulation engine		Transparency of simulation model
Discrete		Black-box
Continuous		Transparent-box
Role of simulation model		Advancing of time
Active generation of decisions		Clock-driven
Clearing device for users' decisions		User-driven
Influence of external data		
With such influences		
Without such influences		
Domain of variables		
Integers		
Real numbers		

Table 1 - Criteria for categorization of computer simulations (Maier und Größler 2000)

Management flight simulators are the second category of single player simulators. They are most easily recognized through their use of a step by step mechanic for simulating the model. Time can be advanced directly by the user or indirectly via the use of a timer or other external (facilitator) controlled mechanism. In management flight simulators insight is distilled through experiences gained over the course of a single (or limited number) of simulation runs. This category of simulators is more like traditional games, with a strong real world relationship where the player is put into a role, the briefing contains a background story and context setting, players are given tasks and there is the possibility for twists, information hiding and obscuring. Management flight simulators are also characterized by the fact that player power is limited and not all decisions can be taken, there is no influence over assumptions and player action is highly constrained by real world limitations. Examples of this category of simulator include HBSP Pricing Simulation: Universal Rental Car, and any single player version of the Beer Game or Fishbanks.

There are simulators which contain elements of both analysis tools and management flight simulators. Typical hybrid simulations play like management flight simulators but have an element of scenario comparison after playing the simulator two or more times. Because of their step by step time advancement technique and the scenario comparison only after finishing all runs, we consider them as management flight simulators.

Multiplayer simulators

Classification within the multiplayer simulator category is more complex than single player. All commonly published multiplayer simulators exhibit the attributes and characteristics of management flight simulators as opposed to analysis tools. Time is advanced step by step, insight is distilled over the course of a single or heavily limited number of runs.

For the class of multiplayer simulators, we will use the terms role, team and facilitator. A role is a way of identifying what information and decisions a player has access to. Only one user can have each role on a team. A team is a group of players with roles who all interact together to create a run of the simulation model. A facilitator is a special role (super user) who can see the results and decisions of all roles and can (depending on the simulator design) affect the course of the game (exists in single player too).

Multi player simulators involve interaction and communication between the roles which interact with the underlying computational model. This means the key behaviors and outcomes of the system are driven by how the players interact and communicate with each other as opposed to purely from internal model dynamics. Within this classification, we only consider simulators where the predominant area of player interaction is within the setting of the simulator. We classify and group multiplayer simulators via their design decisions in the following 3 areas: (1) intra-team goal alignment, (2) team and role assignment and (3) time advancement technique.

The first attribute, intra-team goal alignment, refers purely to the incentives of the roles within a team. There are three forms: competitive, cooperative and hybrid. In the competitive case, each role is in competition with all other roles, likewise for cooperative cases all roles are incentivized purely by a common team goal. The hybrid case is where there is a common goal for all users, but individual incentives may create competition. The second attribute, team and role assignment refers to how teams are created and roles given to players on the team. There are two forms of team and role assignment, ‘prepared’ and ‘on the fly’. The third and final attribute, time advancement technique, refers to how the roles interact (or don’t interact) to advance the state of the simulator. There are four possible time advancement techniques: with a timer, by a facilitator, by a specific role or via a consensus decision.

A summary of the categorization is given in Figure 1.

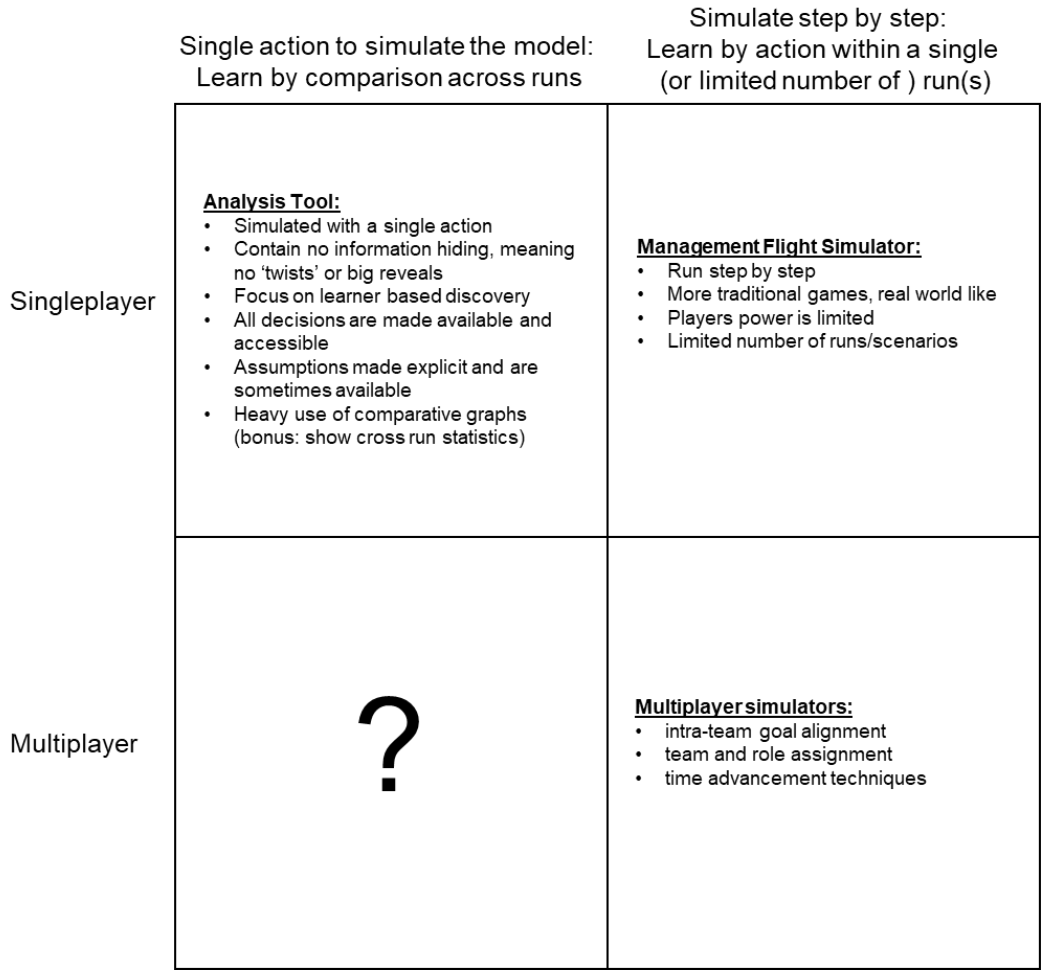

Figure 1 - Categories of Simulator Designs

Learning objectives and knowledge

To link learning objectives with simulator design, we need to explore and make clear our assumptions from the knowledge and learning fields. Scholars in these fields have so far not agreed upon one unified theory of how learning occurs, and there are in fact several coexistent theories (Shuell 1986; Schunk 2012). However, most theories have in common the over-arching idea that learning: has to do with acquiring knowledge and information such as facts and skills (Brown and Palincsar 1986), that it changes and/or updates existent knowledge, and finally that it provokes enduring change in behavior or in the capacity to behave differently (Shuell 1986; Schunk 2012).

Within the field of system dynamics, several authors emphasize the importance of involving decision makers in the whole modeling process to provide the most in-depth learning (Sterman 2000; Lane 1992; Rouwette, Etienne A. J. A. et al. 2002; Vennix 1996). However, it is not always possible to include all or even any stakeholders into the whole modelling process, especially when it comes to education and training. Alessi (2000) therefore distinguishes between building, and using simulation models for educational and training purposes. While the model building process is very well suited for knowledge discovery and transfer, it is not the correct choice for every project; in many cases, just the use of a simulation model in the form of a simulator serves as an effective means for learning. This idea is in line with other authors who argue that: (1) simulation model usage supports learning about dynamic

complexity, policy resistance and policy design (Sterman 2001). (2) That the use of simulation models influence the formation of mental models (Davidsen 2000) and (3) That using simulation models puts people in a position to learn about messy problems (Vennix 1996).

In the context of system dynamics, Sterman (2000) specifies learning as double-loop learning – a theory developed by Argyris (1976). Within this theory, learning consists of two parts: First is a feedback mechanism that closes the gap between the actual state of a system and the desired one. Decisions made may change based on the information received because of prior decisions. The second part of the theory states that decisions are derived from values, beliefs, strategies which are influenced by mental models, which themselves are altered by the information feedback discussed above (Sterman 2000).

Scholars have identified numerous types of knowledge. Like the definition of learning, there are several kinds of knowledge. For instance, Reif (1987) distinguishes between compiled, formal, interpretation, special, coherent and general knowledge. Reif and Allen (1992) add to this by including main interpretation, ancillary, case-specific, definitional, supplementary, entailed and concept knowledge. Each of these terms exists to structure the field of cognitive learning theory and to support specific theories within the field (Jong and Ferguson-Hessler 1996). Two well-known types of knowledge are declarative and procedural knowledge (Jong and Ferguson-Hessler 1996; Schunk 2012).

Declarative knowledge refers to knowledge about facts, objects and events (Jong and Ferguson-Hessler 1996; Jonassen et al. 1993). Declarative knowledge is referred to as knowledge about ‘What’ (Maier and Größler 2000) or ‘knowing that’ (Jonassen et al. 1993). It enables a person to describe objects (Jonassen et al. 1993). Procedural knowledge, on the other hand, refers to knowledge about ‘How’ (Maier and Größler 2000; Jonassen et al. 1993) – thus it contains actions, procedures and manipulations (Jong and Ferguson-Hessler 1996). It is often also referred to as a skill and the knowledge held by an individual of how to perform a specific set of actions (Rouwette and Vennix 2006). Further, this paper uses the term structural knowledge (Jonassen et al. 1993). It describes how concepts are interrelated and refers to knowledge about ‘Why’ (Jonassen et al. 1993; Maier and Größler 2000). All three types of knowledge have been used in the context of system dynamics (see e.g. Schaffernicht 2005; Maier and Größler 2000; Doyle and Ford 1998; Rouwette and Vennix 2006).

Declarative, procedural and structural knowledge fit within the context of double-loop learning since they influence decisions within the first loop and alter and create mental models within the second loop. Declarative knowledge is the basis for procedural knowledge (Jonassen et al. 1993) and procedural knowledge is the basis for structural knowledge. In terms of a system dynamics model, we argue that the transfer of declarative knowledge refers mainly to key variables, their measurements and their real-world equivalent. It encompasses the static description of a goal and problem, e.g. that a key variable is 80% lower than the expected value, as well as the basic context (the model is about a specific business unit, the user simulates as the role of a policy maker etc.). The use of a system dynamics model transfers procedural knowledge in form of information about behavior over time, polarities,

correlations, leverage points and the handling of the simulation. We understand procedural knowledge comprises much more since it is also defined as the knowledge of how to perform a cognitive activity (Schunk 2012) but as we analyze education and training using SD-based simulators, we focus on which knowledge can be transferred. Finally, structural knowledge encompasses cause-effect relations, feedback loops and their dominance and delays. We have summarized the connection between the knowledge types and a system dynamics model in Table 2.

Knowledge	Items in System Dynamics Model
Declarative (Know what, facts)	• Key variables and their connection to reality • Measurements • The Problem description (static) • The Goal • The context of the simulation
Procedural (know how, skills)	• Behavior over time • Porarities • Leverage Points • Correlations
Structural (know why, explain, derive, infer)	• Cause-Effect relations • Feedback Loops • Loop Dominance • Delays

Table 2 - Connecting Knowledge with a system dynamics model

A specific case is learning within a group of individuals. The types of knowledge and especially mental models, may refer to individuals or groups but as Kim (2009) explains, a mental model-like concept shared across a group is not well defined. Kim (2009) argues that all proposed concepts differ at least in two dimensions: First, there is the continuum of locus of thinking which refers to individual and entire group level. Second, there is the continuum of the form, which lies between a product, such as knowledge, or the process. For this reason, Kim (2009) uses the term “processor” which should not present a mental model-like concept in group setting but rather describes a black box that refers to a summary of different concepts.

This paper focuses on the outcomes of SD-based simulators. Therefore, we use product-oriented forms (different types of knowledge) according to Kim (2009). When talking about the learning outcome of multiplayer SD-based simulators we will use the term group mental model to indicate that some knowledge within the group is created.

Learning outcomes and simulator functionality

To understand when to use which kind of simulator and which knowledge is most likely to be transferred, we will discuss the advantages and disadvantages of the simulator functionalities identified in the second

section. We derive an overview about the presented classification of simulators and their connection to the different types of knowledge (declarative, procedural, structural) and if the simulator will stimulate group mental model building. This overview is shown in Figure 2 (page 18). We will start with two distinct types of single player simulators before we elaborate on multiplayer simulators.

Appropriate use of analysis tools

Analysis tools are most appropriate when there is a need to communicate information to a specialized audience that holds detailed knowledge, for example: consider 'students' that have extensive knowledge about a subject like managers, decision makers, implementers. These users typically possess a nearly complete declarative knowledge of the system, they are familiar with the subject matter, and well versed with the detail complexity contained within the simulation model but typically do not yet fully understand the dynamic complexity present within the real-world environment e.g. they need to develop better procedural and especially structural knowledge of the system. Analysis tools need a detailed briefing, so that users can acquire and familiarize themselves with all the required declarative knowledge to make sense of the system presented within the simulator. Therefore, analysis tools are often introduced via lecture or other traditional media tools to provide the necessary foundation for learning to occur.

Because analysis tools have as their goal the development of structural knowledge users typically desire full visibility into model assumptions and generally wish to have control over modification and manipulation of those assumptions. Users typically require control over all decision parameters so they may fully explore the simulation state space to design and develop robust policy interventions which hold up under the widest and most commonly accepted sets of assumptions which is key to the transition from procedural to structural knowledge e.g. the move from trial and error learning to true understanding.

Clearly, analysis tools are especially well-suited when learning should be discovery based. That means the key insights of the computational model are best understood as derivations from a base case. That means that policies or scenarios are tested. In simple English, if the results of the simulation are best understood as 'implementing policy X with assumptions A, B, C, drives key performance indicator Q up' then an analysis tool is probably the best design to deliver this knowledge. This is because analysis tools make heavy use of comparative graphs and tables to highlight the differences in potential 'What if…' scenarios and possible policies. We can extend this case to cover models in which the specific numeric values being reported by the simulation model are unreliable or are other non-physical and indexed values. This allows interface designers to still communicate the directionality and magnitude of change without having to focus on specific point value estimates. Results in these cases are typically presented not as absolute figures but rather as percent change from base case or even just (Good, Neutral, Bad).

Moreover, analysis tools should be used when the user is expected to connect existent knowledge with structural knowledge. That means when it is important for users to develop a strong structural understanding of the system, specifically understanding the difference between assumptions and policies. This becomes very important when assumptions are not fully shared or agreed upon across

the target audience. Analysis tools can be great consensus building tools used to communicate differences in mental models held by a variety of individuals. By classifying assumptions and allowing end-users to manipulate those assumptions interface designers enable skeptical end-users to get past their inherent distrust of the simulation model to generate policies, and to begin the process of simulation based structural knowledge accumulation. By establishing a formal separation between assumptions and policies for end-users, interface designers develop and curate their models' unbiased perception which is critical for simulation acceptance with hostile or skeptical users.

Furthermore, analysis tools should be used when it is necessary to be able to change assumptions, and discuss their importance in the context of KPIs. Developing robust policies requires the manipulation of assumptions and it is therefore critical to be able to change assumptions when the learning goals are based around policy development. The comparative nature of analysis tools complements and supports this mode of exploration.

Finally, the use of analysis tools is appropriate when the decision state space of the computational model is large. This is only if for the simple mechanical reasons of playing 100 runs of a game which require you to simulate step by step will be totally impractical within allocated time bounds.

Advantages of analysis tools

Typically, well-constructed analysis tools lend themselves to a deep exploration of subject matter and the development of procedural and structural knowledge from declarative knowledge. They present less of a bias because they are so much more transparent about their assumptions. Because of these benefits analysis tools tend to give users a more systemic understanding especially in cases with large decision state spaces, and therefore tend to be viewed as more power user oriented. As stated above analysis tools can work well when end-users are skeptical or openly hostile to simulation based learning, and are best used in cases where robust policy development is a key expected outcome.

Disadvantages of analysis tools

Typically, well-constructed analysis tools are perceived as dry and boring. They require much better understanding of the context by the end-user and therefore the development of strong briefing materials becomes critically important. The final disadvantage of analysis tools is that because they engender deep exploration they require robust, well tested models and typically require lots of detail complexity because end-users need to be able 'find themselves within the model'.

Appropriate use of management flight simulators

The most famous use of management flight simulators is training, specifically training users to perform a specific task with the aim to develop declarative and procedural knowledge about the underlying system. The more game-like nature of management flight simulators makes them much more suitable for training purposes: users get immediate feedback on decisions taken. Management flight simulators do not concentrate on developing deep structural understanding since the model structure tends to be

hidden. High engagement, and the real-world feel of these simulators enhance their ability to promote knowledge transfer from in-game to in world. Typically, models used for this case are highly deterministic, and there is an ahead of time known proper solution. The goal of these kinds of management flight simulators is to support and encourage a predetermined form of behavior bringing users from no knowledge of the system to having full declarative and procedural knowledge of the system with limited structural knowledge.

This is a core distinction from analysis tools. While analysis tools require in-depth briefing about declarative knowledge and produces structural knowledge about the system, management flight simulators require the briefing to include context and role definition, but most declarative knowledge is transferred during the use the management flight simulator. The debriefing, should then strive to reinforce the declarative and procedural knowledge and if so desired broach the subject of structural knowledge.

Further, we argue that management flight simulators are appropriate when user engagement is important. When done well management flight simulators are much more engaging than analysis tools because of the in-depth development of a story and context: first given in the briefing but then further developed over the course of using the management flight simulator. For the case where user engagement is critical, the importance of briefing materials, context setting, and role development become critical to the success of the simulator.

Additionally, we see management flight simulators fit to situations when users are completely hostile to any form of simulation (or model) based learning and therefore the model needs to be ‘hidden’. The gaming nature of management flight simulators makes it possible to completely hide the underlying simulation model. If end users are completely hostile to even the approach of simulation modeling it is possible to disguise and downplay the importance of the simulation model and just present the tool as a game, avoiding potential bias that some users may have against simulation based learning.

An important caveat to keep in mind when considering the use of management flight simulators is that key model insights must be revealed within the context of a limited number of runs. Management flight simulators by their very nature are simulated over time in a step by step fashion, therefore the learning must occur as a direct result of user action within the context of a limited number of runs. Typically, management flight simulators take anywhere between 5 and 60 minutes to run through completely so therefore it is unreasonable to expect that many simulation runs will occur.

Management flight simulators can be used as a test for understanding of a concept. Often, they are developed as a testing and evaluation tool for measuring end-user ability to manage a complex system, or perform a task in a dynamic and changing environment. Their real-world boundaries and typical pre-determined solutions make them ideal as testing and evaluation tools for the same reasons as management flight simulators are ideal for training.

Management flight simulators can also be used in groups to encourage competition among a group. When management flight simulators make use of heavy context setting and emphasize their more game like nature it is easy to encourage competition among a group of individuals by incentivizing them to achieve a better outcome on a certain performance indicator. The reason that management flight simulators work much better for this objective than analysis tools is that because of the unlimited nature of analysis tools users will eventually (and often quickly) fall into a trial and error based approach. Whereas with management flight simulators the limit on the number of runs to be done (both externally forced, and time based) requires a greater focus on learning about the problem and internally developing a consistent and successful strategy avoiding the pitfalls and brute force behavior associated with trial and error approaches.

Advantages of management flight simulators

Typically, well-constructed management flight simulators have much higher user engagement, and are highly real world realistic. Because computational models are hidden there is less questioning and it is easier to explain simulation logic. Users tend not to give nearly as much critical thought to the functionality and construction of the underlying simulation model as they do in analysis tools, and instead use that extra thought when properly motived to their performance and understanding of the specific task at hand. This is also reflected in the role of briefing. The briefing sets the role and the context and often the user does not necessarily need any specialized knowledge of the topic of the management flight simulator. Thus, less time for briefing is needed when compared to a similar audience using a comparable analysis tool. The final advantage to these tools is that the models required are typically much simpler, and are less robust across a wide variety of assumptions because only a limit decision state space can possibly be explored.

Disadvantages of management flight simulators

Typically, even well-constructed management flight simulators on their own do not develop a full structural understanding of the underlying simulation model, focusing on almost purely declarative and procedural knowledge acquisition. Knowledge gained is typically directly transferred to a highly similar real world event because of easy recognition, but not applied horizontally to other apparently dis-similar cases. Thus, to support the building of structural knowledge, a debriefing needs to be carefully planned. The hidden model structure can be revealed and assumptions and implications may be discussed. If not executed at the highest levels of quality including briefing and debriefing, management flight simulators can come off as patronizing, and hokey which negate all their engagement and realism based advantages. Especially, if the debriefing is done inappropriately, users will have learned to navigate the simulator but will not be able to connect their gained knowledge to real world applications.

Appropriate use of multiplayer simulators

We identified earlier three attributes which we use to characterize this class of simulators: intra-team goal alignment, team and role assignment and time advancement techniques. In the following sections, we will discuss the appropriate choices to make for each of the three attributes based on desired

outcomes. We argue that intra-team goal alignment has the most impact on learning outcomes. However, it is also important to discuss the other attributes to give a full picture of the simulators functionalities.

Attribute 1: Intra-team goal alignment

The choice of a simulators intra-team goal alignment strongly affects the transfer of procedural knowledge and the creation of a group mental model generated by the users. While competitive simulators encourage competition among all users and therefore motivate users to have the best performance, they are also used to teach about teamwork and leadership through anti-examples via a strong debriefing. Competitive simulators are used when the dynamics of the underlying model require competition and information hiding to be exhibited, e.g. imagine a game where if everyone collaborates everyone does better whereas if everyone backstabs everyone does worse. Competitive simulators can also be used to sort and rank individuals within a team.

Cooperative simulators build teamwork and leadership skills, and improve communication within the group by giving players the opportunity to work together and try out strategies in a relatively risk-free environment. These simulators tend to give users a better declarative and procedural understanding of a specific role's requirements. Cooperative simulators can be used to sort or rank groups of individuals together (as a team), or they can be used when solving the objective is hard and it requires many minds to perform the task.

Hybrid goal alignment simulators are the most real-world realistic and help to discover players selfish vs. altruistic tendencies via the use of competing team and role incentives. They are often used to develop and measure teamwork and leadership skills in a real world realistic environment. Ultimately goal alignment comes down to replicating real-world group dynamics and all forms of intra-team goal alignment besides hybrid are typically abstractions and simplifications of real-world scenarios.

In all multiplayer simulators participants develop deeper insight into the requirements of their specific role and with that procedural knowledge of their roles and a shared understanding of the context of the simulator. This gives hints to the requirements of the briefing and debriefing. Special attention must be paid for briefing multiplayer simulators. For hybrid or competitive multiplayer simulators, it is typical for a separate briefing to be developed for each role. In all cases the briefing should provide enough declarative knowledge for the role, the primary motivation, incentives and the context of the situation. Because of information feedback on decisions made, procedural knowledge is expected to be gained. Structural knowledge, however, is unlikely to be gained because users usually do not have enough access to model structure and assumptions and these simulators do not offer enough opportunity for experimentation with different strategies.

To encourage the development of structural knowledge authors must rely on debriefing materials and potentially even a paired analysis tools. In the competitive setting, every user is making decisions individually and thus the generated feedback is interpreted individually as well. Therefore, we argue that

in competitive multiplayer simulators (without considering the quality of the debriefing), the transfer of structural knowledge is lowest. For all multiplayer simulators groups of individuals are communicating and learning from each other. This communication comes in many forms either, directly (via chat, or talking) or indirectly through user action or inaction (e.g. setting a high price, 'forgetting' to fulfil a promise etc.) and this communication plays an important role in learning. Different interpretations of the information feedback are perceived and either during the game (or in the debrief) are discussed within the team and this drives the creation of group mental model, aligning individual mental models.

The debriefing is important for each multiplayer simulator. First, since competition is created, every game risks the creation of conflict. We suggest that all individuals should be given the opportunity to discuss their solution to any conflict generated over the course of the simulation to stimulate the growth of their individual mental model and to help facilitate the spread of that mental model across the team. Second, the underlying model should be discussed with the players to support the transfer of structural knowledge. Third, the individual mental models held by the players should be discussed to modify and solidify the group mental model. The shared experience will help the teacher to anchor knowledge and to refer to it. The teacher should be aware that in case of different roles, the game will serve as a boundary object.

Attribute 2: Team and role assignment

There are two forms of team and role assignment, prepared and on the fly. Simulators which use prepared team and role assignment have a facilitator who creates the teams and assign the roles to individuals on the team. When user's login to the simulator it immediately starts and there isn't any choice by the player about the role they are going to play or the other members of their team. The second form of team and role assignment is on the fly where users create their own teams and pick roles on those teams based on what is currently available. In this case users login, examine teams in progress of forming, see what roles are available and either choose to join an in progress forming team or start their own new team with any role of their choice. This form of team and role assignment is required when the players aren't known in advance.

Prepared team and role alignment is used when all the individuals in the audience are known ahead of time. This form of team creation works well when dealing with communication issues among a specific set of individuals and it gives facilitators fine grained control over group dynamics. Prepared team and role assignment can be used to create a copy of a real world situation; thus, individuals are typically assigned to roles that fit their knowledge and strengths and teams will represent groups of individuals that know each other, work together or have any kind of relation, e.g. individuals that share a language when all others are from different countries. Although in some cases assigning roles like that is not ideal, and it is best to assign players to unfamiliar or conflicting roles in order to force individuals to gain perspective and develop new communication techniques. Prepared team and role assignment has the capacity for creating the most extreme group dynamics and modes of behavior. Finally, from a technical perspective it is the easiest implement.

On the fly team and role assignment is the only choice for mass market simulators, or any simulators where the audience is not known ahead of time. It's the hardest to implement from a technical perspective, but works well when users are spread out in time and space and therefore it is not possible to predict when people will be available to devote time to the simulator.

Team and role assignment can have a significant influence on learning outcomes, especially when teams are prepared to create conflict or exhibit other more extreme behaviors within group dynamics. While every individual will have the opportunity to learn declarative and procedural knowledge by using the simulator, the transfer of structural knowledge and the creation of a group mental model is highly dependent on the team composition and communication. Groups of individuals that know each other and that have roles of their knowledge domain may be more effective in deriving decisions but may be prone to group think or domain specific thinking (e.g. déformation professionelle). If teams are assembled of people that do not know each other and roles are assigned according to weaknesses of knowledge domains other than held by the individual, communication barriers may play an important role. Further, domain specific terminology may extend the time for playing the game.

Prepared team and role assignment should therefore be planned in detail. The explanation of roles and tasks are a big part of the briefing. Especially, communication barriers (like unknown terms, special rules of interaction) should be tackled and declarative knowledge should be delivered in the beginning. The debriefing should treat especially the group processes and modify group mental models.

Attribute 3: Time advancement techniques

There are four forms of time advancement used in multiplayer simulators. Time can be advanced by a single role e.g. a single individual on the team is the only player who can advance time. Time can be advanced automatically using a timer, there are two sub forms of timers, the first are timers which operate relatively quickly e.g. teams have 5 minutes to make decisions before time moves to the next time step; the second are long running simulators which run in 'real' time where each in simulator day is equal to a day of real world time (or nearly so). The third form of time advancement is via consensus where all roles must agree to advance time together. This form of time advancement is often paired with a timer to keep the simulator moving with at least some predictability. The fourth and final form of time advancement is via a facilitator where a professor, or some other individual external to the team is responsible for moving time forward.

Single role time advancement is the simplest to implement from a technical perspective and establishes a leader or dominant role on the team shaping the shared knowledge, i.e. group mental model created by playing the simulator. Single role time advancement is not advisable in the purely competitive setting since it would give one player an advantage over all others. In cooperative and hybrid simulators, it is applicable and forces communication between the other players and the player who has the power to advance time. This time advancement technique can create long games if not paired with a timer and players are obsessive.

Timer based time advancement is also straightforward to implement from a technical perspective and establishes a non-biased timing and creates games which last for a predictable amount of time. When using timers with a long duration it is possible to create simulators where users do not have to be logged in simultaneously. Short timers create pressure to make decisions and this has a loosely correlated positive impact on communication when paired with cooperative or hybrid goal alignment simulators.

Consensus based time advancement is the hardest to implement from a technical perspective. It relieves the pressure to make decisions (except when combined with a timer) and it has a strong positive correlation with communication and lets the team develop its own leadership strategy. This technique nurtures communication, and gives everyone a chance to get their say. This form of time advancement works well with any goal alignment.

Facilitator based time advancement is easier to implement then consensus based time advancement and is on the same level as advancing via a timer. It gives the facilitator precise control over pacing which lets them create or relieve stress as necessary. It works well with any form of goal alignment.

Outside of pressure, and leadership dynamics discussed above, we do not see a large influence of time advancement techniques on knowledge transfer. Authors should therefore keep in mind that the time spent in the simulator can be regulated depending on the time advancement, but that the acquisition of knowledge and development of communication skills will need time and repetition.

Discussion and Implications

Our assumptions

After having derived the distinct contributions of SD-based simulators, we want to reflect on the assumptions and discuss our findings. We started this paper by presenting the situation of teaching a subject with the use of a SD-based simulator. Current research, however, looks at how to teach system dynamics (Richardson 2014a, 2014b, 2014c; Schaffernicht and Groesser 2016) with models, modelling, simulators and simulation. Our paper connects to current research since the teaching methods may include simulators. Richardson (2014b), for instance, presents the exploration of an existent model as basis for becoming an experienced modeler. In our case, the model would have been combined with graphical user interface to become a simulator. Further, Schaffernicht and Groesser (2016) point out that the dynamic reasoning and model analysis are crucial skill sets in learning system dynamics. We argue that some of the skills listed in the competence framework can be achieved by using simulators. In short, our paper connects to current research about teaching system dynamics but is not only restricted to it since other topics can be taught by using SD-based simulators without teaching system dynamics. Therefore, we place our paper not only in the field of system dynamics but also in different knowledge domains which may use the representation of the topic to be taught by a system dynamics model.

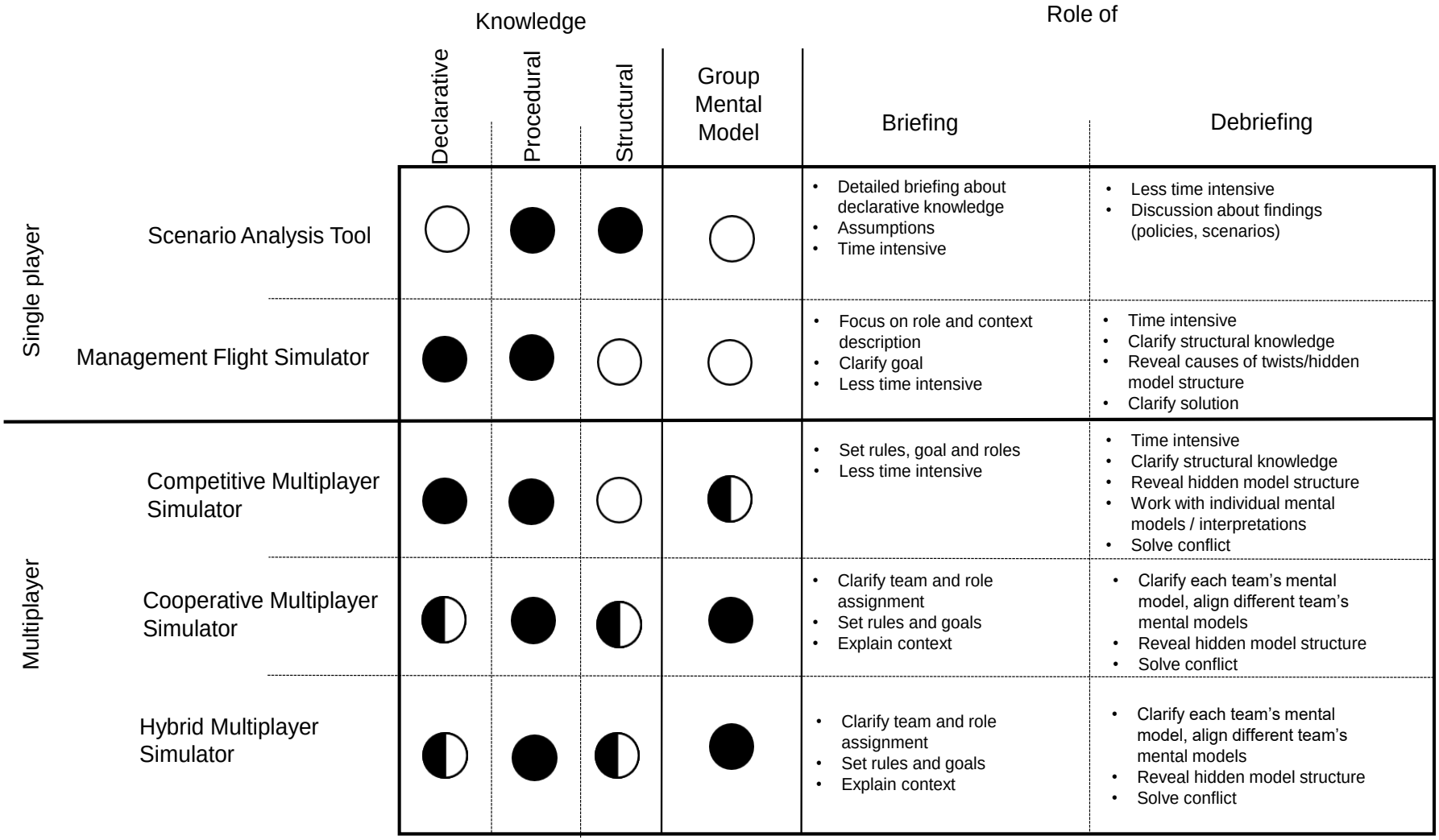

Figure 2 - Learning outcomes and simulator functionality

With the purpose of presenting the most famous functionalities, we present five different designs: analysis tools, management flight simulators, competitive, cooperative and hybrid multiplayer simulators. By introducing these terms, we thwarted the aim of Maier and Größler (2000) to use common terms. We agree that a vast variety of terms may create confusion and hinder ongoing research. We connected to the proposed taxonomy but we feel the need to extend it since the division into subcategories based on time advancement plays an important role for knowledge transfer. We argue that after the publication of the taxonomy, scholars of the field did not switch to the suggested terminology. We do not believe that suggestions by Maier and Größler (2000) are not accepted but rather that something is missing to stimulate further research. This is why we try to extend the taxonomy by adding the connection to learning outcomes.

The presented categories are not a result of a scientific literature search but are formulated based upon the experience of the authors in discussion with others across the SD field who have been working on building simulators and tools to create simulators over the past decade. A rigorous collection and deconstruction with analysis of the most famous simulators is missing and encouraged. Especially a comparative study of the functionalities and the learning outcome may shed new light on the effectiveness of simulators and updates to our taxonomic system.

In this paper, we connect the functionality of simulators with learning outcomes. We treat the creation and transfer of knowledge as learning outcomes. Specifically, we look at declarative, procedural and structural knowledge. That is an immense reduction of the variety of types of knowledge existent in the literature. Nonetheless, these three types of knowledge are well-accepted concepts (see e.g. Schaffernicht 2005; Maier and Größler 2000; Doyle and Ford 1998; Rouwette and Vennix 2006). We argue that every type of knowledge serves its theory where the term is embedded in and these terms are well-accepted in constructivist learning theories, the area in which double-loop learning is embedded. Therefore, we believe that the use of these knowledge types is reasonable. However, by using the three knowledge types and applying them to SD-based simulators, we reduce these concepts. By definition, declarative knowledge does not only contain facts but also certain beliefs and domain-specific methods; and procedural knowledge contains the skill of applying methods to certain problems (Schunk 2012). We reduce those definitions since we answer the question which knowledge is created or transferred by using a SD-based simulator. We conclude the results shown in Table 2 (page 8). We think it is reasonable to connect those items of a system dynamics model to the indicated knowledge types. We acknowledge that the knowledge types contain much more than only the indicated items – however, since this paper is no experiment on learning processes we only focus on these product-oriented reduced knowledge types. The processes that lead to the acquisition of knowledge are of high importance but are not a subject of this paper.

Last, we use the term “group mental model” without proper definition. We rather refer to it as a blurry term that describes that something is happening to a group of people that involves learning, discussion, decision making. Using this term, we follow Kim (2009) suggestion that such a term contains too many different concepts that more research is needed on constructs of group learning. For us group mental

models contain the knowledge of the individuals, thus the aforementioned declarative, procedural and structural knowledge, sometimes it is shared and agreed sometimes not. We intendedly did not focus on the discussion of what exactly it means but we want to raise attention that something is happening to the group of individuals and we think that the extent to what something is happening differs depending on the functionality. We consequently use this term to relate to games.

Implications of the results of our research

In this section, we want to present our findings chronologically and discuss them.

As mentioned before, we introduced the new terms analysis tool, management flight simulator and multiplayer simulators to the taxonomy developed by Größler et al. (2000) . We regard these terms as an extension of the taxonomy since we used the criteria for classification to define them. Though the aim of the taxonomy was to clarify the use, we increased the amount of terms used. But somehow, we believe that the taxonomy did not intend to limit research on simulators but rather clarify. It helped us to categorize existent simulators and therefore our contribution is a little extension. It shows that further clarification is needed to derive the use of simulators and thus conduct research on the effectiveness of simulators to support learning.

By the categorization presented in Figure 1 (page 6), we show that there seems to be a lack of multiplayer simulators that simulate on one action and generate knowledge through comparative analysis. The question however arises what kind of simulator might that be? This category implies that every user has all information and they would work together to immediately create new policies based on the behavior of all participants. With full information, it seems that such a design might connect to game theory since the behavior of the different teams/users is uncertain but the basis for model outcome. Competition does not seem to work in such a setting since one action will create a scenario. The C-Roads simulator might be described as a simulator close to this category since it simulates on one action and scenarios are compared. However, C-Roads is an analysis tool since it does not have multiple users simultaneously interacting to create those scenarios. We believe that the functionalities of such a class of tools is probably being handled today via the use of analysis tools where users can share runs with each other and build new scenarios by taking policy changes from other users. With that said, we encourage research into the subject of how best to fulfil the learning objectives of analysis tools within a multiplayer context.

We connect learning outcomes with simulator functionality. We show that analysis tools are the most appropriate tools to transfer structural knowledge. Management flight simulators are more appropriate when the focus of training and teaching lies on procedural and declarative knowledge. Multiplayer simulators focus on team building, communication and group mental models. The summary of our research is shown in Figure 2. By using the empty circles, we do not rule out knowledge transfer of a certain kind. We argue that the functionality does not seem appropriate to deliver this knowledge or that the briefing / debriefing should deliver this knowledge if desired.

The classification scheme presented is useful for both traditional educators as well as consultants when they are constructing and using simulators. It is our hope that all builders of simulators will be able to make the key design decisions for their tools in the way which best supports their learning objectives based on the kinds of learning they want to encourage. This extension of the taxonomy of computer simulation to support learning ought to influence software vendors to produce tooling which support all varieties of simulators described. Especially for multiplayer simulators there are significant technical challenges. For analysis tools, for instance, we highlight the importance of displaying percent change metrics, and other cross run statistics. Thus, our research outcome can also be used by software vendors as an aid to help novice authors to design the basic form of their simulation via in software based wizards which generate page templates to avoid 'blank page syndrome'.

While all high-quality simulators require briefing and debriefing materials, this system helps authors to understand the kinds of knowledge which must be embedded into those materials to encourage the full spectrum of learning (across all knowledge types). E.g. for management flight simulators and multiplayer learning environments to maximize the chances for structural knowledge creation to take place debriefing materials are critically important because the simulator itself is typically deficient in its ability to create this kind of knowledge in its users. For analysis tools the briefing and introductory materials must make sure that users have a good handle on the declarative knowledge contained within the system because without that, the structural and procedural learning which is usually the purpose of building such systems becomes out of reach.

Conclusion

The motivation behind our research is to promote and support learning about complex dynamic systems. There is wide agreement in the system dynamics field that the best way to promote the deepest learning is to involve key decision makers in the whole model-building process, but this is not always possible. Hence, key decision makers and students are confronted with ready-made system dynamics models for interaction. Consequently, promoting learning via SD-based simulators is important, and therefore the design of those simulators is critical. The presented extension of the taxonomy makes it easier for simulator authors to create tools which meet their learning objectives and to promote deeper learning.

We categorize simulators first on the number of player interacting simultaneously to generate a single run of the simulation model. For single player simulators, we then define two categories. Category 1: Analysis tools are simulators that are characterized by simulating with a single action, and make heavy use of comparative graphs and tables to support learning across a series of runs. Analysis tools are best used to impart procedural and structural knowledge of the system to users and require extensive declarative knowledge of the system to make use of. Category 2: Management flight simulators, on the other hand, are simulators which are characterized by their step by step time advancement technique and lack of comparative graphs and tables. They support learning that happens over the course of a single or limited number of runs. Management flight simulators are best used to impart declarative and procedural knowledge of a system to their users.

The classification of multiplayer simulators is much more complex. First, multiplayer simulators are highly related to management flight simulators imparting declarative and procedural knowledge best, and tend heavily towards supporting knowledge creation over the course of a single or limited number of runs. We have noticed that there are no commonly known simulators which are multiplayer which would fall into a category more closely matching the analysis tool where learning occurs over the course of multiple runs. Therefore, we categorize these simulators based on three attributes: intra-team goal alignment, team and role assignment technique, and time advancement technique. Through design decisions made by simulator authors on each of these attributes we can generalize about learning objectives and outcomes which are useful to new simulator authors getting started with new work.

Finally, through the paper, we linked learning outcomes in form of the knowledge types (declarative, procedural and structural), the creation of a group mental model and the role of briefing and debriefing to the functionality of the simulators. We believe that this link is critical for the effective development of useful and productive SD based simulators to support learning in education and training environments.

Acknowledgements

We thank the members of SDS Peer Mentoring Group in which this research was presented. The fruitful discussion enabled this structure and made clear what precisely we want to say.
Further, the Onlinekolloquium was very forthcoming with intense feedback, which we used to develop the figures presented to even better support our arguments.
Finally, we also thank Forio and isee systems for providing us with the number of simulators published to their online platforms.

References

Alessi, S. (2000): Designing Educational Support in System-Dynamics-Based Interactive Learning Environments. In: Simulation & Gaming 31 (2), S. 178–196. DOI: 10.1177/104687810003100205.

Argyris, C. (1976): Single-Loop and Double-Loop Models in Research on Decision Making. In: Administrative Science Quarterly 21 (3), S. 363–375.

Bakken, Bent; Gould, Janet; Kim, Daniel (1992): Experimentation in learning organizations A management flight simulator approach. In: European Journal of Operational Research 59 (1), S. 167–182. DOI: 10.1016/0377-2217(92)90013-Y.

Brown, A. L.; Palincsar, A. S. (1986): Guided, cooperative learning and individual knowledge acquisition. Cambridge, Mass.: Bolt Beranek and Newman Inc. (Technical report no. 372).

Davidsen, P. I. (2000): Issues in the Design and Use of System-Dynamics-Based Interactive Learning Environments. In: Simulation & Gaming 31 (2), S. 170–177. DOI: 10.1177/104687810003100204.

Davidsen, P. I.; Spector, J. M. (2015): Critical Reflections on System Dynamics and Simulation/Gaming. In: Simulation & Gaming 46 (3-4), S. 430–444. DOI: 10.1177/1046878115596526.

Doyle, James K.; Ford, David N. (1998): Mental models concepts for system dynamics research. In: System Dynamics Review 14 (1), S. 3–29. DOI: 10.1002/(SICI)1099-1727(199821)14:1<3::AID-SDR140>3.0.CO;2-K

Gary, Michael Shayne; Wood, Robert E. (2011): Mental models, decision rules, and performance heterogeneity. In: Strategic. Management Journal 32 (6), S. 569–594. DOI: 10.1002/smj.899.

Gary, Michael Shayne; Wood, Robert E.; Pillinger, Tracey (2012): Enhancing mental models, analogical transfer, and performance in strategic decision making. In: Strategic Management Journal 33 (11), S. 1229–1246. DOI: 10.1002/smj.1979.

Größler, A.; Maier, F.; Milling, P. (2000): Enhancing learning capabilities by providing transparency in business simulators. In: Simulation & Gaming 31 (2), S. 257–278.

Größler, Andreas (2004): Don't let history repeat itself - methodological issues concerning the use of simulators in teaching and experimentation. In: System Dynamics Review 20 (3), S. 263–274. DOI: 10.1002/sdr.286.

Hsu, E. (1989): Role-Event Gaming Simulation in Management Education A Conceptual Framework and Review. In: Simulation & Gaming 20 (4), S. 409–438. DOI: 10.1177/104687818902000402.

Jonassen, David H.; Beissner, Katherine; Yacci, Michel (1993): Structural knowledge Techniques for representing, conveying, and acquiring structural knowledge. Hillsdale, NJ: Erlbaum.

Jong, Ton de; Ferguson-Hessler, Monica G.M. (1996): Types and qualities of knowledge. In: Educational Psychologist 31 (2), S. 105–113. DOI: 10.1207/s15326985ep3102_2.

Kim, Hyunjung (2009): In search of a mental model-like concept for group-level modeling. In: System Dynamics Review 25 (3), S. 207–223. DOI: 10.1002/sdr.422.

Lane, David C. (1992): Modelling as learning A consultancy methodology for enhancing learning in management teams. In: European Journal of Operational Research 59 (1), S. 64–84. DOI: 10.1016/0377-2217(92)90007-V.

Langton, N. H.; Addinall, E.; Ellington, H. I.; Percival, F. (1980): The Value of Simulations and Games in the Teaching of Sciennce. In: European Journal of Education 15 (3), S. 261. DOI: 10.2307/1503242.

Machuca, J. A. D. (2000): Transparent-box business simulators: An aid to manage the complexity of organizations. In: Simulation & Gaming 31 (2), S. 230–239.

Maier, Frank H.; Größler, Andreas (2000): What are we talking about? A taxonomy of computer simulations to support learning. In: System Dynamics Review 16 (2), S. 135–148. DOI: 10.1002/1099-1727(200022)16:2<135::AID-SDR193>3.0.CO;2-P.

Morecroft, John; Wolstenholme, Eric (2007): System dynamics in the U.K A journey from Stirling to Oxford and beyond. In: System Dynamics Review 23 (2-3), S. 205–214. DOI: 10.1002/sdr.368.

Moxnes, E. (1998): Not Only the Tragedy of the Commons: Misperceptions of Bioeconomics. In: Management Science 44 (9), S. 1234–1248.

Moxnes, Erling; Jensen, Lene (2009): Drunker than intended: misperceptions and information treatments. In: Drug and alcohol dependence 105 (1-2), S. 63–70. DOI: 10.1016/j.drugalcdep.2009.06.012.

Pruyt, E.; Kwakkel, J. H. (2012): A Bright Future for System Dynamics: From Art to Computational Science and Beyond. In: E. Husemann und D. C. Lane (Hg.): 30th international conference of the System Dynamics Society 2012. St. Gallen, Switzerland, 22 - 26 July 2012. Red Hook, NY: Curran.

Reif, F. (1987): Instructional design, cognition, and technology Applications to the teaching of scientific concepts. In: Journal of Research in Science Teaching 24 (4), S. 309–324. DOI: 10.1002/tea.3660240405.

Reif, Frederick; Allen, Sue (1992): Cognition for Interpreting Scientific Concepts A Study of Acceleration. In: Cognition and Instruction 9 (1), S. 1–44. DOI: 10.1207/s1532690xci0901_1.

Richardson, George P. (2014b): "Model" teaching II Examples for the early stages. In: System Dynamics Review 30 (4), S. 283–290. DOI: 10.1002/sdr.1520.

Richardson, George P. (2014c): "Model" teaching III Examples for the later stages. In: System Dynamics Review 30 (4), S. 291–299. DOI: 10.1002/sdr.1524.

Richardson, George P. (2014a): “Model” teaching. In: System Dynamics Review 30 (1-2), S. 81–88. DOI: 10.1002/sdr.1512.

Rouwette, Etiënne A. J. A.; Größler, Andreas; Vennix, Jac A. M. (2004): Exploring influencing factors on rationality A literature review of dynamic decision-making studies in system dynamics. In: Systems Research and Behavioral Science 21 (4), S. 351–370. DOI: 10.1002/sres.647.

Rouwette, Etiënne A. J. A.; Vennix, Jac A. M. (2006): System dynamics and organizational interventions. In: Systems Research and Behavioral Science 23 (4), S. 451–466. DOI: 10.1002/sres.772.

Rouwette, Etienne A. J. A.; Vennix, Jac A. M.; van Mullekom, Theo (2002): Group model building effectiveness A review of assessment studies. In: Syst. Dyn. Rev. 18 (1), S. 5–45. DOI: 10.1002/sdr.229.

Salas, Eduardo; Wildman, Jessica L.; Piccolo, Ronald F. (2009): Using Simulation-Based Training to Enhance Management Education. In: Academy of Management Learning & Education 8 (4), S. 559–573. DOI: 10.5465/AMLE.2009.47785474.

Schaffernicht, M. (2005): Are you experienced? – a model of learning systems thinking skills. In: J. D. Sterman und N. D. Repenning (Hg.): Proceedings of the 23rd. International Conference of the System Dynamics Society, July 17-21 2005, Boston, Massachuesetts, US.

Schaffernicht, Martin F. G.; Groesser, Stefan N. (2016): A competence development framework for learning and teaching system dynamics. In: System Dynamics Review 32 (1), S. 52–81. DOI: 10.1002/sdr.1550.

Schunk, Dale H. (2012): Learning theories An educational perspective. 6. ed. Boston, Mass.: Pearson.

Shuell, T. J. (1986): Cognitive Conceptions of Learning. In: Review of Educational Research 56 (4), S. 411–436. DOI: 10.3102/00346543056004411.

Spector, J. M. (2000): System Dynamics and Interactive Learning Environments Lessons Learned and Implications for the Future. In: Simulation & Gaming 31 (4), S. 528–535. DOI: 10.1177/104687810003100406.

Sterman, J. D. (2000): Business Dynamics Systems Thinking and Modeling for a Complex World. Boston: Irwin/McGraw-Hill.

Sterman, J. D. (2001): System Dynamics Modeling Tools for Learning in a Complex World. In: California Management Review 43 (4), S. 8–25.

Sterman, John (2014): Interactive web-based simulations for strategy and sustainability The MIT Sloan LearningEdge management flight simulators, Part I. In: System Dynamics Review 30 (1-2), S. 89–121. DOI: 10.1002/sdr.1513.

Stouten, Hendrik; Heene, Aimé; Gellynck, Xavier; Polet, Hans (2012): Learning from playing with microworlds in policy making An experimental evaluation in fisheries management. In: Computers in Human Behavior 28 (2), S. 757–770. DOI: 10.1016/j.chb.2011.12.002.

Vennix, J. A. M. (1996): Group Model Building: Facilitating Team Learning Using System Dynamics. Chichester: Wiley.