TECH REPORT

Exploratory Didactics by Plug and Play in Prolog

ADICOM® SOFTWARE KG
Frauentorstraße 11
99423 Weimar

Email software@adicom-group.de
Tel +49 (0) 3643 85594-0

KLAUS P. JANTKE
RAINER KNAUF
OKSANA ARNOLD
JÜRGEN CLEVE

Version 1.00
30.04.2019

ÖFFENTLICH / PUBLIC

01-2019

Klaus P. Jantke
ADICOM Software KG
Frauentorstr. 11
99423 Weimar
Germany

Rainer Knauf
Ilmenau University of Technology
P.O. Box 10 05 65
98684 Ilmenau
Germany

Oksana Arnold
Erfurt University of Applied Sciences
Altonaer Str. 25
99085 Erfurt
Germany

Jürgen Cleve
Wismar University of Applied Sciences
P.O. Box 1210
23952 Wismar
Germany

Impressum

ADICOM TECH REPORT

ISSN (Print) 2627-0749
ISSN (Online) 2627-0757

ADICOM Software KG
Frauentorstr. 11
99423 Weimar

info@adicom-group.de
www.adicom-group.de

Geschäftsführer: Christian Hölzer
AG Jena: HRA501731

Cover Page Design: Robert Krause
Picture Credits:
World Map 45° Lines Vector
Author: pajhonka
https://www.vecteezy.com/map-vector/24225-world-map-45-lines-vector
Creative Commons (BY)

Contents

Foreword

This is the authors' first written version of their contribution to ISIP 2019, an invitation-based scientific meeting named the 13^{th} International Workshop on Information Search, Integration, and Personalization, May 2019, Heraklion, Crete.

One of the authors' intentions is to provide the workshop audience with a more detailed and in-depth information going beyond the limits of a 20 to 25 minutes presentation including the presentation slides.

However, the focus of the present publication is kept narrow to allow for a possibly following contribution to the planned volume of ISIP 2019 selected papers.

With respect to digital technologies, the authors aim at the introduction, explanation, and propagation of what they call *plug & play* within their original framework of *digital storyboarding* including *storyboard interpretation technology.*

The underlying research and development is motivated by the known vagueness and ambiguity of concepts in the humanities, especially in the educational sciences, that are due to the subject to investigation exceeding in complexity all other subjects of science and technology.

1 Plug & Play – the Touch and Feel

In this section, the authors put the cart in front of the horse. The technicalities to come are quite tedious. Therefore, it is deemed desirable to know in advance where the efforts might lead.

In technology enhanced learning, even vague didactic concepts are getting a precise syntactic appearance. So-called pedagogical patterns as discussed in [Ped 2012] may be rewritten precisely in varying ways [Jan 2013b]. Formal methods allow for an explication of substantially different alternatives. It is frequently unclear whether or not authors uttering vague ideas are aware of the largely varying precise implementations. Furthermore, there are cases in which different educators are in favor of different interpretations preferring the one or the other implementation.

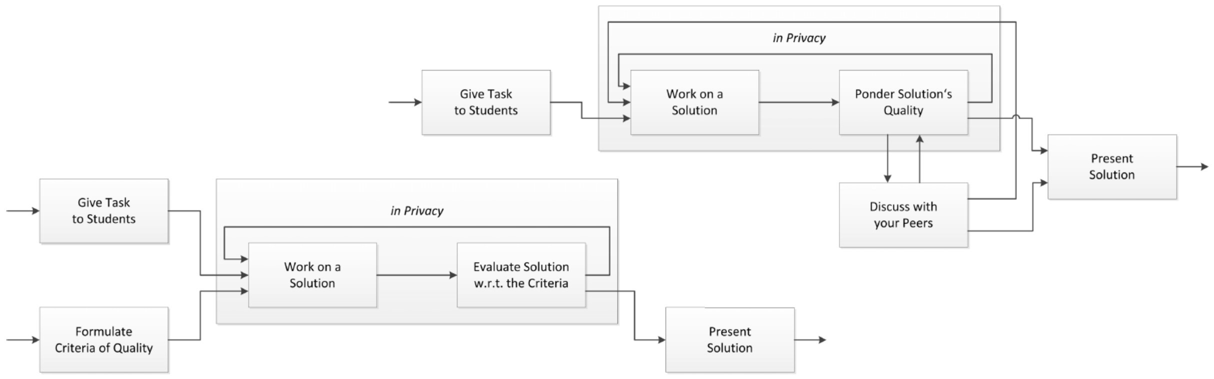


Figure 1: Pedagogical Pattern "Built-In Failure" [Ped 2012] Interpreted Formally [Jan 2013b]

By way of illustration, consider a pedagogical principle discussed in [Ped 2012] and named "Built-In Failure". Due to the ambiguity of language, the original circumscriptions allow for a variety of interpretations. Does the principle include the formulation of criteria of success? Does the learning process give space to a communication with co-learners? Answers to only these two sample questions may result in four different interpretations and – in technology enhanced learning – in four different implementations. Figure 1 above shows two of the interpretations that occur as figures 4 and 5, respectively, in [Jan 2013b].

Assume some digitalized learning environment in which processes are formally described by graphs as above. Within a more comprehensive learning scenario, when it comes to an educational situation to invoke the built-in failure principle, which of the varying variants shall be selected? The *plug and play* principle and technology enables educators to "plug in" one of the available variants and see what happens.

On the one hand, one may imagine educators to prefer the variant in which learners get some criteria of success to enable them to ponder the quality of their own intermediate solutions (lower left graph in figure 1, figure 4 in [Jan 2013b]).

On the other hand, when more social interaction is deemed important, educators may prefer the variant in which a priori criteria of success are dropped to encourage communication in the peer group of learners (upper right graph in figure 1, figure 5 in [Jan 2013b]).

But does it really work to invoke the second variant? Educators are at a strife. Plug & play is the approach and the methodology to answer this question and, even more important, to arrive at educational settings that are personalized and adaptive to the largely unforeseeable learners' needs and desires. The authors' *plug and play* is a principle and a technology of adaptivity and personalization, especially in educational settings.

One may check out automatically, whether of not the communication cycles really take place.

2 Educational Gamification as Transformation

The authors refrain from an in-depth discussion of the gamification concept and direct the interested reader to [AJ 2018][1], esp. section 2 on page 3. An even more comprehensive discussion, but in German, may be found in [Jan 2018b][1].

Essentially, gamification is *not*, as Deterding et al. claim, "the use of elements of game design in non-game contexts" [DOS+ 2011] (page 2; very similarly, [DDKN 2011], page 10). Bogost is calling opinions like that "bullshit" and frankly names the protagonists "bullshitters" [Bog 2011]. Bogost is basically right, because the usage of elements such as score points, badges, and rankings by themselves do not establish any exciting experience of game play.

The ultimate intention of gamification is to make available the fascinating attractiveness, the affectivity, and the effectivity of game play for serious purposes such as learning and training. As the authors of [AJ 2018], preface, p. 1, put it, "imagine – if successful – teaching, learning and training offers that are addictive. Learners always want more and learn more, trainees cannot stop to train, and those who learn and train, form communities on the internet where they exchange their experiences and successes in acquiring knowledge and sharing skills, as well as inspiring others to participate. Imagine, textbook publishers producing educational materials of an addictive nature and where school becomes our children's favorite venue."

To keep it short, educational gamification (EG) aims at the facilitation of playful experiences being attractive, affective, possibly even addictive, and practically effective for learning and training. EG means "the transformation of given learning or training material and/or educational environment into a form that bears the potential of playful experiences that are likely to unfold when humans accept to engage" ([AJ 2018], section 4.1, page 5).

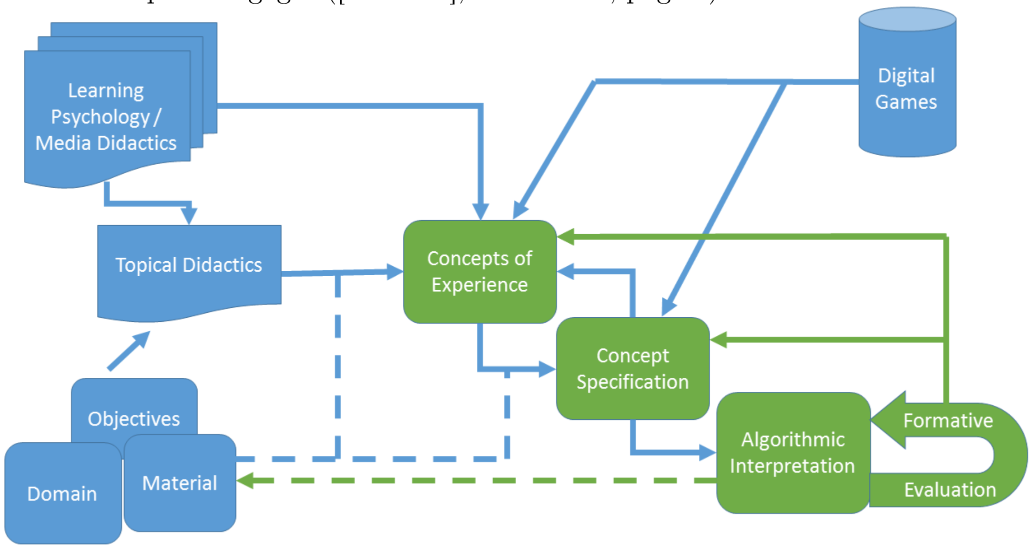


Figure 2: Gamification as Transformation according to [Jan 2018a][1]

The design and the description of anticipated game experiences form the central task of the gamification process. [FBJ 2015][2] exemplifies the transformation process in an application case.

Following [JK 2005], specifications are thoroughly digital and, therefore, may be subject to algorithmic interpretation (see section 3.3 below).

[1]The source is freely available in the internet on ResearchGate.

[2]This conference paper, by the way, is the first author's most frequently read publication on ResearchGate with 7,575 reads by April 30, 2019.

3 Introduction into the Underlying Technology

The authors' original storyboarding concept has been introduced by Jantke & Knauf in [JK 2005]. This approach originates from Arnold's conceptualization of dynamic plan generation [Arn 1996]. It has been used for a quite large variety of projects where the formalisms are of considerable value (see [AJV$^+$ 2010], e.g.).

To keep the present report self-contained, the details are introduced below, but stripped to the essentials.

3.1 Digital Storyboarding

Storyboarding means *the organization of experience* [JK 2005], in general, and the anticipation and organization of playful experiences, in particular, that bear the potential to unfold when human beings are willing to engage. Emphasis is put on wrapping didactic concepts into game playing experience [KJ 2014].

The perspective of *layered languages of ludology* exhibits the power as well as the necessity of varying levels of design [Len 2009] based on the eralier introduction of the key principles [Jan 2006]. This allows for top-down design, for buttom-up design as well as for largely varying dovetailings of both methodologies.

Some approaches to didactics are on a high level of abstraction [Fle 1996], others are on several difficult to specify medium levels [Ped 2012], whereas digitalization projects cannot be reluctant to fine granularity [KJ 2014].

At a first glance, a storyboard seems to be a graph. A closer look reveals that there are graphs of different granularity – an issue of layered languages of ludology. Conceptually, a storyboard is a *hierarchically structured family of graphs*. To allow for unambiguous graph substitution, it is assumed that graphs have well-defined input and output nodes, a concept named *pin graph* in the topical literature [HLW 1992, Arn 1996].

A *storyboard* is a hierarchically structured controlled family of pin graphs $\mathcal{F} = [\{\mathcal{G}_i\}_{i=1,\ldots,k}, c]$ where every pin graph $\mathcal{G}_i$ of the form $[V_i, E_i, \gamma_i, P_i^{in}, P_i^{out}, Ep_i, sub_i]$ meets the following conditions

1.1 $[V_i, E_i]$ is a finite, directed, acyclic graph with the set of vertices V_i and the set of edges $E_i \subset V_i \times V_i$.

1.2 γ_i assigns to every edge its logical conditions of usage[3], an issue of particular relevance at branching points that may describe either alternatives or parallelism.

1.3 $P_i^{in} \cup P_i^{out}$ contains the pins, i.e., the input and output nodes $P_i^{in}, P_i^{out} \subseteq V_i$ as follows:
$P_i^{in} = \{v \mid v \in V_i \wedge \nexists u \in V_i((u,v) \in E_i)\}$
$P_i^{out} = \{v \mid v \in V_i \wedge \nexists u \in V_i((v,u) \in E_i)\}$

1.4 Vertices in $Ep_i \subseteq V_i$ are called episodes that are to be substituted by other graphs later on. $V_i \setminus Ep_i$ is called the set of scenes that have a semantics in the domain.

1.5 $sub_i : Ep_i \rightarrow 2^{\{1,\ldots k\}} \setminus \emptyset$ is a mapping that assings to every episode graphs for potential substitution.

and the mapping c defined on $\{1, \ldots, k\}$ assigns to every graph its logical conditions of usage[3].

The design of experiences of game play taking pedagogical concepts into account takes place as an interdisciplinary process of negotiating graphs, conditions, and their mutual interdependencies. Storyboarding works well for large-scale training applications [AFJ 2013a, AFJ$^+$ 2013b].

[3]The issue of logics is suppressed and the authors confine themselves to what is provided by Prolog.

3.2 Dynamic Storyboard Expansion

From a formal point of view, a storyboard as introduced in the preceding subsection 3.1 forms a certain type of graph grammar [AJ 1994, Kir 1995].

The crucial peculiarity is that the grammar generates particular graphs at execution time by dynamic graph expansion in dependence on context conditions including recent human user profiles. This goes beyond the expressive limits of all former graph grammar concepts [Kir 1995] and, in this way, provides a formal explanation of the current methodology's flexibility and reach. The present subsection is aimed at a sketch of the dynamic graph expansion concept.

Episodes are placeholders for varying anticipated experiences. Didactic design means the pedagogical preparation of potential experiences including the specification of preconditions.

Formally, if $\mathcal{G}_i$ is a graph of the storyboard and $e \in EP_i$ is one of its episodes, $sub_i(e)$ names all the graphs that may be used for substitution, provided that the respective substitution condition is valid. For $j \in sub_i(e)$ with a valid condition $c(j)$, $\mathcal{G}_i[e \hookleftarrow \mathcal{G}_j]$ denotes the result of substitution.

Using logic programming as a technology of automated reasoning, there is no need to treat graphs as complex data structures. Instead, it is sufficient to maintain predicates that describe nodes and edges, respectively. The authors will make use of this simplicity subsequently.

However, the usage of any particular programming paradigm assumes the understanding of an abstract machine that determines the meaning of syntactic constructs.

Formal methods based on mathematical notions and notations appear much more complex, at a first glance. But in the very end, they explain everything on a few lines.

To avoid ambiguity – in theory, not necessarily in programmimg practice – it helps to rename nodes that are substituted into another graph such that (i) it remains clear where the nodes are coming from and (ii) it is obvious where they are residing now. Formally, this looks as follows:

When a certain substitution $\mathcal{G}_i[e \hookleftarrow \mathcal{G}_j]$ takes place, every node $d \in V_j$ is renamed to $e.d$. Accordingly, $e.V_j$ is shorthand for $\{e.d \mid d \in V_j\}$.

Based on this terminology, the expansion of an episode e in some graph $\mathcal{G}_i$ by an admissible graph $\mathcal{G}_j$ results in a pin graph $\mathcal{G}_i[e \hookleftarrow \mathcal{G}_j]$ of the form $[V, E, \gamma, P^{in}, P^{out}, Ep, sub]$ such that the following conditions are satisfied (in 2.3, double parentheses dropped, *sub* written relationally).

2.1 $V = (V_i \setminus \{e\}) \;\cup\; e.V_j$

2.2 $E = (E' \setminus E'') \cup E''' \cup E''''$

(a) $E' = E_i \cup e.E_j$

(b) $E'' = V_i \times \{e\} \;\cup\; \{e\} \times V_i$

(c) $E''' = \{v \mid (v, e) \in E_i\} \times P_j^{in}$

(d) $E'''' = P_j^{out} \times \{v \mid (e, v) \in E_i\}$

2.3 $\gamma(u, v) = \gamma_i(u, v)$ for $(u, v) \in E \cap E_i$ and $\gamma(e.u, e.v) = \gamma_j(u, v)$ for $(u, v) \in E_j$

2.4 $P^{in} = \begin{cases} (P_i^{in} \setminus \{e\}) \cup e.P_i^{in} & \textit{if } e \in P_i^{in} \\ P_i^{in} & \textit{otherwise} \end{cases}$

2.5 $P^{out} = \begin{cases} (P_i^{out} \setminus \{e\}) \cup e.P_i^{out} & \textit{if } e \in P_i^{out} \\ P_i^{out} & \textit{otherwise} \end{cases}$

2.6 $EP = EP_i \setminus \{e\} \;\cup\; e.EP_j$

2.7 $sub = (\, sub_i \setminus \{(e, sub_i(e))\} \,) \;\cup\; \bigcup_{d \in C_j} \{(e.d, sub_j(d))\}$

Graph expansion determines a rewrite relation that terminates with graphs that do not contain any more episodes. Those are formal representations of all the possible human experiences.

3.3 Storyboard Interpretation Technology

The term *storyboard interpretation technology* denotes the authors' original approach to storyboard expansion at execution time; in the context of educational gamification that means when playing and learning. It works in large-scale application scenarios [AFJ 2013a, AFJ$^+$ 2013b].

The authors refrain from disclosing all the details and prefer to give an introduction into the essentials that are sufficient for the present report and, in particular, for the related Prolog implementation.

Without loss of generality, one may assume that the top-level graph of a storyboard has exactly one input node.

Storyboard interpretation basically means that there are pointers (initially one, later on perhaps more than one) indicating the actions to execute. (i) In case the action pointed at is a scene, its operational semantics is to be executed such as playing a cutscene or a video, offering buttons or click areas for making a learner's choice, providing documents for download, etc. (ii) Instead, if the pointer indicates an episode, graph expansion is executed. For this purpose, the candidate graphs are found by the function *sub* and the substitution conditions provided by the function c are checked. The first admissible substitution is executed[4]. (iii) Finally, the pointer is moved forward along admissible edges (see function γ) and, in case there is more than one admissible, split accordingly. In case (ii) of substitution, pointers are placed on all substituted input nodes. (iv) At junctions where edges meet, pointers coalescence.

Notice that only case (i) is perceived by the human learner/player, whereas (ii), (iii) and (iv) take place in the background.

Adaptivity and personalization is implemented by means of checking the validity of the formulas provided by c, *sub*, and γ.

Instead of a shallow discussion of many further issues, the authors prefer to complete the present subsection by an in-depth discussion of just a single issue.

The point (iv) above seems to cause difficulties, if different pointers arrive at a junction at different points in time. At least, this is an interesting observation worth to be discussed.

According to the original storyboarding approach developed in [JK 2005], edges represent a partial order of events. This coincides with the slightly later introduction of story spaces as partially ordered spaces of events [Jan 2009], where the evolution of stories over time motivates dramaturgical aspects of storytelling such as non-linearity and non-monotonicity.

For all these considerations, it is sufficient to interpret an edge from some node 1 to some other (graphs are acyclic ...!) node 2 as just the statement that the event at node 1 precedes the event at node 2.

Consequently, there exists a standard algorithmic interpretation which, interestingly, is the computationally most simple one. At any junction, whenever the first pointer reaches this point, the corresponding node is ready for execution. All pointers that arrive later have no impact and resolve at the junction.

Apparently, there do exist different, more complex interpretations. In application cases, domain experts, educators, and others may have their good reasons for varying interpretations. The hierarchical structure, the modularity, and the lucidity of logical conditions of storyboards provide an ideal basis for negotiations and decisions. The authors briefly mention an alternative.

By way of illustration, one may see all paths that lead to a junction as necessary preconditions of execution. In this case, the system simply waites for the arrival of all expected pointers.

[4]By the way, this perfectly fits the logic programming mindset.

4 Plug & Play – the Touch and Feel – Revisited

After introducing the essentials a bit more formally, it becomes clear that the two graphs of the introductory example do not meet all the requirements of section 3. In particular, these graphs are not acyclic. The present section is aimed at fixing the problem.

The authors present a small collection of graphs seen as part of an overall storyboard under consideration. In this way, authors and readers exercise the usage of the formal concepts and exemplify didactic design within the authors' framework to approximate the gist of the introductory example. Graphs are numbered, as usual. There may be other graphs in the storyboard not of interest here, say as many as exactly 100. Thus, our additional graphs begin with index 101. For readability, nodes are denoted by strings instead of numbers. Input and output nodes are indicated by a lighter background color. Scenes are represented by cycles.

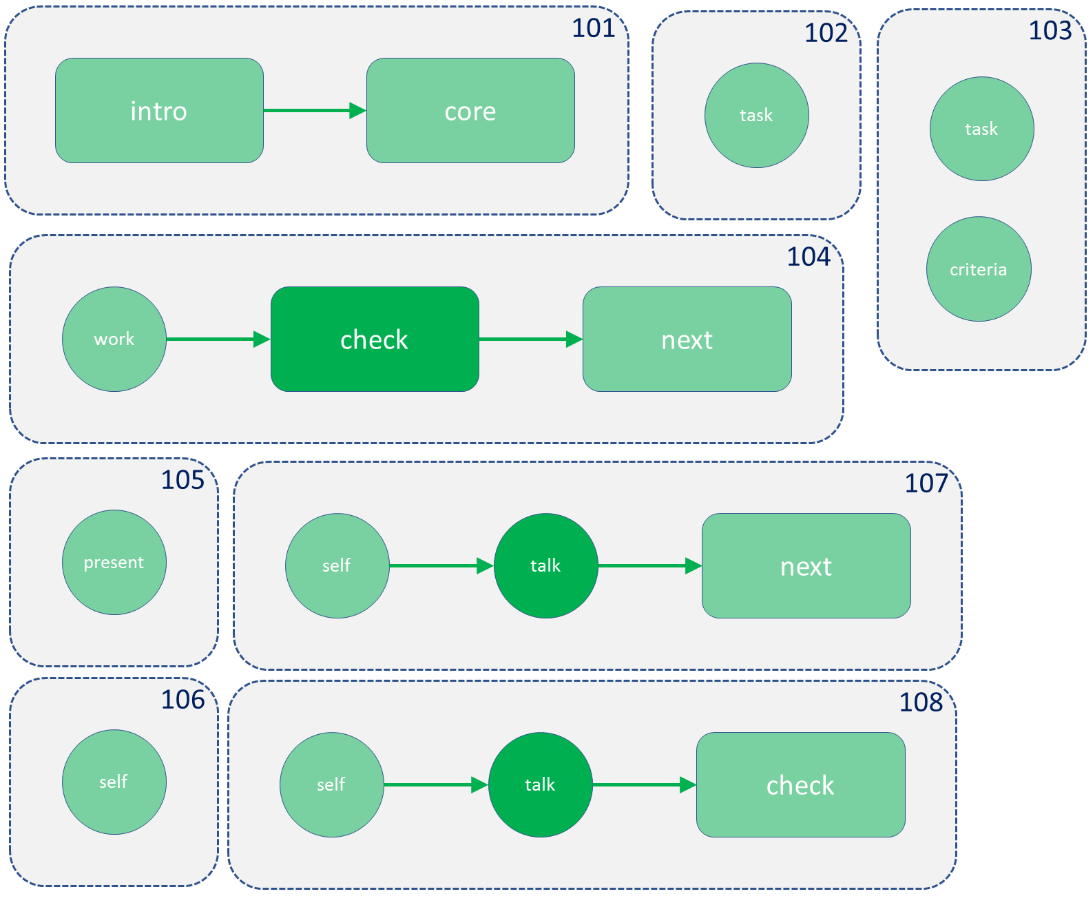


Figure 3: Informal Visualization of Graphs for the Pedagogical Pattern "Built-In Failure"

Subsequently, dynamic storyboard expansion is briefly illustrated beginning with graph 101. Formalities are mentioned, but used loosely.

Graph $\mathcal{G}_{101}$ consists of two episodes named *intro* and *core*, respectively. For expansion of the episode *intro*, two graphs are available. In formal terms, $sub_{101}(intro) = \{102, 103\}$. This allows for the two different beginnings as visualized in figure 1. The occurring scenes have a certain semantics in the domain such as, e.g., explanatory videos or texts. As indicated in [JK 2005], the meaning is not necessarily digital. Such a scene may also mean a teacher's introductory talk.

The episode core is to be substituted by $\mathcal{G}_{104}$, i.e., $sub_{101}(core) = \{104\}$. The substitution condition $c(104)$ must be always true. Otherwise, graph expansion gets stuck.

The episode *next* allows for recursive substitutions reflecting the idea of cycles as visualized in figure 1. $sub_{104}(next) = \{104, 105\}$.

The expansion of the node *check* yields either the activity of the learner to check by her(him)self her(his) intermediate results in privacy or to consult a peer. The communication with a peer may loop through several cycles as already indicated in figure 1. $sub_{104}(check) = \{106, 107, 108\}$. Similarly, $sub_{107}(next) = \{104, 105\}$ and $sub_{108}(check) = \{106, 107, 108\}$.

Reasonably, the substitution conditions $c(107)$ and $c(108)$ should require the availability of a peer for communication.

The following figure 4 exemplifies the result of several subsequent steps of graph expansion.

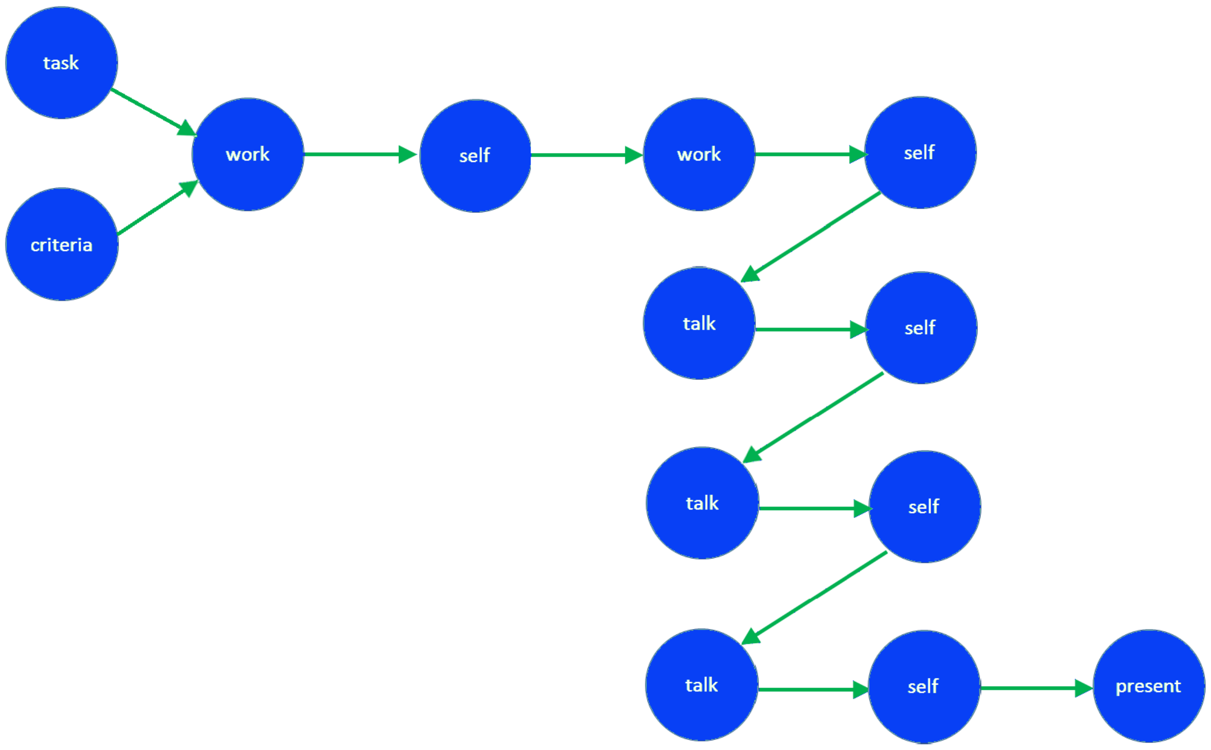


Figure 4: Interaction Process Based on the Pedagogical Pattern “Built-In Failure”

The scene *work* describes activities of the student intended to work on the initial *task* and *self* means the student’s thinking about the intermediate solution in the light of the initial *criteria* of success. When peers are available, the scene *talk* means to discuss the current state of the problem solving process. After discussion and repeatedly pondering the own results, a student may decide to finish the work and to *present* the results.

This particular interaction process reflects features of the two graphs on display in figure 1. It begins with the two scenes of the first graph and unfolds one of the recursions of the second.

The graph informally represented by figure 4 is one element – in fact, only a cutout of one element – of the formal graph language generated by the underlying storyboard. Usually, storyboards as introduced in section 3 define infinite formal graph languages. Members of such a formal language are not yet fully defined at planing time (!) when the storyboard is designed. They emerge at execution time in dependence on unforeseeable context conditions and behavior.

5 Plug and Play in Prolog

In technology enhanced learning and training in general and in game-based learning and training in particular domain experts, educators, designers, and educational system developers investigate and negotiate didactics and ludology. Within the authors' framework of digital storyboarding, the outcome is digital and, thus, may be subject to direct execution [Jan 2013a].

Many of the issues under discussion have a logical component – validity of context conditions, achievements of learners reflected in user profiles, preconditions of the usage of didactic concepts, and the like. Logic programming provides an appropriate language for such an interdisciplinary discussion. Whatever the partners agree about, it becomes immediately executable.

Notice that in the present technical report, the authors do not intend to demonstrate anything such as an implementation of the storyboard interpretation technology by means of logic programming. Such a software-technological endeavor would be missing the ISIP contribution's focus on adaptivity and personalization.

In the present section, the authors confine themselves to an explanation that, why, and how *logic prgramming is appropriate* to represent features of the *plug and play technology* which is the core of *dynamic formative evaluation.*

5.1 Static Plug and Play

The present subsection is aimed at the discussion of the most simple and, therefore, extremely lucid variant of plug and play with didactic concepts. The intention is to bridge the gap from the intuitive understanding to precise formalisms that allow for a straightforward implementation.

Recall the introductory example of section 1. Apparently, there are didactic alternatives. Static plug and play means to carry out most simple modifications – plug – and try them – play. After employing the lower left graph, one may modify the process specification such that the upper right graph is invoked. When the system is running, one may record whether or not and, if so, how frequently learners make use of the opportunity to consult their peers. This supports formative evaluation (see figure 2).

Assume any storyboard $\mathcal{F}$ with a top-level graph $\mathcal{G}_1$ and with the particular strong restriction that within every graph $\mathcal{G}_i$ in $\mathcal{F}$ and for every of its episodes $e \in EP_i \subseteq V_i$ it holds $|\, sub_i(e)\,| = 1$. In other words, there is never any choice and always only one graph for substitution available. As a side-effect, the mapping c must necessarily always return the constant $\top$, the value "*true*".

$\mathcal{F}$ unfolds during interaction dynamically, but the result is always the same terminal graph.

Educators may monitor the process of game-based learning based on $\mathcal{F}$ and ponder the one or the other opportunity of change.

Assume that educators focus on a particular graph $\mathcal{G}_i$ in $\mathcal{F}$ having some doubts about its effectiveness. Then, they are challenged to think about an alternative $\mathcal{G}'_i$, very much like the two graphs in figure 1 are alternatives to each other. Plug and play means to replace $\mathcal{G}_i$ by $\mathcal{G}'_i$ in $\mathcal{F}$. Needless to say that this requires some care to guarantee termination of graph rewriting.

Pedagogy is difficult, because humans are complex and learning is something manifold and not sufficiently well-understood. Human media reception is highly individual and so is game play. Concurrent pedagogical concepts are really concurring, i.e., one cannot assume that one is always superior to the other. Educational gamification makes it even more involved.

The best is to try it out; *plug and play* is the appropriate technology that allows for comparison of clearly specified and locally encapsulated didactic and/or game play concepts.

5.2 Dynamic Plug and Play

One of the key original features of the authors' approach is dynamics inherited from [Arn 1996] through [JK 2005]. Consequently, the plug and play concept has to be seen dynamically as well.

In Arnold's approach to Artificial Intelligence control of disturbed complex systems [Arn 1996], dynamics means to respond to information that comes in during the process of system therapy. You do not know in advance all details of a seriously disturbed technological process and, thus, you do not know completely what to do. Over time, you learn about the problem and, in this way, you learn how to take over control. In dynamic environments, *planning is learning* [AJ 1996].

The process of educational gamification by transformation may be also seen as a dynamic planning process. Storyboards are plans that specify a space of potential future interaction processes of playing and learning. The particular plan to be realized is not known in advance. It unfolds during interaction dynamically in dependence on conditions that may change over time such as the context, states of the environment possibly including process simulations, and the emerging learner/player profile.

The crux is that in such a dynamic world it cannot be clear from the early beginning and in every detail which didactic concepts are appropriate in a developing context, facing running process simulations, and taking individually evolving human learner/player profiles into account. The problem is complicated by the open question whether or not humans can identify with settings of game play and feel encouraged to engage in some previously unknown virtual world. The intriguing relationship between the real and the virtual [JL 2012] comes into play as well.

When a digital storyboard is set up in practice (as in [AFJ 2013a] and [AFJ$^+$ 2013b], e.g.), it remains open whether the one or the other didactic concept fits better both the environmental conditions and the needs and desires of the current users.

Consequently, evaluation must by dynamic as well. It has to take place at execution time, i.e., at the time of playing and training. A gamified educational system is never completed and does never remain unchanged. It evolves when being used due to emerging didactic insights.

So, assume gamified learning or training in acton. The storyboard unfolds (sections 3.2, 3.3). Educators monitoring the running game-based learning and training process may undertake exploratory control measures.

The simplest form of intervention is to modify the storyboard's control function c which determines the conditions of graph substitution. $c(i)$ defines when $\mathcal{G}_i$ is allowed for substitution. Changes to $c(i)$ may have immediately recognizable effects. The extreme setting $c(i) = \perp$, so to speak, unplugs graph $\mathcal{G}_i$. In contrast, $c(i) = \top$ may increase the likelyhood of using graph $\mathcal{G}_i$. More refined changes to c may result in better adjusted adaptation and personalization.

However, all these potential changes remain within the a priori anticipated assignment of potential expansions to episodes. Changes to the sub_i function in graphs go beyond this limit. The simplest change is to remove some $j \in sub_i(e)$ what, a bit more subtly, means to unplug graph $\mathcal{G}_j$, not in general, but from the candidates to expand the episode e in $\mathcal{G}_i$.

More creative, more far reaching, and more risky is to extend in some Graph $\mathcal{G}_i$ for some episode e the expansion repository $sub_i(e)$. This may change the educational process substantially and requires didactic care.

As a result, knowledge about the suitability of didactic concepts, about learner and trainee acceptance of playful human-system and peer to peer interaction, and about the effectiveness of learning and game play amalgamation emerges throughout educational gamification application and formative evaluation. *Plug and play* is the key technology of *exploratory didactics*.

5.3 Logical Reasoning and Programming

Logical reasoning does not need a particular programming paradigm as the pioneers of Artificial Intelligence, Allen Newell and Herbert Simon, have been demonstrating when developing – more than 60 years ago – the first automated theorem prover in IPL, their own assembly language that provided important features generalized, streamlined, and then incorporated into Lisp[5].

However, the transdisciplinary educational gamification process (reconsult the transformation perspective in section 2) involves specialists from largely verying disciplines such as domain experts, educators, AR and VR developers, game designers, and the developers carrying out the transformation. They investigate and negotiate manifold logical issues such as (i) the conditions controlling the invocation of didactic concepts, i.e., the substitution of graphs at time of play, (ii) the interpretation of context data and learner/player profiles, (iii) the reasons of branching, and (iv) the control of parallelism. Conceptually, these are all issues of logics (see section 3).

It is advantageous, if all the crucial decisions have a readable declarative representation in the system that can be made visible and that is, if necessary, easy to modify. Logic programming in whatsoever form [CM 1981, SS 1986, Kna 1993] satisfies the needs of this transformation process.

5.3.1 Plug

Essentially, a graph is just a set of nodes and edges. This is easily represented using two predicates

```
node(Graph,Node)
edge(Graph,StartNode,EndNode}
```

Furthermore, the process of graph expansion requires knowledge about input nodes and output nodes easily provided by two related predicates.

```
inputnode(Graph,Node)
outputnode(Graph,Node)
```

Who wants to expand one graph by substituting another graph into one of its episodes just has to name the first graph, the episode, and the second graph. Simply add the following fact

```
subst(Graph1,Episode,Graph2).
```

to the Prolog base and the substitution is done. This is comprehensible to specialists from any of the involved disciplines.

In the background, the system has knowledge about graph substituion formalized as follows.

```
substedge(Graph1,Episode,Graph2,StartNode,EndNode) :-
   subst(Graph1,Episode,Graph2),
   edge(Graph1,StartNode,Episode),
   inputnode(Graph2,EndNode).

substedge(Graph1,Episode,Graph2,StartNode,EndNode) :-
   subst(Graph1,Episode,Graph2),
   edge(Graph1,Episode,EndNode),
   outputnode(Graph2,StartNode).
```

[5]http://www-formal.stanford.edu/jmc/history/lisp/node2.html

The system knows furthermore that edges that result from substitutions may be treated as usual edges, because it has the following background knowledge.

```
edge(Graph1,StartNode,EndNode) :-
   substedge(Graph1,Episode,Graph2,StartNode,EndNode).
```

Because plug and play is a principle that allows for playing with varying alternatives, the ability to unplug one graph substituted before into another for a particular episode is a necessary feature. Tell the system to retract all edges that are inserted by substitution of a second graph into a first one for a particular episode.

```
?- retractalledges(Graph1,Episode,Graph2).
```

The system understands “retractalledges” as follows.

```
retractalledges(Graph1,Episode,Graph2) :-
   retract(substedge(Graph1,Episode,Graph2,StartNode,EndNode)),!,
   retractalledges(Graph1,Episode,Graph2).
retractalledges(Graph1,Episode,Graph2).
```

These are the essentials of Prolog based graph expansion including revision by unplugging.

5.3.2 Play

The second author undertook several projects of storyboard verification [KD 2008] the concepts of which have been implemented by several students resulting in comprehensive sources such as, pars pro toto, [Düs 2007].

What may happen during game play and learning after some exploratory plugin was executed? Due to dynamics, answers to questions like this change over time. For the time being, complex logical simulation approaches as the authors' one in [JA 1996], esp. figure 2, p. 185, are skipped.

Simpler, but less reliable is to ask which nodes[6] may be reached by pointers in the course of storyboard interpretation. There are some initial facts and rules known to the system such as the following, where facts as shown here are given for every input node of the top-level graph.

```
pointeron(Graph1InputNode).
pointeron(Node) :-
   edge(_,NodeBefore,Node),
   pointeron(NodeBefore),
   gamma(_,NodeBefore,Node).
```

The present section is closed by directing the audience's attention to enormous potentials that are – to the authors very best knowledge – not yet considered in any storyboarding project. The underlying central concept is meta data.

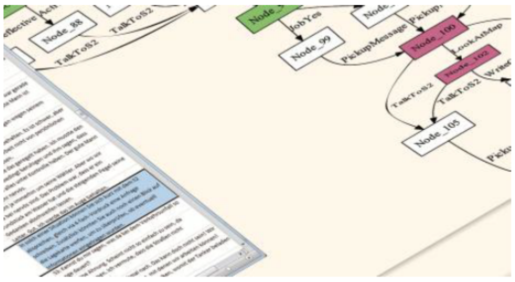
One may easily equip whole graphs and single nodes with meta data such as didactic concepts. On this basis, interesting features of storyboards may be checked automatically such as the occurrence, the frequency of the occurrences, or the alternating occurrence of didactic concepts in potential unfoldings, i.e., normal forms with respect to graph rewriting, of a given storyboard. From a higher perspective, one may ask for the occurrence of pattern instances in unfolded graphs.

[6]Nodes relate to didactic concepts, to human activities, to exercises, to videos, documents of any type, etc.

6 Scenarios of Exploratory Didactics

Scenarios of exploratory didactics describe courses of action to implement the ideas developed in section 5.2. A comprehensive treatment, on the one hand, would be worth a separate publication, maybe even a manual, guidelines, a handbook, or anything like this, and on the other hand, it definitely requires more experimentation in practice than the authors have undertaken so far. By way of illustration, sketches of two scenarios shall do here within the limited space available.

In the storyboard of the authors' staff training system "TraSt" developed for the German Federal Office for Civil Protection and Disaster Management (consult [AFJ 2013a, AFJ$^+$ 2013b]),



from [AFJ+ 2013b]

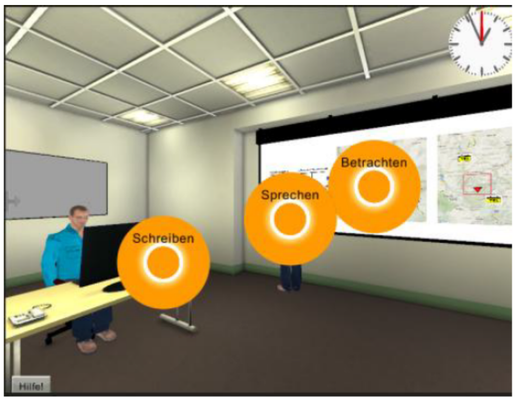


Figure 5: Practical Example from the Staff Training System "TraSt"with Focus on Node 100

there is the particular `Node_100` on display in the left picture of figure 5. The implementation is shown on the right. In this scene, the human user is offered a choice between three alternatives: filling a form (in German: Schreiben), talking to another staff member (Sprechen), and looking at the situation map (Betrachten).

There are some staff trainees who usually avoid filling forms whereas others avoid talking. It is worth to try substitutions (i) in which trainees are forced to do what they tend to avoid, (ii) in which they have other alternatives, or (iii) in which the usually avoided alternative is easier to select (more central, larger) and, in this sense, syntactically suggested.

Another interesting scenario of analysis arises from recursion as shown figures 1, 3, and 4, where in the latter one it is unfolded. By counting the number of unfolded loops that occur throughout amalgamated game play and learning, one may find out whether or not trainees draw advantage from the opportunity of repetition.

Substitution conditions $c(i)$, usage conditions $\gamma(u, v)$, and repositories for substitution $sub_i(e)$ may be modified, e.g., to force trainees into one or a few initial loops.

Acknowledgements

Part of the present work has been substantially supported within the R & D project MeTa DG, contract 2016 FE 0153, by the Thüringer Aufbaubank (TAB). In particular, some ideas for and concepts of adaptivity and personalization at user interfaces are carried over from MeTa DG to educational gamification.

Some colleagues and friends who contributed to theory development, to applications, and to formative evaluation are, among others, Sebastian Arnold, Jun Fujima, and Jacqueline Schuldt née Krebs.

References

[AFJ 2013a] Sebastian Arnold, Jun Fujima, and Klaus P. Jantke. Storyboarding serious games for large-scale training applications. In Owen Foley, Maria Teresa Restivo, James Uhomoibhi, and Markus Helfert, editors, *Proceedings of the 5th International Conference on Computer Supported Education, CSEDU 2013, Aachen, Germany, May 6-8, 2013*, pages 651–655, 2013.

[AFJ$^+$ 2013b] Sebastian Arnold, Jun Fujima, Klaus P. Jantke, Andreas Karsten, and Harald Simeit. Game-based training for executive staff of professional disaster management: Storyboarding adaptivity of game play. In Deyao Tan, editor, *Proceedings of the International Conference on Advanced Information and Communication Technology for Education (ICAICTE 2013), Sept. 20-22, 2013, Hainan, China*, pages 68–73. Atlantis Press, 2013.

[AJ 1994] Oksana Arnold and Klaus P. Jantke. Therapy plans as hierarchically structured graphs. In *Fifth International Workshop on Graph Grammars and their Application to Computer Science, Williamsburg, Virginia, USA*, November 1994.

[AJ 1996] Oksana Arnold and Klaus P. Jantke. Planning is learning. In Werner Dilger, Michael Schlosser, Jens Zeidler, and Andreas Ittner, editors, *Machine Learning, 1996 Annual Meeting of the Special Interest Group of Machine Learning of the German Computer Science Society (GI), Chemnitzer Informatik-Berichte CSR–96–06*, pages 12–17. TU Chemnitz, 1996.

[AJ 2018] Oksana Arnold and Klaus P. Jantke. Educational Gamification & Artificial Intelligence. ADICOM TECH REPORT 03-2018, ADICOM Software, December 2018.

[AJV$^+$ 2010] Oksana Arnold, Klaus P. Jantke, Christoph Vogler, Hans-Rainer Beick, and Jan Opfermann. The reach of dynamic plan generation for enterprise planning Web services. In *The IET International Conference on Frontier Computing, Proceedings (DVD), Taichung, Taiwan, August 4-6, 2010*, pages 359–365, 2010.

[Arn 1996] Oksana Arnold. *Die Therapiesteuerungskomponente einer wissensbasierten Systemarchitektur für Aufgaben der Prozeßführung*, volume 130 of *DISKI*. St. Augustin: infix, 1996.

[Bog 2011] Ian Bogost. Gamification is bullshit. *The Atlantic*, August 2011. [https://www.theatlantic.com/technology/archive/2011/08/gamification-is-bullshit/243338/ last access on April 30, 2019].

[CM 1981] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog*. Berlin, Heidelberg, New York, London, Paris, Tokyo: Springer, 1981.

[DDKN 2011] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart E. Nacke. From game design elements to gamefulness: Defining "gamification". In *Proc. MindTreck'11, Sept. 28-30, 2011, Tampere, Finland*, pages 9–15, 2011.

[DOS$^+$ 2011] Sebastian Deterding, Kenton O'Hara, Miguel Sicart, Dan Dixon, and Lennart E. Nacke. Gamification: Using game design elements in non-gaming contexts. In *Proc. CHI'11, May 7-12, 2011, Vancouver, BC, Canada*, pages 1–4, 2011.

[Düs 2007] Horst Düsel. Konzeption und Realisierung von Methoden der formalen Verifikation von Storyboards. Diploma thesis, Ilmenau University of Technology, Faculty of Economic Sciences, 2007.

[FBJ 2015] Susanne Friedemann, Lisa Baumbach, and Klaus P. Jantke. Textbook gamification. Transforming exercises into playful quests by using webble technology. In Markus Helfert, Maria Teresa Restivo, James Uhomoibhi, and Susan Zvacek, editors, *Proceedings of the 7th International Conference on Computer Supported Education, CSEDU 2015, Lisbon, Portugal, May 23-25, 2015*, pages 116–126. SCITEPRESS, 2015.

[Fle 1996] Karl-Heinz Flechsig. *Kleines Handbuch didaktischer Modelle.* Eichenzell, Germany: Neuland, 1996.

[HLW 1992] Franz Höfting, Thomas Lengauer, and Egon Wanke. Processing of hierarchically defined graphs and graph families. In Burkhard Monien and Thomas Ottmann, editors, *Data Structures and Efficient Algorithms*, volume 594 of *LNCS*, pages 44–69, 1992.

[JA 1996] Klaus P. Jantke and Oksana Arnold. A modal temporal logic and its models underlying variants of planning algorithms. In Luca Chittaro, Scott Goodwin, Howard Hamilton, and Angelo Montanari, editors, *TIME–96, Intern. Workshop, Key West, FL, USA, May 19–20, 1996*, pages 182–187. IEEE Computer Society Press, 1996.

[Jan 2006] Klaus P. Jantke. Layered Languages of Ludology: The Core Approach. Diskussionsbeiträge 25, TUI IfMK, November 2006.

[Jan 2009] Klaus P. Jantke. The evolution of story spaces of digital games beyond the limits of linearity and monotonicity. In I.A. Iurgel, N. Zagalo, and P. Petta, editors, *Proceedings of the 2nd International Conference on Digital Storytelling, Dec. 9-11, 2009, Erfurt, Germany*, number 5915 in LNCS, pages 308–311. Springer-Verlag Berlin Heidelberg 2009, 2009.

[Jan 2013a] Klaus P. Jantke. Direct execution learning technology (Invited Keynote). In Oksana Arnold, Wolfgang Spickermann, Nicolas Spyratos, and Yuzuru Tanaka, editors, *Webble Technology, First Webble World Summit, WWS 2013, Erfurt, Germany, June 2013*, volume 372 of *Communications in Computer and Information Science*, pages 90–109. Springer, 2013.

[Jan 2013b] Klaus P. Jantke. Pedagogical patterns and didactic memes for memetic design by educational storyboarding. In Oksana Arnold, Wolfgang Spickermann, Nicolas Spyratos, and Yuzuru Tanaka, editors, *Webble Technology, First Webble World Summit, WWS 2013, Erfurt, Germany, June 2013*, volume 372 of *Communications in Computer and Information Science*, pages 143–154. Springer, 2013.

[Jan 2018a] Klaus P. Jantke. Konzepte und Technologien der Gamification durch Transformation. ADICOM TECH REPORT 04-2018, ADICOM Software, December 2018.

[Jan 2018b] Klaus P. Jantke. No Thrill – No Skill. Ein systematischer Zugang zum Konzept Gamification. ADICOM TECH REPORT 02-2018, ADICOM Software, November 2018.

[JK 2005] Klaus P. Jantke and Rainer Knauf. Didactic design through storyboarding: Standard concepts for standard tools. In Beate R. Baltes, Lilian Edwards, Fernando Galindo, Jozef Hvorecky, Klaus P. Jantke, Leon Jololian, Philip Leith, Alta van der Merwe, John Morison, Wolfgang Nejdl, C. V. Ramamoorthy, Ramzi Seker, Burkhard Shaffer, Iouliia Skliarova, Valery Sklyarov, and John Waldron, editors, *First International Workshop on Dissemination of E-Learning Technologies and Applications, DELTA 2005, in: Proceedings of the 4th International Symposium on Information and Communication Technologies, Cape Town, South Africa, January 3–6, 2005*, pages 20–25. Computer Science Press, Trinity College Dublin, Ireland, 2005.

[JL 2012] Klaus P. Jantke and Denise Lengyel. Die Realität in virtuellen Welten. *Zeitschrift für e-Learning*, (1):7–22, 2012.

[KD 2008] Rainer Knauf and Horst Duesel. Towards verification of storyboards. In *21st International Florida Artificial Intelligence Research Society Conference 2008, Coconut Grove, FL, USA*, pages 371–372. AAAI Press, 2008.

[Kir 1995] Daniel Kirsten. Properties of formal languages of therapy plans created by graph grammars. In Martin Kutrib and Thomas Worsch, editors, *5. Theorietag Automaten und Formale Sprachen, Schloß Rauischholzhausen, 28./29. September 1995*, pages 42–47, 1995.

[KJ 2014] Jacqueline Krebs and Klaus P. Jantke. Methods and technologies for wrapping educational theory into serious games. In Susan Zvacek, Maria Teresa Restivo, James Uhomoibhi, and Markus Helfert, editors, *Proceedings of the 6th International Conference on Computer Supported Education, CSEDU 2014, Barcelona, Spain, May 1-3, 2014*, pages 497–502. SCITEPRESS, 2014.

[Kna 1993] Rainer Knauf. *Logische Programmierung und wsissensbasierte Systeme – Eine Einführung.* Düren, Germany: Shaker Verlag, 1993.

[Len 2009] Carsten Lenerz. Layered Languages of Ludology – Eine Fallstudie. In Anja Beyer and Gunther Kreuzberger, editors, *Digitale Spiele – Herausforderung und Chance*, pages 39–52. Boizenburg, Germany: vwh, 2009.

[Ped 2012] Pedagogical Patterns Editorial Board, editor. *Pedagogical Patterns: Advice for Educators.* Joseph Bergin Software Tools, 2012.

[SS 1986] Leon Sterling and Ehud Shapiro. *The Art of Prolog.* Cambridge, ME, USA: MIT, 1986.

List of the ADICOM TECH REPORTS

01-2018	Klaus P. Jantke, Sebastian Drefahl & Oksana Arnold The Power and the Limitations of Concepts for Adaptivity and Personalization Characterized by Benchmarks of Inductive Inference Version 1.00, 31.10.2018
02-2018	Klaus P. Jantke No Thrill – No Skill. Ein systematischer Zugang zum Konzept Gamification Version 1.00, 26.11.2018
03-2018	Oksana Arnold & Klaus P. Jantke Educational Gamification & Artificial Intelligence Version 1.00, 17.12.2018
04-2018	Klaus P. Jantke Konzepte und Technologien der Gamification durch Transformation Version 1.00, 28.12.2018
01-2019	Klaus P. Jantke, Rainer Knauf, Oksana Arnold & Jürgen Cleve Exploratory Didactics by Plug and Play in Prolog Version 1.00, 30.04.2019