Dynamic Cognitive Modeling for Adaptive Serious Games

Alexander Streicher[1], Julius Busch[2], and Wolfgang Roller[1]

[1] Fraunhofer IOSB, Karlsruhe, Germany
firstname.lastname@iosb.fraunhofer.de
[2] KIT, Karlsruhe, Germany
julius.busch@accenture.com

Abstract. Cognitive modeling can be a viable tool to assess the cognitive state of the users and to determine their current learning needs. For instance, adaptive educational systems must match the learning needs by estimating the level of memorization or forgetting. The research question is, how to model latent cognitive variables such as memory degradation and how to make use of it for adaptivity scenarios in the e-learning context. Tools like cognitive architectures with established psychological underpinnings can help here. However, development of cognitive architecture models is often complex, domain- and application-specific and its transfer or general applicability is not evident. We present an innovative dynamic modeling approach which automatically creates declarative rules from interoperable activity stream observations to form models for the cognitive architecture ACT-R. The developed framework uses those models to analyze user actions according to their frequency, temporal occurrence and memory activation levels. An adaptive e-learning system can use the chunks' activation levels to assess which concepts need repeated user attention. A prototype implementation for a serious game for process training demonstrates the feasibility of the approach.

Keywords: user modeling; learner state; cognitive modeling; adaptivity

1 Introduction

User modeling is key for adaptive e-learning or assistance systems which aim to react to the users' (learning) needs. Adaptivity here means to personalize the usage experience to the individual needs of the users and their current working context. The quality of adaptive systems directly depends on their understanding of the users, i.e., on their user models. The more realistic these models represent the mental states and cognitive processes, the more precise the approximation of needs can be, and the better is the automatic parametrization of assistance functions. That is of particular interest for the e-learning domain, where Intelligent Tutoring Systems (ITS) target an optimal support for the learners [29]. A central active research question for adaptive e-learning systems addresses the correct timing of adaptive educational systems, i.e., the issue of when to actually adapt. The question when to react or adapt is especially important for

digital game-based learning systems, e.g., for educational serious games, because we want to keep the users immersed in their gaming experience, not disturbing their game flow. Serious games are games which should entertain but have at least one additional characterizing goal [10]. In the case of educational serious games this goal is learning whereas the learning objectives should be aligned at the gaming experience and its objectives [18]. To achieve high learning outcomes, the game design should foster immersion and flow [13]. High immersion comes along with high intrinsic motivation to (continue) playing – and, hence, to continue to learn [13]. The very essence of digital game-based learning is to allow the users to play, to make mistakes, and to immerse themselves in a virtual environment [10]. Misdirected automatic reactions, at the wrong time, can have an extremely negative effect on immersion. For optimal efficacy and high user acceptance, the correct point in time is of fundamental importance. For example, an adaptive educational system could react when attention decreases, cognitive load increases, or when the user seems to be in a repetitive cycle with no real observable progress, or when there are signs of forgetting. Cognitive modelling addresses such kind of issues.

The research question is how cognitive modeling can contribute to "intelligent user models" and how it can be used for adaptivity in digital learning games. More specifically, how to use a data-driven approach to dynamically generate cognitive models which include latent cognitive variables such as cognitive load or memory decay.

The contribution of this work is a concept for cognitive user models for digital learning games by means of cognitive modelling and its application to adaptivity.

In comparison to cognitive models for general assistance systems, educational serious games come with additional training and learning aspects such as learning styles, achievement of learning goals, gameplay instruments such as trial and error, gamification, etc. This has to be considered when selecting a cognitive modeling tool. Our concept shows how to apply the cognitive architecture ACT-R to educational serious games, and how to control adaptivity by using memory activation levels from learned activity concepts. We have chosen ACT-R based on a subjective value-benefit study. The main criteria were its established psychological underpinnings and maturity, availability of implementation frameworks and community support, and its suitability to dynamically modify the declarative and procedural models. Key aspects of our approach are the dynamic generation of the ACT-R model based on observations, as well as its use of standardized activity stream data as a general scheme for the observed input data. This allows to apply the approach to other application domains. In our e-learning domain we make use of the tracking standard *Experience API* (xAPI) which is an established method to track learning, also in serious games [21]. Our work is embedded in research on frameworks for adaptive assistance systems [24].

A major challenge in the use of cognitive architectures is the required extensive modeling effort. Expert knowledge is needed for both the cognitive architectures and the application domain. Taatgen and Anderson (2010) discuss

Fig. 1. Example for adaptive hints (bottom, dashed rectangle) in the serious game Exercise Trainer (EXTRA) [25]. The virtual agent recommends considering an activity that was estimated to receive too little attention.

that it takes "substantial intellectual commitment to learn to understand models of a particular architecture and to learn to construct models" [26]. We address this issue by following a data-driven modeling approach. Our ACT-R model is designed as a combined user and domain model. User interactions are captured via xAPI and stored in a *Learning Record Store* (LRS) [1]. This interaction data together with the data in the declarative and procedural module of ACT-R forms the user models. A special feature of our approach is the dynamic modeling of the production rules. At runtime, interaction data is translated into ACT-R production rules for the virtual procedural ACT-R memory. Based on this model, cognitive processes can be simulated and analyzed. We make use of this to query the cognitive architecture which concept the user probably has "forgotten" – ACT-R allows to query the activation energy of the model's concepts. This activation energy – based on the Memory Decay Theory and Spreading Activation – is mapped to a continuous, real number and it is determined by the cognitive architecture at each simulation time step. Important game concepts, which are neglected by the user, can be identified by looking at their decreasing activation levels. A threshold function on the memory decay curve and ACT-R activation levels indicates "forgetting", and an adaptive system can react to this situation and offer appropriate assistance [24]. In our implementation the adaptive system offers hints on the next recommended or expected activities. This

recommendation is aligned at the simulated cognitive user model. Fig. 1 depicts an example for the serious game *Exercise Trainer* (EXTRA) [25].

2 Related Work

Adaptive serious games are related to *Intelligent Tutoring Systems* (ITS) [29,28]. Central to many ITS implementations is the monitoring of student interaction with the systems and to maintain a student or user model of knowledge and activity [9,29]. The literature shows multiple approaches of using cognitive architectures for Intelligent Tutoring Systems (ITS), such as the ACT-R based *Pump Algebra Tutor* by Anderson et al. (2001) [5] or the *Cognitive Tutoring Authoring Tools* (CTAT) by Aleven et al. (2006) [2]. Many ITS approaches utilize cognitive architectures for modeling [16,5]. At the core of this modeling is the division of knowledge into declarative and procedural units. Declarative knowledge represents atomic facts, while productions represent abstract individual goal-oriented problem-solving steps of a larger task. The modeling goal is to realistically represent human problem-solving thinking [16]. In this context, production rules comprise all possible solution paths that a student can undergo. A basic technique of cognitive tutors is called *model tracing* [5], which we follow for our solution approach. In model tracing, each action of the student is simulated simultaneously in the cognitive architecture. Subsequently the student's action can be classified as correct or incorrect by comparing it with this simulation. This allows the cognitive tutor to interact with the student in real time. Developers of adaptive learning systems based on cognitive architectures are confronted with complex modeling tasks, e.g., for the domain model. It requires expert knowledge and experience [26,20]. To deal with this problem, systems are developed that start from a higher level of abstraction and automatically design the more complex model from it. Examples of this are ACT-Simple and G2A, which are based on the Framework GOMS (Goals, Operators, Methods and Selection) [20,3]. The framework was conceptualized to make predictions about which methods and operations users apply in digital systems to complete known tasks. ACT-Simple uses a simple scripting language that compiles to ACT-R models [20]. Compared to our approach, here, the modeling effort should also be reduced and simplified, but our concept tries to simplify it by moving the modeling to the data observation level and to the activities to be observed. Several derivatives have evolved from GOMS, such as KLM-GOMS, CMN-GOMS, NGOMSL or CPM-GOMS. KLM-GOMS is a simple framework for the sequential description of expert behavior. ACT-Simple builds on this and combines it with the power of ACT-R. A similar work is the G2A system, which automatically GOMSL-Models transformed to ACT-R [3]. GOMSL is an abstract modeling language of the GOMS family that is more powerful than KLM-GOMS, for example, and supports several of ACT-R features features, e.g., representation of mental objects, working memory, primitive internal and external operators, composition methods, or even various flow-of-control constructs [3].

Combining serious gaming and cognitive architectures is not a new concept [28,11]. Gentile et al. (2019) provide an analysis on the potential role of cognitive architectures for serious games, with focus on A.I. and non-player characters (NPC). Such kind of virtual agents are often realized with Soar, for example in the work by Wray et al. (2013) [30]. Mills and Dalgarno (2007) present a conceptual architecture for game-based learning ITS implementations [17]. The authors discuss the role ACT-R for various modeling tasks and how to build declarative and procedural rules. In the context of serious games, we see a further relation to the *Player Experience Modeling* (PEM) by Yannakakis (2012): for personalization, adaptivity engines should be able to recognize and model the learning style and detect the cognitive state of the users [31]. This very detection of the cognitive states is the central aspect of our work and corresponds to the user modeling efforts, as stated in the introduction. ACT-R has been used to develop a virtual agent for use in training simulations for military operations in urban areas [7]. A complex production set was modeled in order to achieve autonomous action by the agent.

Although all these studies are similar in terms of modeling and use cases for serious gaming, we did not find any literature that resembles our approach.

3 Cognitive Architectures & E-Learning

Our cognitive modeling approach for adaptivity is related to modeling for intelligent tutoring systems (ITS). Generally, an ITS architecture includes various models, mainly the user interface model, a student model, a domain model, and a tutor model, all of them with varying interconnections (Fig. 2). In our work we focus on the student model and the domain model since we need information on the cognitive state of the user, and to which concepts or activities in the computer application these states are linked to.

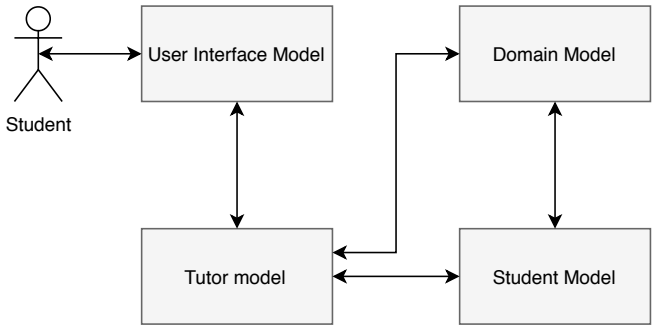

Fig. 2. General scheme of the models in an intelligent tutoring system.

In the context of e-learning, the formal modelling of the cognitive learning state is of particular interest. This includes modelling the cognitive abilities of

humans, including memory, language, perception and problem solving. We focus on approaches to model forgetting or attention. The focus for adaptive knowledge transfer concentrates on the possibility of forgetting: a cognitive architecture can provide the adaptive system with information about content or activities that a user may have forgotten or that are highly likely to be forgotten. One possible adaptation strategy could be to remind and draw attention to these activities. From a neurological point of view this means that the experience and their underlying neuronal ensembles are consolidated (in the long-term memory). Cognitive architectures largely consider such psychological and neuro-scientific principles.

However, there is no precise definition of a "cognitive architecture" and it is often unclear when a concrete (software) instance belongs to the class of cognitive architectures [14]. Cognitive architectures attempt to recreate Artificial General Intelligence (AGI), i.e., computer systems that correspond to human intelligence [19]. According to Russel and Norvig (2009) there are four categories for the realization of AGI [19] that are aimed at replicating:

- systems which think like humans – here we make use of a cognitive architecture with its psychological underpinnings to simulate thinking.
- systems which act like humans – our simulation of the users' thought processes enables an approximation of a human behavior.
- systems which think rationally – since we base our modeling approach only on observations, we depend on a rational behavior by the users.
- systems which act rationally – the simulation process typically is deterministic, therefore the adaptive decision-making processes appear rational.

Cognitive architectures are typically classified according to how they represent and process information. Three paradigms have emerged:

- symbolic, also called cognitivistic: use of explicit symbolic representations to represent information. Cognitivist architectures are also referred to as symbolic architectures and A.I. approaches. Although they are quite successful, they lack generality to be applicable across domains [12].
- emergent, in the sense of connectionism: information is processed in a network of connected computing units that communicate in parallel. The units receive stimuli through their incoming connections, perform nonlinear computation and influence other units through their outgoing connections.
- hybrid, a combination of both.

The most recent overview of the last 40 years of research on cognitive architectures was conducted by Kotseruba and Tsotsos (2018) [14]. They analyzed a set of 84 architectures, of which 49 are still actively developed. They estimate the number of existing cognitive architectures to be around 300. Other comparative reviews have been conducted by Asselman et al. (2015) [6] and Thórisson and Helgasson (2012) [27]. The most prominent representatives of cognitive architectures are the "classical" candidates that have existed since the time of the

emergence of cognitive architecture in the 1970s. Most widely used are the two hybrid architectures ACT-R [4] and Soar [15].

We conducted a subjective value-benefit analysis to select a suitable cognitive architecture for our dynamic cognitive modeling approach. Criteria for the selection process included:

- general applicability and modeling flexibility.
- support for programming languages, maturity of implementing frameworks.
- complexity and learning curve.
- available documentation and community support.
- licensing costs, whereby free, open-source tools are favored.
- number of available scientific publications, indexed by Google Scholar, Web of Science and Scopus.

From our applied research perspective, we focused on the implementation aspects. The result (*cf.* Fig. 3) of our analysis indicated ACT-R as the most versatile and for our approach of dynamic modeling suitable cognitive architecture. The decisive factor for the result was above all the comprehensible code examples of Python ACT-R.

Subjective Rating of **Cognitive Architectures** (Subjective Measures 1-10 increasing)		Weighting (1-10)	ACT-R	LEABRA	SOAR	CLARION	LIDA	ICARUS	ART
Features	Hybrid Character	6	10	2	10	10	10	3	2
	Psychological Foundation	7	10	7	3	7	6	0	3
	Relevance to Image Interpret.	3	0	4	0	0	0	0	10
	AGI	4	9	4	9	3	5	8	7
	Weighted Category Sum	7	8,3	4,5	5,9	6,1	6,1	2,5	4,6
Implementation	Complexity	4	2	4	3	3	1	3	1
	Programming Languages	9	7	5	5	5	7	4	5
	Weighted Category Sum	4	5,5	4,7	4,4	4,4	5,2	3,7	3,8
Score (weighted)			**7,3**	**4,5**	**5,3**	**5,4**	**5,8**	**2,9**	**4,3**

Fig. 3. Results of our subjective value-benefit analysis.

ACT-R tries to implement the formal abstraction of human behavior with a module concept. At its core is the procedural module. The information generated in the surrounding modules flows into this module and is then processed so that the next action step of an agent can be determined. Production rules in ACT-R thus serve as circuit functions that map certain information patterns in the modules' memories, called buffers, to changes in buffer contents [4]. We make use of these declarative and goal modules. In the declarative module, knowledge is stored in the form of chunks. These chunks can be retrieved using productions

if the goal buffer is set to a specific chunk. Therefore, the goal module serves as a state manager to determine which productions will be run in each simulation step. Based on ACT-R theory, various calculations are performed to find out which actions in the other modules, which are involved in a particular production, should be performed to fulfill the currently set goal. Section 5 explains the dynamic generation of production rules and declarative chunks.

4 Concept for User Modeling and Artificial Intelligence (UMAI)

We developed a concept for user modeling with A.I., named *User Modeling and Artificial Intelligence* (UMAI). The A.I. part is the cognitive modeling by integrating cognitive architectures for the modeling process. As stated before, our concept targets applied research and therefore the direct applicability of cognitive architectures for assistance systems. We make use of triple-structured activity data streams which can be used to track almost every user interaction event. In this article we focus on the cognitive architecture ACT-R for educational serious games. A key aspect is the dynamic generation of the ACT-R model based on observed user interaction data. The output contains estimated memory or chunk activation levels which an adaptive system can incorporate in its decision processes.

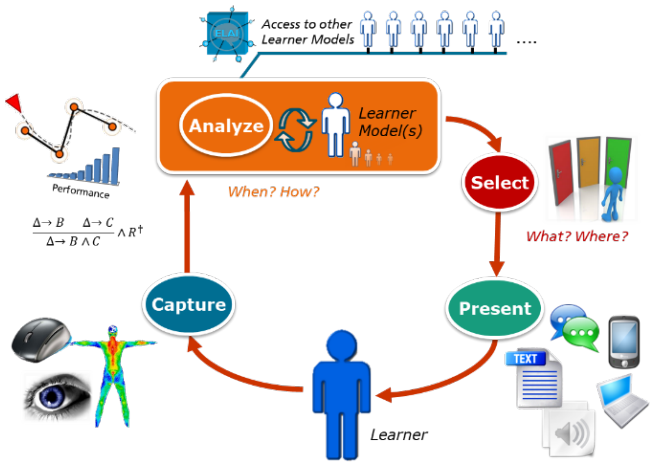

Fig. 4. Extended 4-phased adaptivity cycle (based on [22]) with combined analysis-learner-models-phase.

Our view on adaptive systems follows the 4-phased adaptivity cycle by Shute et al. (2012) [22]. The cycle depicted in Fig. 4 structures an adaptivity process in four consecutive phases or stages where each new run depends upon the previous

run, hence forming a cycle. Its main components are the four phases (1) capture, (2) analysis, (3) select and (4) present, plus an additional user or learner model after the analysis phase. However, we incorporate the user models into the analysis phase (2). The argumentation is that the select phase (3) not only builds upon and uses the user models but also incorporates additional analysis results, such as usage pathways models. UMAI is located in the analysis phase (second phase) and it contributes to the formation of the user or learner models. A special feature of our approach is the dynamic modeling of the production rules. At runtime, an ACT-R user model is generated based on interaction data which is captured using the standard protocol xAPI. The interaction data is translated into ACT-R production rules for the virtual procedural ACT-R memory. Based on this model, cognitive processes can be simulated and analyzed. We make use of this to query the cognitive architecture which concept the user probably has "forgotten". ACT-R allows to query the activation energy of the modelled concepts. This activation energy, which is based on the Memory Decay Theory, can be mapped to a continuous, real number and determined by the cognitive architecture per simulation time step. Important game concepts, which are neglected by the user, can be determined by the decreasing activation energy. A threshold function is used to model "forgetting" and an adaptivity system can react to this situation and offer appropriate assistance [24].

The general process is as follows (*cf.* Fig. 5):

1. The main UMAI program is started. The start request includes a user identifier for which the cognitive modeling and simulation should take place.
2. Retrieve xAPI statements from an xAPI Learning Record Store (LRS). The query typically includes filtering on the active user and on his latest usage session. This is achieved through recording an *initialized*-statement for events like a beginning a new level or starting a session.
3. Execute the model generator which takes the recorded activities (from xAPI) and generates a simulation program. This dynamic generation of an ACT-R program containing a user model depicts our dynamic modeling approach (implementation details in section 5).
4. The simulation program is started and simulates the various steps the user has undertaken. In accordance with classical cognitive modeling the simulation builds a cognitive model on the sequence of user actions. By using a cognitive architecture such as ACT-R for storing and processing the events we can determine cognitive variables such as chunk activation levels (e.g., memorization level of an activity).
5. For each observed activity a chunk is generated, and an activation value is computed using ACT-R.
6. Thresholding on the activation levels produce those chunks for which repeated attention is recommended.
7. The main program returns the selected activities and their activation levels.

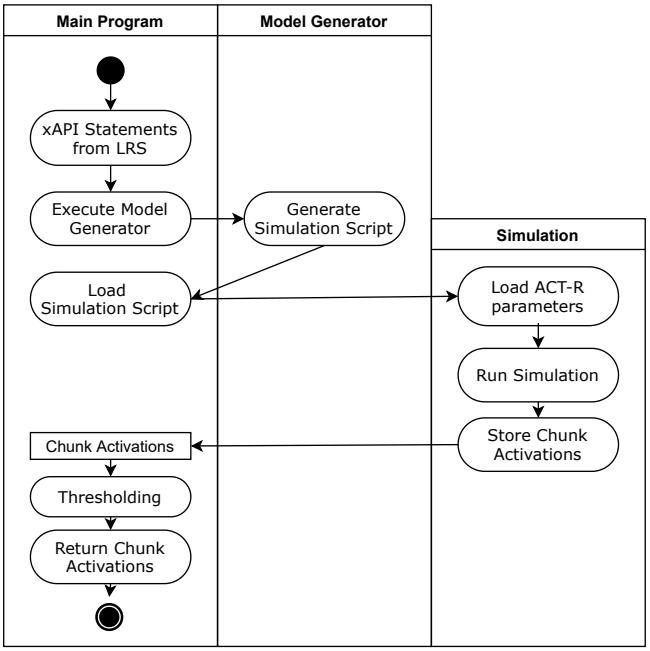

Fig. 5. Model generation process for one simulation request to the UMAI system.

5 Software Architecture, Implementation

Interactions with the game are transferred via xAPI to an LRS. This interaction data together with the data in the declarative and procedural module of ACT-R forms the learner model or student model.

The Experience API (xAPI) standard [1] allows to monitor a user's varying experiences in learning systems, ideally in a consistent format [21,1]. For general application the related W3C activity stream standard [23] works similarly, from a technical point of view, but with conceptual differences [8]. For implementation, one needs to build an activity stream or xAPI adapter which tracks selected events (many libraries for different programming languages exist). The choice of events should be aligned at the overall analysis goal or the overall research question [21].

In the designed software architecture (Fig. 6) the "intelligent user model" is implemented as a micro service, and the communication with connected assistance systems is done via HTTP RESTful services.

After each user action, the xAPI adapter generates an event statement and sends it to the LRS. On request from the user (student or tutor) UMAI retrieves all recorded statements from the beginning of the session. Subsequently, a retroactive simulation is started in the cognitive architecture.

This simulation can be repeated as often as desired to model different configuration variants, e.g., individualized memory decay and threshold parameters.

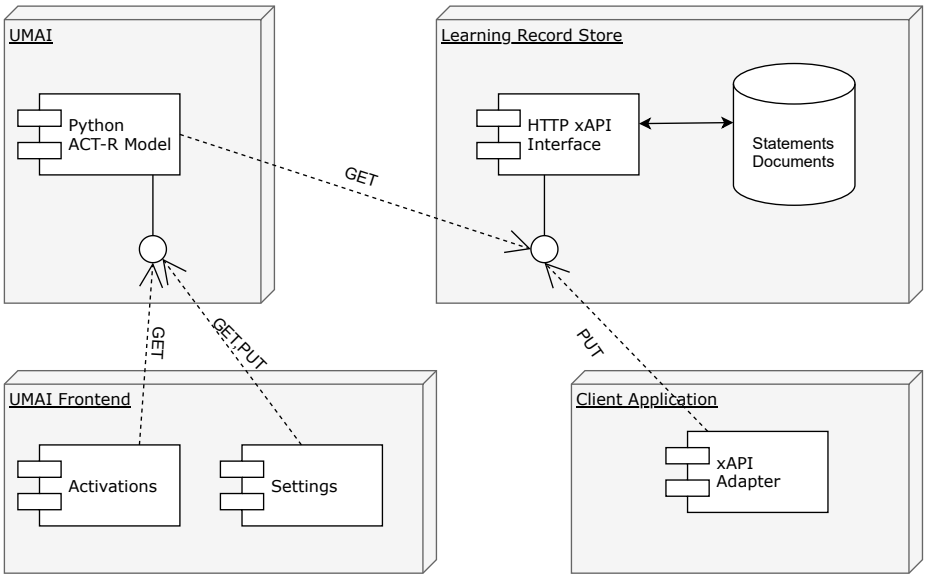

Fig. 6. System Architecture for the UMAI concept and implementation.

```
def action_i(focus='action_i'):
    if (action_stream[counter] in new_actions):
        new_actions.remove(action_stream[counter])
        dmstring = 'action_i:' + action_stream[counter]
        DM.add(dmstring)
        focus.set('action_' + str(conuter+1))
    else :
        DM.request(dmstring)
        focus.set('action_' + current_chunk_number + '_')

def forgot_i(focus='action_i_', DMBuffer=None, DM='error:True'):
    print ("I forgot something")
    if len(action_stream) == counter :
        focus.set('stop')
    else :
        fired_actions[str(counter)]= action_stream[counter]
        focus.set('action_' + str(counter+1))

def remember_i(focus='action_i_', DMBuffer="action_i:?action"):
    print ("I remember action_i:",action)
    if len(action_stream) == counter :
        focus.set('stop')
    else :
        DM.add('action_0:?action')
        DMBuffer.clear()
        focus.set('action_' + str(counter+1))
```

Fig. 7. Python ACT-R code for dynamically creating production rules.

After each simulation, data accumulated in the cognitive architecture can be used for subsequent adaptivity decisions, e.g., offering recommendations or for dynamic difficulty adjustment. For learning analytics purposes, we can visualize the chunk activation values for each observation step (line diagram Fig. 9).

Once the activity stream from the session start to the last fired action is retrieved, an ACT-R agent is run on a production set which is dynamically generated from the stream.

In the first step the xAPI statements are parsed into an action stream array where the sequential order is maintained. Our modeling in ACT-R is based on the resulting stream and is independent of the domain of the underlying learning system. The methodology analyzes the occurrence and frequency of action steps as time series. As described in section 3, in ACT-R each chunk is assigned an activity rate which degenerates over time. Chunks of declarative memory are dynamically created for each of these action descriptions by dynamically creating code fragments. An example for the code template is shown in Fig. 7.

If actions occur more than once and chunks have already been created for them during the processing of the action stream, queries are made to the declarative module. These requests may fail if the activity rate of the respective chunk of the correlating action is below a pre-defined threshold.

The processing logic is as follows (*cf.* Fig. 7):

- For each action, a base production is first generated.
- If an action has been executed for the first time it is added to the declarative memory as a chunk. Also, an identifying number is stored in a `chunk_numbers` array and the focus buffer is set to the subsequent action.
- If an action occurred previously its chunk is retrieved from the declarative memory and the focus buffer is set to the current chunk number.
- For each action, the production is called only once and a chunk of the form `action_0:action_descriptor` is created. In Python ACT-R, a request is made to a declarative chunk by passing `DM.request()` to the function `action_0:?action`. Before this request can be sent, the chunk number is determined when the particular action first occurred.
- Afterwards, the focus is not set to the next action, but to the production that matches the chunk number. This is therefore the production at the point where the action occurred for the first time. The focus is not set on the base production, but on the productions `forget` and `remember`, which are also generated for each action (Fig. 7).
- In case the declarative action-chunks have an activation less than or equal to the `threshold` and the action is called, the forget-production is called. Here, within the production rule, only the focus is set on the serial action. If the activation is greater than the `threshold`, the `remember` production is called. In case of calling the forget-production, the respective chunk cannot be reactivated.
- In the `remember` production it is first checked whether all actions have been processed. If this is the case, the focus is set to `stop` so that the simulation of the agent can be ended in the next step with the `stop` production. If

the processing status of the action stream has not yet reached the end, the focus is set on the next action. In contrast to the original implementation, in Python ACT-R it is necessary to add the called action to the declarative memory again in order to increase the activation of the respective called chunk.

6 Application & Results

For technical verification of the UMAI concept we implemented it for our serious game *Exercise Trainer* (EXTRA) [25] (Fig. 1 & 8). An adaptive assistance system [24] linked to EXTRA is meant to point out little noticed – but essential – concepts to the user. The UMAI concept can provide the necessary modeling for the adaptivity response model.

In the following section, we briefly explain EXTRA, then we describe how EXTRA relates to the UMAI concept and its application.

6.1 Serious Game: Exercise Trainer (EXTRA)

The game concept of EXTRA is designed as an isometric, turn-based simulation game (Fig. 8). The general form of the game is geared towards process training games which are related to logistics or business processes. The learning objective is to learn the actors and relationships involved in complex processes. The application background is technically complex IT scenarios with different subsystems, inputs and outputs. In EXTRA, the complex roles, activities and processes as well as the technical system-of-systems structure are abstracted into a flexible game world and described by metaphors. For this purpose, the player must build logistics chains with factories and infrastructure (*cf.* Fig. 8). The learning objects are interwoven with the gameplay so as not to compromise immersion. The terminology and the properties of the factories or connections are therefore based on real systems, and the game supports training and receptive knowledge transfer in a transparent way. The didactic goal is that essential components and relationships of complex system-to-system structures are being learned. An application example is the training of a process for image-based intelligence in which the users experience the varying activities of tasking, collection, processing, exploitation and information dissemination. The actual game goal is to satisfy demanding "customers" with changing product desires (metaphor for tasks) by constructing optimal logistic chains (metaphor for data or information links) to optimally distribute products to markets. Gamification approaches such as high-scores help to motivate players to compete with others and to repeat playing the game. The score reflects the degree of success in satisfying customer needs. If the requirements are not met in time, the score decreases; if the score drops to zero, the game is lost.

Fig. 8. The serious game Exercise Trainer (EXTRA) [25] for playful learning and training of complex processes.

6.2 Application of UMAI

To verify the UMAI concept we focused on four essential EXTRA activities (*cf.* Fig. 8) which the user can perform, and the sequence and timing of which can be arbitrary:

- place street road: the user connects two stations in the supply chain with a street. Either a street or a waterway must be placed, the type depends on the player's strategy.
- place waterway: like streets but with higher capacity although slower.
- increase workforce: the user increases the workforce at a factory in the supply chain to speed up the production and decrease the time to fulfil the customers' needs. To successfully master a level this activity should be used.
- sell images: once the user has established a fully functioning supply chain, he is able to sell fabricated products, in this case imagery products. To sell products he must click on an icon to place them in the distribution center (the market). This is a required activity.

Each of these activities or events trigger sending of corresponding xAPI statements which are used by UMAI as input data in order to build the ACT-R agent.

As shown in Fig. 5 the dynamic modelling is based on a file generator which makes use of a Python template. From this template a simulation script is generated and run to calculate and return chunk activation values to an API endpoint. This enables not only reporting use cases but also adaptive tutoring if the activity rates are sent back to a tutoring model.

Figure 9 shows a report of activity rates which was recorded in EXTRA. In this session the player started a level by connecting a factory to a market with a street. This allows to bring products from the factory to the market. In the next step he sold products at the market. In UMAI a chunk in the declarative module is generated for each of the actions, the user has performed. After each iteration

the activation level of these chunks is stored. The spikes in the diagram represent an initial performing of an action or a memory retrieval. A good example would be the tracking of the chunk which represents selling of products (bottom green line in Fig. 9). If the threshold would be set to 0.75 the chunk would be forgotten in the simulation and the corresponding production would have been set by the goal buffer as described in section 5.

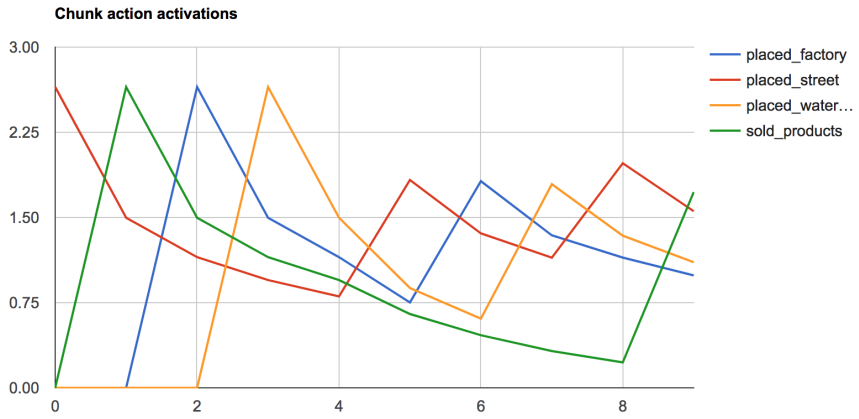

Fig. 9. Chunk activation values for 4 selected activities in the EXTRA game. The bottom green line shows the memory degradation computed by ACT-R for the neglected activity "sell products". In step 7 an adaptive hint was displayed which refreshed the chunk and its activation value.

The implemented architecture has been extended to provide adaptive hints, as shown in Fig. 1. To accomplish this, the tutoring model was extended to send a request to UMAI after certain events or in a time interval. After every request UMAI will simulate the current session and returns the corresponding activity values. The computation of the simulation in our case took less than one second. Based on thresholding the tutoring model uses these results to offer preemptive hints to support the user.

Although the concept has been applied successfully, the technical study revealed some limitations in respect to the power of the resulting model. As stated, the complex and time-consuming modelling of declarative memory (factual knowledge) or production rules (action knowledge) makes direct application unattractive. Whilst our solution approach addresses this issue by the dynamic generation of procedural rules, real “intelligence” is not obvious. The intelligence lies in the underlying mechanisms of cognitive architecture and its implementation. The presented modelling of “forgetting” by means of ACT-R can, considered on its own, be realized more easily without a cognitive architecture. However, this is only one aspect and ACT-R offers many more features that can be used

for adaptivity decision making, such as cognitive load measurement or semantic cognitive association tracking.

7 Conclusion

We developed a concept for data-driven user modeling with the cognitive architecture ACT-R. The concept targets applied research and the direct applicability of cognitive architectures for assistance systems. We make use of triple-structured activity data streams which can be used to track user interaction events.

The research question is how cognitive modeling can contribute to "intelligent user models" and how it can be made operational for adaptivity.

The contribution of this work is a concept for data-driven, cognitive user models for digital learning games by means of dynamic cognitive modelling and its application to adaptivity.

We focus on the cognitive architecture ACT-R and its application for adaptive serious games. A key aspect is the data-driven, dynamic generation of the ACT-R model based on observed user interaction data. The output contains estimated memory or chunk activation levels which an adaptivity system can incorporate in its decision processes, e.g., to find the right point in time to offer context-sensitive recommendations.

The concept has been applied to a turn-based serious game in which an adaptivity component in form of a virtual agent giving hints is controlled by the cognitive model. Although the concept has been applied successfully, the technical study will be developed further to incorporate the full potential of the cognitive architecture.

Future work aims at the application to other serious games and assistance systems. Further research looks at Hierarchical Bayesian Models for cognitive models to estimate latent cognitive variables such as workload, attention, planing, perceived difficulty or forgetfulness.

Acknowledgments The underlying project to this article is funded by the Federal Office of Bundeswehr Equipment, Information Technology and In-Service Support under promotional references.

References

1. ADL.net: Experience API (xAPI) specification 1.0.3, `https://github.com/adlnet/xAPI-Spec/blob/xAPI-1.0.3/xAPI-Data.md`, publisher: ADL
2. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **4053 LNCS**, 61–70 (2006). https://doi.org/10.1007/11774303_7
3. Amant, R., Ritter, F.: Automated GOMS–to–ACT-r model generation. Proceedings of the 6. ICCM, Mahway, NJ: ... p. 6 (2004)

4. Anderson, J., Christian, L., Taatgen, N., Sun, R.: Modeling paradigms in ACT-r. In: Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation, pp. 29–52. Cambridge University Press (2006)
5. Anderson, J.R., Gluck, K.: What role do cognitive architectures play in intelligent tutoring systems. Cognition & Instruction: Twenty-five years of progress pp. 227–262 (2001)
6. Asselman, A., Aammou, S., Nasseh, A.E.: Comparative study of cognitive architectures. International Research Journal of Computer Science (IRJCS) Issue **9**(2), 8–13 (2015)
7. Best, B.J., Lebiere, C., Scarpinatto, K.C.: Modeling synthetic opponents in MOUT training simulations using the ACT-R cognitive architecture. Proceedings of the 11th Computer Generated Forces Conference pp. 2–56,505–516 (2002)
8. Bowe, M.: Tin Can vs. Activity Streams (2013), `https://tincanapi.com/tin-can-vs-activity-streams/`
9. Brusilovsky, P., Millán, E.: User models for adaptive hypermedia and adaptive educational systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web: Methods and Strategies of Web Personalization, pp. 3–53. Lecture Notes in Computer Science, Springer (2007). https://doi.org/10.1007/978-3-540-72079-9_1
10. Dörner, R., Göbel, S., Effelsberg, W., Wiemeyer, J.: Serious Games - Foundations, Concepts and Practice. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-40612-1
11. Gentile, M., Città, G., Lieto, A., Allegra, M.: Some Notes on the Possibile Role of Cognitive Architectures in Serious Games. In: Liapis, A., Yannakakis, G.N., Gentile, M., Ninaus, M. (eds.) Games and Learning Alliance. pp. 231–241. Lecture Notes in Computer Science, Springer International Publishing, Cham (2019)
12. Gudivada, V.N.: Cognitive computing: Concepts, architectures, systems, and applications. In: Gudivada, V.N., Raghavan, V.V., Govindaraju, V., Rao, C.R. (eds.) Handbook of Statistics, Cognitive Computing: Theory and Applications, vol. 35, pp. 3–38. Elsevier (2016). https://doi.org/10.1016/bs.host.2016.07.004
13. Kiili, K., de Freitas, S., Arnab, S., Lainema, T.: The design principles for flow experience in educational games. Procedia Computer Science **15**, 78–91 (2012). https://doi.org/10.1016/j.procs.2012.10.060
14. Kotseruba, I., Tsotsos, J.K.: 40 years of cognitive architectures: Core cognitive abilities and practical applications. Artificial Intelligence Review (Jul 2018). https://doi.org/10.1007/s10462-018-9646-y
15. Laird, J.E., Newell, A., Rosenbloom, P.S., Nilsson, N., Bobrow, D.G., Laird, J.E., Al, E.: SOAR: An architecture for general intelligence. Arttfictal Intelhgence **33**, 1–64 (1987). https://doi.org/10/ccvjs7
16. Ma, W., Adesope, O.O., Nesbit, J.C., Liu, Q.: Intelligent tutoring systems and learning outcomes: A meta-analysis. Journal of Educational Psychology **106**(4), 901–918 (2014). https://doi.org/10.1037/a0037123
17. Mills, C., D, B.: A conceptual model for game based intelligent tutoring systems. In: Proceedings of ASCILITE - Australian Society for Computers in Learning in Tertiary Education Annual Conference 2007. pp. 692–702. Australasian Society for Computers in Learning in Tertiary Education (2007)
18. Prensky, M.: Digital game-based learning, vol. 1. Paragon House (2003). https://doi.org/10.1145/950566.950596
19. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach (3rd Edition). Prentice Hall (2009)

20. Salvucci, D.D., Lee, F.J.: Simple cognitive modeling in a complex cognitive architecture. In: Proceedings of the conference on Human factors in computing systems - CHI '03. p. 265. ACM Press (2003). https://doi.org/10.1145/642611.642658
21. Serrano-Laguna, A., Martínez-Ortiz, I., Haag, J., Regan, D., Johnson, A., Fernández-Manjón, B.: Applying standards to systematize learning analytics in serious games. Computer Standards & Interfaces **50**, 116–123 (2017)
22. Shute, V., Zapata-Rivera, D.: Adaptive educational systems. Adaptive technologies for training and education **7**(1), 1–35 (2012). https://doi.org/10.1017/CBO9781139049580.004
23. Snell, J., Prodromou, E.: Activity streams 2.0 (2017), `https://www.w3.org/TR/2017/REC-activitystreams-core-20170523/`
24. Streicher, A., Roller, W.: Interoperable adaptivity and learning analytics for serious games in image interpretation. In: Lavoué É., H., D., K., V., J., B., M., P.S. (eds.) Data Driven Approaches in Digital Education: 12th European Conference on Technology Enhanced Learning, EC-TEL 2017, Proceedings. vol. 10474 LNCS, pp. 598–601. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-66610-5_71
25. Streicher, A., Szentes, D., Gundermann, A.: Game-based training for complex multi-institutional exercises of joint forces. In: Verbert, K., Sharples, M., Klobučar, T. (eds.) Adaptive and Adaptable Learning: 11th European Conference on Technology Enhanced Learning, EC-TEL 2016, Lyon, France, September 13-16, 2016, Proceedings. pp. 497–502. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-45153-4_49
26. Taatgen, N., Anderson, J.R.: The past, present, and future of cognitive architectures. Topics in Cognitive Science **2**(4), 693–704 (2010). https://doi.org/https://doi.org/10.1111/j.1756-8765.2009.01063.x
27. Thórisson, K., Helgasson, H.: Cognitive architectures and autonomy: A comparative review. Journal of Artificial General Intelligence **3**(2), 1–30 (2012)
28. Van Eck, R.: Building artificially intelligent learning games. In: Gibson, D., Aldrich, C., Prensky, M. (eds.) Games and Simulations in Online Learning: Research and Development Frameworks, pp. 271–307. IGI Global (2007). https://doi.org/10.4018/978-1-59904-304-3.ch014
29. Woolf, B.P.: Building Intelligent Interactive Tutors. Morgan Kaufmann (2009)
30. Wray, R.E., Woods, A.: A Cognitive Systems Approach to Tailoring Learner Practice. Proceedings of the Second Annual Conference on Advances in Cognitive Systems ACS **21**, 18 (2013)
31. Yannakakis, G.N.: Game AI revisited. In: Proceedings of the 9th Conference on Computing Frontiers - CF '12. p. 285. ACM Press, Cagliari, Italy (2012). https://doi.org/10/ggctr7