Design of a Neuronal Training Modeling Language

Exemplified with the AI-Based Dynamic GUI Adaptation

Marcus Grum[1*[0000-0002-3815-4238]], Werner Hiessl[2], Karl Maresch[3], Norbert Gronau[1]

[1] University of Potsdam, Potsdam 14482, Germany
[2] Asseco Solutions AG, Germany
[3] Salesbeat GmbH, Welser Str. 26, 4060 Leonding, Austria
[*] mgrum@lswi.de

Abstract. As the complexity of learning task requirements, computer infrastructures and knowledge acquisition for artificial neuronal networks (ANN) is increasing, it is challenging to talk about ANN without creating misunderstandings. An efficient, transparent and failure-free design of learning tasks by models is not supported by any tool at all. For this purpose, particular the consideration of data, information and knowledge on the base of an integration with knowledge-intensive business process models and a process-oriented knowledge management are attractive. With the aim of making the design of learning tasks expressible by models, this paper proposes a graphical modeling language called *Neuronal Training Modeling Language* (NTML), which allows the repetitive use of learning designs. An example ANN project of AI-based dynamic GUI adaptation exemplifies its use as a first demonstration.

Keywords: AI and Business Informatics, Development of AI-based Systems, AI-Based Decision Support System, Cooperative AI (Human-In-The-Loop), Process-oriented Knowledge Acquisition, Modeling Language.

1 Introduction

Faced with an increasing complexity of IT infrastructures [1], providing greater numbers of autonomous systems and decentralized data sources because of Internet of Things and cloud computing [2], learning tasks in the sense of Artificial Intelligence (AI) are becoming more complex to design: the location of data occurrence, knowledge generation and knowledge use are decoupled. Beside the integration of knowledge carriers, companies are often challenged regarding the acquisition and harmonization of knowledge, information and data for the training design for Artificial Neuronal Networks (ANN). Regrettably, prominent representatives of programming libraries and tools do not support the specification of training and testing sets from various sources and from different knowledge carriers. Further, they do not consider underlying (business) process models. A graphical notation in the sense of a modeling language is only provided by the *CoNM* [3], which is overwhelming because of numerous modeling items and perspectives but technically satisfying. Hence, an efficient communication

with customers about training runs for ANN is lacking and the need for a neuronal training modeling language (NTML) becomes apparent.

Although not limited to it, ERP contexts in particular are suited for the demonstration of this NTML because first, ERP software directly supports business processes and second, this kind of software refers to applications that are able to flexibly adapt to its current software users with the aid of ANN. Here, the Graphical User Interface (GUI) serves as target for learning. It is understood as a kind of dashboard, which is decomposed to GUI elements. As this adaptation can satisfy various dimensions, this contribution focuses on the adaptation of the software's appearance in regard with GUI elements. Hence, the question gains in significance how a dynamic GUI adaptation can be specified as learning task for ANN as well as how a learning task can be visualized in order to support an efficient communication about ANN designs and corresponding results. In this context, the following research questions are pertinent:

1. How can AI-based learning tasks be specified with the aid of a modeling language?
2. How can dynamic software interface be optimized by AI-based GUI adaptations?
3. How can AI-based recommendations be considered without depriving the responsibility of software users and vendors?

The questions gain in significance in regard with usability contexts as inefficient software interface designs worsen the performance of software users. This paper intends not to discuss concrete ANN architectures. It focuses on the effective modeling of learning tasks for the sophisticated *CoNM* tool use.

According to the DSRM of [4, 5], the remainder structures as follows. The second section presents the theoretical foundation of an dynamic GUI adaptation as well as training specification and it identifies current tools as inadequate to specify both. The third section provides a methodological proceeding, which will be applied for the demonstration of the NTML design. The fourth section designs artefacts required for the specification of a dynamic GUI adaptation learning task. In the fifth section, artefacts are demonstrated with the aid of an ERP prototype software. While its evaluation is presented in a fifth section, the final section concludes the paper.

2 Theoretical Foundation

AI Learning and ANN. State-of-the-art representatives of programming libraries, such as *Theano* [6], *Caffe* [7], *PyTorch [8], TensorFlow* [9], provide graphical representation mechanisms for ANN implemented, which are more or less interactive. Modeling notations for ANN, such as *LEMS* [10] or *NeuroML* [11] enable the graphical specification of ANN, but do not consider technical implementations adequately. Since only the *CoNM* [3] can be considered as graphical ANN modeling approach, that results in ANN implementations adequately and enables the interactive specification of (a) underlying (business) process models, (b) organizational knowledge bases and (c) relevant ANN systems, its Neuronal Modeling and Description Language (*NMDL*) seems to be an adequate foundation for this contribution. However, as the NMDL enables the sophisticated specification of required parameters for various domains, such as the concrete

training course, specific testing strategies, their evaluation and competence-oriented verification in simulation runs and corresponding model management as well as their biological compartments, the language designed here focuses on the training only.

Dynamic GUIs. Graphical user interfaces (GUI) provide software interfaces that follow underlying software process models: if a certain button is activated or the software state changes, the current view changes and dynamically provides relevant GUI elements. Advanced systems reflect (business) process progresses on behalf of messages sent by workflow engines [12]. An application providing an adaptable GUI, which builds on AI-based recommendations, and adapts to arbitrary many criteria, such as the user's preferences and process progresses, therefore provides a new level of flexibility.

Knowledge & Process Modeling Languages. Prominent representatives of (business) process and knowledge modeling languages enable the specification of many perspectives, such as the behavioral, functional perspective, informational, knowledge, organizational and communication perspectives [13–15]. Here, one can find the *eEPC* [16], *BPMN* [17], *UML* 2 [18] as process modeling languages and the *Promote* [19], *GPO-WM* [20], *KMDL* [21], *NMDL* [3] as process-oriented knowledge modeling languages.

As only the NMDL modeling approach considers both, business processes as well as knowledge for the specification of ANN learning, the following therefore builds on the most current and sophisticated version of the NMDL. It is the only representative providing a meta-model that satisfies the sophisticated collection of common perspectives presented [15]. However, the following focuses on the minimal set of modeling items required for the effective learning task specification.

Research Gap. A modeling language having the minimal set of modeling items at all, enabling the interactive explication of ANN learning tasks on the base of underlying (business) process and knowledge models has not been realized, yet. Further, an ANN visualization enabling the interactive explication of the question, how knowledge is acquired and in which process knowledge generated by ANN is used, is still missing at a minimal set of modeling items, too. In their joint consideration, the research gap for this contribution becomes apparent.

3 Method

Methodology. The methodological approach of this research contribution follows the design-science-oriented method of Peffers et al. [4, 5]. As we suggest for the definition of learning tasks, such as the improvement of inefficient GUI designs, a model for the proceeding is established, that considers knowledge for the specification of learning tasks in the sense of knowledge management with the aid of a modeling language.

Proceeding. The procedure model has been used for demonstration purposes in this contribution. Similar to [1], one can observe seven sequential phases, although each

phase allows the returning for previous phases. So, an iterative proceeding is enabled, which supports the failure tolerant modeling close to real-world.

The *first phase* refers to the acquisition of ANN projects, and includes contractual issues, sales, and marketing, among others.

The *second phase* defines the focus of the learning task. This includes definitions about temporal dynamics, learning modes (supervised, unsupervised and reinforced learning) [22], codification schemes [23], learning problems (clustering, classification and regression) [24] and the intended ANN performance.

Within the *third phase*, the modeling of knowledge-intensive processes is realized, which will serve as foundation for the definition of learning tasks. Here, processes are selected, which are relevant for the specification of the learning task intended.

In the *fourth phase*, the knowledge generation is specified. This issues questions about knowledge carriers, the identification of information systems as knowledge bases and the characterization of knowledge objects. Further, the knowledge use is specified, so that it becomes clear, how ANN-based results are considered for the realization of processes, such as business processes or processes about the adaptable GUI design.

In a *fifth phase*, the learning task can be specified with the aid of models constructed so far. This will be the foundation for the automated data collection and set realization, so that a learning can be carried out efficiently.

In a *sixth phase*, the ANN technique, will be trained, tested and evaluated, up to the point the AI satisfies the performance agreed with. As this phase probably demands for specialized proceedings supporting individual ANN technique aspects, specialized proceedings can be nested here in the sense of a mixed method [25, 26].

Lastly, in a *seventh phase*, the AI system is continuously run, maintained and improved in the sense of CIP [27].

4 Design

The design of a modeling language enabling the specification of knowledge use for ANN learning tasks is realized as follows. First, in the sense of the DSRM, requirements are collected, which are independent form the design. Intending to consider requirements collected and design decisions made (section 4.1), the sections 4.2 and 4.3 generate a solution, which will be applied to a demonstration example in the fifth section.

4.1 Requirement Collection

Demands for the modeling of an AI-based dynamic GUI adaptation are grouped by individual design artefacts. Here, one can find the language aiming to specify learning tasks. Further, there are requirements focusing on a concept for the GUI adaptation.

Modeling Language Requirements. Focusing on the specification of the training modeling language, the design needs to satisfy research consensus about process modeling language construction. Further, it needs to enable the state-of-the-art of ANN training. By this, the design needs to…

1. …follow a process modeling language meta-model. Here, any relation will be visualized, so that modeling rules become apparent.
2. …consider the current design of an organization's business processes. Here, any kind of knowledge-intensive process relevant for a learning tasks is issued.
3. …make the organization's current knowledge base accessible for the learning. Here, any kind of knowledge carrier and various knowledge forms are issued.
4. …characterize the knowledge generation, which considers dynamic aspects of knowledge and so complements the current knowledge base.
5. …specify learning tasks, so that the learning objective becomes apparent. This includes the specification of training and test sets and issues the data, information and knowledge use for learning tasks.
6. …characterize the knowledge use, so that it becomes clear in which context and in which business process of an organization AI-based results are applied.

GUI Requirements. Focusing on the specification of the AI-based dynamic GUI adaptation, the design needs to satisfy the research consensus about software interface construction. Further, it needs to apply state-of-the-art ANN training. By this, the design needs to…

7. …consider the user's current process context. Particularly, ERP systems here provide a sophisticated base of business process descriptions. Just because of the view of a user on a process-specific software interface, a current state within a process system can be derived.
8. …consider the user's current profile (industry, company, domain, etc.).
9. …consider the user's interaction with the software. Each click, mouse movement or GUI element-specific change is valuable in the context of a learning task.
10. …consider the vendor's expertise, who has gained a high specialization in various industries and is able to provide optimal designs. Hence, aside of an AI-based recommendation, a vendor-based adaptation is attractive.
11. …consider the community's recommendations, who have collected experiences in the daily software use. Hence, aside of an AI-based recommendation, a community-based adaptation is attractive.
12. …allow the software user's decision for or against an adaptation. This refers to a self-determined and self-responsible software use, so that the software user is not incapacitated and overrun by an AI system.
13. …consider the transfer of knowledge about GUI elements. This refers to first, the transfer of GUI elements and second, the transfer of knowledge about the use of GUI elements.

Design Decisions. Going beyond the definition of requirements, that an artefact needs to satisfy in order to serve as problem solution, a collection of design decisions limits the degree of freedom artefacts can follow for the course of requirement satisfaction. Here, one can find the following:

1. Each visual element is considered as process instance. This enables GUI elements to be dynamic and to be modeled by a modeling language.
2. Each process instance follows an individual life cycle. This allows GUI elements to be brought to life and to have a lifespan.
3. Knowledge can be derived by an creation of a process instance. So, others can profit from someone's positive attitude about a GUI element.
4. Knowledge can be derived by an destruction of a process instance. So, others can profit from someone's negative attitude about a GUI element.
5. Knowledge can be derived by any modification of a process instance. This includes changes in the general software interface design as well as changes in the current software interface configuration, so that other users can profit from someone's GUI element configuration.

4.2 Design of an Adaptive GUI

The design component of an adaptive GUI refers to a concept, that enables the flexible adaptation of software interface elements in general. For this, each GUI element is considered as individual process having a lifespan. As the adaptation of GUI elements shall include knowledge from various kinds of knowledge carriers, further, a concept for the abstract knowledge use is designed thereafter.

Adaptivity by Life Cycle Design. Each software interface element, that is intended to be part of an adaptation process, is assumed to be dynamic and therefore can be modeled with the aid of a process modeling language. The dynamic process understanding is operationalized by a life span, which starts with the placement within the software interface and ends with the deletion from the software interface. The process of this life cycle form the view point of a GUI element, is visualized in Fig. 1.

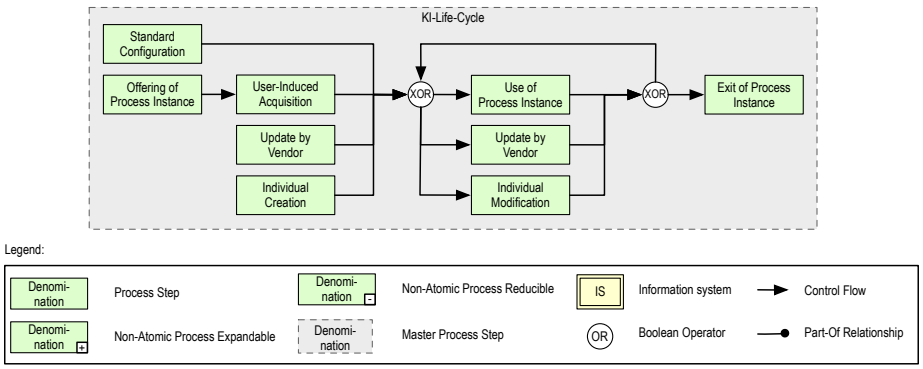

Fig. 1. Life-Cycle for Visual Process Instances (Process View).

Either an element is brought to the software interface as part of the standard configuration of a software, which is determined by a software vendor, or it is brought to the software interface latterly by three options: First, the software user acquires software interface elements by decision from a pool of software interface elements offered. These can be recommended by an AI-based system, by the vendor or by the software user community. Second, it is updated by the vendor e.g. because an adaptation goes along with security relevant changes. Here, the software is updated by decision of the vendor. Third, software interface elements are created by the individual software user. As the creation might help others to optimize their current software interface, by decision, the software user can transfer software interface elements to the community.

Abstract Knowledge Use Design. Independent from the question who constructs a certain software interface element, one and the same software interface element will appear in a different form, when it is included at a user's software interface. This is because it will operate on a different data base. Hence, for the provision, software interface elements are abstracted, so that the core functionality is specified. Later, for the recommendation, the software interface element is provided as abstract concept, that adapts to the individual software context of a user.

4.3 Design of an Neuronal Training Language

The design component of a modeling language, that is able to flexibly and efficiently specify learning tasks, such as the adaptive software interface element recommendation, is carried out in the following. Fig. 2 presents its meta-model, which shows any relation possible. Each view is exemplified in the following.

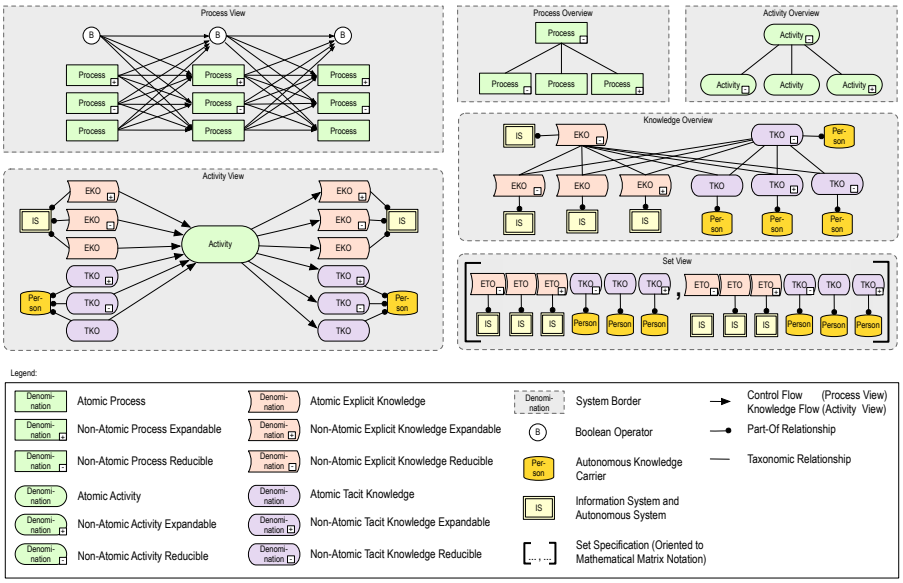

Fig. 2. Meta-Model of Neuronal Training Modeling Language.

Process View. The behavioral perspective on business processes and processes specifying the routing through a software system is specified by the *Process View*. These consist of a sequence of modeling elements, such as process steps, sometimes referred to as *tasks*, Boolean operators as well as a control flow relating modeling elements. Further, this perspective specifies the sequential proceeding of the training of AI-based systems.

Process Overview. Processes are collected by a taxonomic *Process Overview*. For the purpose of AI-based GUI applications, these e.g. can contain a representation of GUI elements according to process understanding presented in section 4.2. Since this view does not only provide a hierarchical ordering of process steps, it supports the modeling of different levels of abstraction, granularity and topics. With this, it makes GUI elements decomposable to an arbitrary level of granularity, so that GUI elements can be considered as a group or as various individual compartments.
Analogous to the *Process Overview*, the *Activity Overview* provides the taxonomic view on activities. It therefore enables the knowledge acquisition on various levels of abstraction, granularity and topics.

Knowledge Acquisition. The *Activity View* specifies, which knowledge is provided by a certain knowledge carrier. Here, data and information streams are characterized, that mainly can be found at IT systems. Within the context of a certain learning task, they serve as *explicit knowledge object*. Implicit and non-transparent knowledge, that is mostly provided by humans and more recently by AI-based systems and CPS, is modeled by *tacit knowledge objects*. Since each *Activity View* characterizes the knowledge perspective of a certain process step or task, these are related by an 1-m relationship. This supports the presentation of clean models.

Knowledge Refinement. Acquired knowledge is characterized by the *Knowledge Overview*, so that a domain-specific hierarchy can be established. This view is not just hierarchical, since it supports the modeling of different levels of abstraction, granularity and topics of knowledge objects. By this, various orderings can be established depending on the current issue considered. Here, the very bottom level, an atomic knowledge object refers to numbers, with which an ANN learning will be carried out. Because of the hierarchical arrangement of knowledge objects, the dealing with a learning task can be considered at different observation levels and this view provides the foundation for the set specification.

Set Specification. With the aid of the *Process View*, it is specified at which process step knowledge is acquired. As process steps are ordered by the *Process Overview*, this specification is realizable on various processual levels. With the aid of the *Activity View*, knowledge is acquired for any processual level. Analogous, the *Activity Overview* enables the knowledge description on arbitrary knowledge levels. Since each knowledge object is part of a taxonomic network (*Knowledge Overview*), either, simple training and test set designs can be based on an abstract level dealing with knowledge

objects technically representing a group of concrete knowledge objects, or more detailed set designs consider the concrete knowledge objects themselves. This is realized with the aid of the *Set View*, which can be used for the design of numerous trainings and test sets on behalf of any knowledge specified so far.

5 Demonstration of NTML and Dynamic GUI Adaptation

In accordance with the procedure model presented in section 3, the following presents the demonstration of artefacts designed in section 4. As a software prototype, Asseco's ERP system called ap+ was chosen, for which GUI elements were intended to flexibly adapt to the interests of its users. The AI-based learning task was intended to be modeled with the aid of the NTML and so demonstrates the dynamic GUI adaptation on behalf of AI. The final GUI prototype can be found in Fig. 3. First, the prototype will be exemplified, then the corresponding NTML modeling use will be presented.

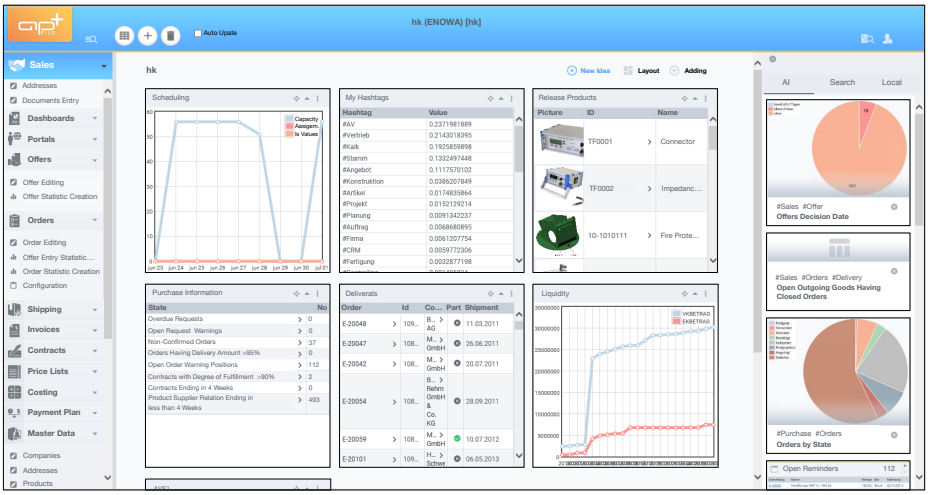

Fig. 3. AI-Based Software Interface Recommendations (Software Prototype ap+).

Here, one can see the dashboard region right within the software interface. It consists of a collection of individual dashboard items. On the right of the software interface, AI-based recommendations for further dashboard items are presented. Further, community-based and individually created, local software interface elements are provided by separate tabs. Only when the user accepts a certain software interface element, it is positioned within the dashboard region.

Faced with this kind of presentation, the design for the *Abstract Knowledge Use* becomes apparent. Each software interface element provided considers the user's data base although it was designed on behalf of someone else's data base. So, the thumbnail window will meet the expectations, when the user has accepted an element, although it was designed within somebody else's context.

As this software interface suits for both, knowledge origin and destination, the following shows how knowledge is accessed through the use of the software prototype and how knowledge is used in order to improve the software use by recommendations.

The *Process Overview* considers GUI elements as processual elements following the life cycle defined in Fig. 1. Beside furthers, such as menu elements, shortcut elements, etc., Fig. 4 presents elements of the dashboard region only (Fig. 1), which consists of a collection of dashboard items from *1* to *n*.

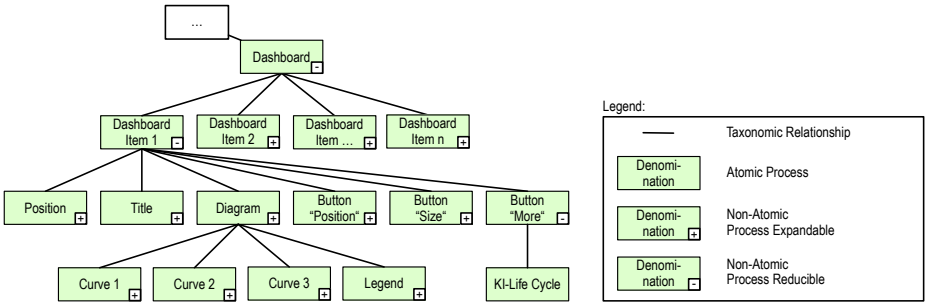

Fig. 4. Taxonomic Overview of GUI Process Instances (Process Overview).

In the example presented, *Dashboard Item 1* is characterized in detail. Its position refers to the placement within the dashboard region, such as the *ordering id* or *x axis* and *y axis*. A *title* can be placed at the top of the dashboard container, that refers in the case of *Dashboard Item 1* to "Scheduling" of Fig. 3. The diagram element consists of three *curves* and a corresponding *legend*. Next to the title, one can see three *buttons* providing a predefined functionality. The possibility to expand non-atomic process steps is indicated by a "+". The corresponding possibility to minimize expanded process steps is indicated by a "-". At the very bottom level, e.g. for the "Button 'More'", one can see the reference to the Life-Cycle (section 4.2).

Considering the *Activity View*, the knowledge acquisition is specified. Learning material can be found at the process called "User-Induced Acquisition" and "Exit of Process Instance" (Fig. 1) and is characterized in Fig. 5.

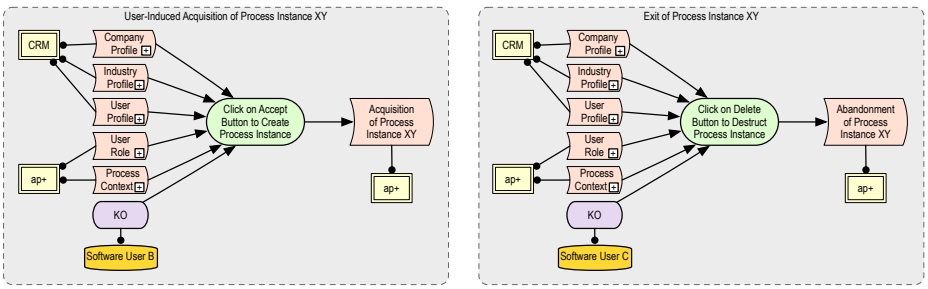

Fig. 5. Knowledge Acquisition for Visual Process Instances (Activity View).

Learning material refers to technical parameters, such as the *company profile*, the *industry profile* and *user profile*, are provided by the *CRM* system. The user's *role* and *process context* is provided by the ERP system.

Here, a recommendation for a certain GUI instance is identified by the acceptance of its process instance at the process step "User-Induced Acquisition". Alternatively, a recommendation could have been based on its use for a certain period. A recommendation against the corresponding GUI instance is identified by the destruction of its process instance at the process step "Exit of Process Instance". Alternatively, a recommendation could have been on the base of a non-acceptance, e.g. when GUI instance are suggested one by one and a next button is clicked.

On behalf with this learning material, training and test sets can be established comfortably. They so specify the underlying learning tasks efficiently. Fig. 6 exemplarily shows how knowledge objects are arranged in order to specify the test set for the learning tasks called "Offering of Process Instance XY".

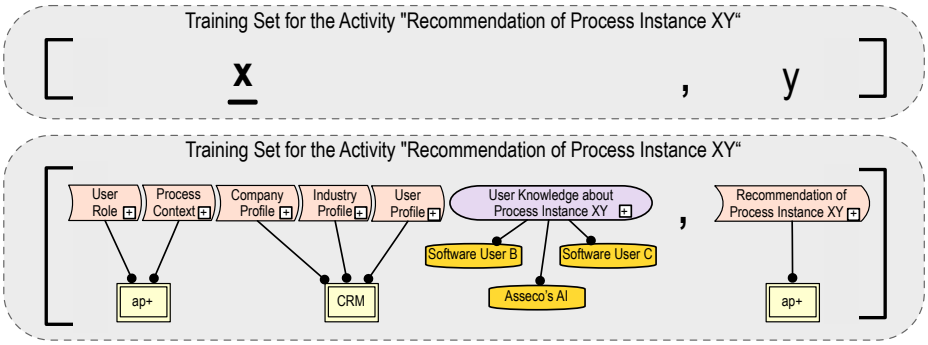

Fig. 6. Training Set Definition (Set View).

Since knowledge objects are characterized incl. their taxonomic relationship by the *Knowledge Overview*, a set definition be concretized by the expansion and reduction in the *Set View* as desired. Fig. 7 presents the corresponding visualization.

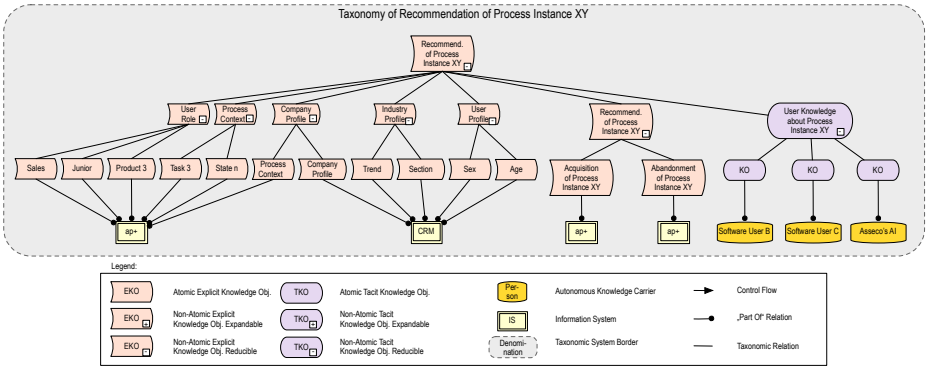

Fig. 7. Knowledge Taxonomy for Visual Process Instances (Knowledge Overview).

Here again, the possibility to expand non-atomic knowledge objects is indicated by a "+". The corresponding possibility to minimize expanded knowledge objects is indicated by a "-".

Having defined the learning tasks by the knowledge acquisition for visual process instances, the knowledge use can be specified at the process step called "Offering of Process Instance" (Fig. 1), which is characterized by Fig. 8.

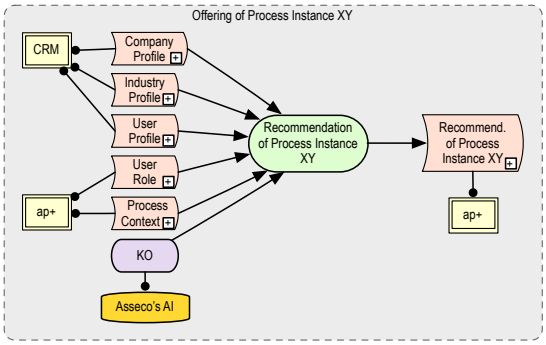

Fig. 8. Knowledge Use for Visual Process Instances Recommendation (Activity View).

Here, it is specified how recommendations of the AI-based system on the base of an ANN are used within the software prototype called ap+. Since any GUI element is considered as an individual process instance, which is positioned thoughtfully within its taxonomic hierarchy (Fig. 4), each accepted dashboard element can be refined by the process step "individual modification". Similar to the "user-induced acquisition" process step, here, each kind of activity generates knowledge. E.g. one could reposition a dashboard item or one could select the configuration parameter of a *permanent presence* of a dashboard item.

6 Evaluation

Being faced with requirements collected in section 4.1, the following evaluates inhowfar artefacts designed have been satisfied within the demonstration of section 5. In total, all requirements formulated have been satisfied with artefacts demonstrated.

NTML. The requirements of the NTML are satisfied as follows. Req. 1 is satisfied by Fig. 2. The organization's business processes as well as its current knowledge base is satisfied by the *Process View* (Req. 2) and the *Activity Overview* (Req. 3). The knowledge generation and use is issued with the *Activity View* (Req. 4, 6) and exemplified in Fig. 5 and Fig. 8. The learning tasks are explicable by the *Set View* designed (Req. 5) and exemplified with Fig. 6.

GUI Adaptation. The requirements of the GUI adaptation are satisfied as follows. The user's process context (Req. 7) and profiles (Req. 8) are considered for the generation (Fig. 5), the set definition (Fig. 6) and use within the process (Fig. 8). Direct interactions of the user with the software are separated by individual processes within the Process View (Fig. 1) and the specific activation as activity within the *Activity View* (Fig. 5 and Fig. 8). Further, recommendation of the vendor (Req. 10) and the community (Req. 11) are issued in Fig. 1 and exemplified as prototype in Fig. 3. The specific activity for an creation or destruction (Fig.5) and AI-based recommendation (Fig. 8) clarifies how a self-determined and self-responsible software use in cooperation with AI can be realized (Req. 12). The transfer of knowledge is considered (Req. 13) by first, the provision of GUI elements by process design (Fig. 1) with the aid of the *Abstract Knowledge Use Design* and by second, the acquisition and use of knowledge about the use of GUI elements, which corresponds to the ANN learning tasks exemplified.

Design Decisions. Decisions about the design have been considered as follows. Dec. 1 has been considered by the interpretation of GUI elements as process elements (Fig. 4). Their appearance and disappearance (Dec. 2) have been considered by the life cycle (Fig. 1). How knowledge is derived (Dec. 3-5) becomes apparent in Fig. 5.

All in all, requirements and design decisions have been considered jointly by artefacts designed and demonstrated with the aid of the software prototype ap+.

7 Conclusion

Summary. The first research question ("*How can AI-based learning tasks be specified with the aid of a modeling language?*") can be answered with the provision of a modeling language, which is adequate for the specification of learning tasks for ANN e.g. As the precise AI-algorithms are not concretized by the NTML, any other form of AI algorithm can be associated with this learning task, such as evolutionary algorithms, fuzzy systems, Bayesian networks or probabilistic methods [28]. Since the NMDL enables the graphical modeling, designs can be constructed cooperatively with the aid of modeling tools. Because of the consideration of business process models and the organization's current knowledge bases, a modeling of AI-based learning tasks is adequate. As one example design was presented, it has been demonstrated in the context of GUI element recommendation on the basis of supervised ANN algorithms. Please remark here, that the specific background of that ERP system is not necessary for the method to work, but it demonstrates the functioning of the approach presented in a real-world application context.

The second research question ("*How can dynamic software interfaces be optimized by AI-based GUI adaptations?*") can be answered on a level of general overview of dependencies by the provision of an understanding of GUI elements as process. This includes their atomic process characterization with the aid of a life span. Further, since AI-based recommendations go along with the knowledge transfers from source users to the vendor as intermediary and from the vendor to target users, the *Abstract*

Knowledge Use Design enables the user-independent possibility to externalize knowledge. This has been demonstrated by an example.

The third research question ("*How can AI-based recommendations be considered without depriving the responsibility of software users and vendors?*") can be answered with the provision of an adaptive life cycle design for GUI elements. Since this considers the user as final decision instance for the overtaking, creation, modification or destruction of objects of decision, that refer in this example to GUI elements, the responsibility of the user is not deprived. The user's freedom is only limited, if the vendor provides security relevant, critical GUI changes.

Jointly, concepts provided draw an artefact, that enables the AI-based, dynamic GUI adaptation. With this, in the long run, a surface is constructed by the interplay of software users, vendor, community and AI systems, that perfectly fits the current needs of a user.

Limitations. As concepts presented have been demonstrated by a software prototype, the focus of this contribution has laid on the proof of concept. Neither the NTML nor the dynamic GUI adaptation have been evaluated on a quantitative empirical base, such as customer feedbacks about the application performance increase or their acceptance about GUI adaptation by customers in general. This restricts the validity of evaluation results.

Outlook. Although the functioning of the NTML was demonstrated by a learning task example, its usage becomes particular effective when the NTML is integrated in a modeling tool, such as demonstrated by the *CoNM* [3]. Since the NTML refers to a subset of the *CoNM*'s *NMDL* standard, the example presented has been realized on behalf of the CoNM. Here, the training has been concretized on the basis of recurrent ANN structures. Outlined research will focus on the efficient learning task model creation by surveying cooperative model constructors.

Further research will also extend the example collection showing more versatile learning tasks and process designs than the ones presented. This supports the applicability to many learning contexts and research domains.

Future NTML examples shall contain temporal dynamics, learning modes (supervised, unsupervised and reinforced learning) [22], codification schemes [23] and learning problems (clustering, classification and regression) [24]. Here, research will need to discuss if the NTML shall provide further parameters, so that it is (1) more concrete and can specify the use of different concrete neuronal mechanisms similar to the original NMDL, and it is (2) further abstracted and can specify the use of any sort of AI mechanism, such as evolutionary algorithms, fuzzy systems, Bayesian networks or probabilistic methods. Please note, either algorithm-specific NTML extensions or the composition of modeling languages can be identified as design alternatives. Both akterbatuves support the creation of clear models, which are created on the basis of modeling languages with compact syntax. Instead, one joint Machine Learning Modeling Language (MLML) can be imagined, whose syntax is not compact. However, it would be very powerful.

Further research shall focus on the creation of *conformity checks*, that will analyze if training task designs are appropriate. An example can be found in Fig. 7: It is essential

that the knowledge objects called “User-Induced Acquisition” and “Exit” both provide categorial data, so that these are allowed to be grouped to the knowledge object called “Recommendation of Process Instance XY”. Otherwise, the learning task cannot be carried out properly.

References

1. Grum, M., Bender, B., Alfa, A.S., Gronau, N.: A decision maxim for efficient task realization within analytical network infrastructures. Decis. Support Syst. (2018). https://doi.org/10.1016/j.dss.2018.06.005.
2. Al-Roomi, M., Al-Ebrahim, S., Buqrais, S., Ahmad, I.: Cloud computing pricing models: a survey. Int. J. Grid Distrib. Comput. 6, 93–106 (2013).
3. Grum, M.: Concept of Neuronal Modeling (CoNM). , https://github.com/MarcusGrum/CoNM.
4. Peffers, K., Tuunanen, T., Gengler, C.E., Rossi, M., Hui, W., Virtanen, V., Bragge, J.: The Design Science Research Process: A Model for Producing and Presenting Information Systems Reseach. 1st Int. Conf. Des. Sci. Inf. Syst. Technol. DESRIST. 24, 83–106 (2006).
5. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. Manag. Inf. Syst. 24, 45–78 (2007).
6. The Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. ArXiv E-Prints. (2016).
7. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R.B., Guadarrama, S., Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding. CoRR. abs/1408.5093, (2014).
8. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. CoRR. abs/1912.01703, (2019).
9. Wongsuphasawat, K., Smilkov, D., Wexler, J., Wilson, J., Mané, D., Fritz, D., Krishnan, D., Viégas, F.B., Wattenberg, M.: Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow. IEEE Trans. Vis. Comput. Graph. 24, 1–12 (2018).
10. Cannon, R.C., Gleeson, P., Crook, S., Ganapathy, G., Marin, B., Piasini, E., Silver, R.A.: LEMS: a language for expressing complex biological models in concise and hierarchical form and its use in underpinning NeuroML 2. Front. Neuroinformatics. 8, 79 (2014). https://doi.org/10.3389/fninf.2014.00079.
11. Gleeson, P., Crook, S., Cannon, R.C., Hines, M.L., Billings, G.O., Farinella, M., Morse, T.M., Davison, A.P., Ray, S., Bhalla, U.S., Barnes, S.R., Dimitrova, Y.D., Silver, R.A.: NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail. PLOS Comput. Biol. 6, 1–19 (2010). https://doi.org/10.1371/journal.pcbi.1000815.
12. van der Aalst, WMP.: Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. In: Desel, J., Reisig, W., and Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets: Advances in Petri Nets. pp. 1–65.

Springer Berlin Heidelberg, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27755-2_1.
13. List, B., Korherr, B.: An Evaluation of Conceptual Business Process Modelling Languages. In: Proceedings of the 2006 ACM Symposium on Applied Computing. pp. 1532–1539. ACM, New York, NY, USA (2006). https://doi.org/10.1145/1141277.1141633.
14. Sultanow, E., Zhou, X., Gronau, N., Cox, S.: Modeling of Processes, Systems and Knowledge: a Multi-Dimensional Comparison of 13 Chosen Methods. Int. Rev. Comput. Softw. IRECOS. 3309–3319 (2012).
15. Grum, M., Gronau, N.: Process Modeling within the Augmented Reality - The Bidirectional Interplay of Two Worlds. In: Proceedings of the Eighth BMSD. pp. 206–214 (2018).
16. van der Aalst, WMP: Formalization and verification of event-driven process chains. Inf. Softw. Technol. 41, 639–650 (1999). https://doi.org/10.1016/S0950-5849(99)00016-6.
17. Rosing, M. von, White, S., Cummins, F., Man, H. de: Business Process Model and Notation (BPMN). In: Rosing, M. von, Scheer, A.-W., and Scheel, H. von (eds.) The Complete Business Process Handbook. pp. 433–457. Morgan Kaufmann, Boston (2015). https://doi.org/10.1016/B978-0-12-799959-3.00021-5.
18. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley (2005).
19. Karagiannis, D., Telesko, R.: The EU-Project PROMOTE: A Process-oriented Approach for Knowledge Management. In: Proc. of the Third Int. Conf. of Practical Aspects of Knowledge Management. (2000. pp. 9–18 (2000).
20. Heisig, P.: The GPO-WM® Method for the Integration of Knowledge Management into Business Processes. In: International Conference on Knowledge Management. pp. 331–337. , Graz, Austria (2006).
21. Gronau, N.: Modeling and Analyzing knowledge intensive business processes with KMDL. Comprehensive insights into theory and practice. Gito (2012).
22. Deru, M., Ndiaye, A.: Deep Learning mit TensorFlow, Keras und TensorFlow.js: Einstieg, Konzepte und KI-Projekte mit Python, JavaScript und HTML5. Rheinwerk Verlag GmbH (2019).
23. Kruse, R., Borgelt, C., Braune, C., Klawonn, F., Moewes, C., Steinbrecher, M.: Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze. Springer Fachmedien Wiesbaden (2015).
24. Nauck, D., Borgelt, C., Klawonn, F., Kruse, R.: Neuro-Fuzzy-Systeme: Von den Grundlagen künstlicher Neuronaler Netze zur Kopplung mit Fuzzy-Systemen. Vieweg+Teubner Verlag (2013).
25. Cresswell, J.W.: Die Entwicklung der Mixed-Methods-Forschung. In: Kuckartz, U. (ed.) Mixed Methods: Methodologie, Forschungsdesigns und Analyseverfahren. pp. 13–26. Springer Fachmedien Wiesbaden (2014).
26. Kuckartz, U.: Mixed Methods: Methodologie, Forschungsdesigns und Analyseverfahren. Springer Fachmedien Wiesbaden (2014).
27. Masaaki, I.: Kaizen: The key to Japan’s competitive success. N. Y. Ltd McGraw-Hill. (1986).
28. Konar, A.: Computational Intelligence: Principles, Techniques and Applications. Springer Berlin Heidelberg (2006).