Refinement of Adaptivity by Reflection

Klaus P. Jantke*, Hans-Rainer Beick**, Yuriy Brovko* and Sebastian Drefahl*

Abstract—Adaptivity is a system’s ability to respond flexibly to dynamically changing needs. Adaptivity to human needs, wishes and desires—even to those that might be unconsciously present—is a particularly ambitious task. A digital system which is expected to behave adaptively has to learn about the needs and desires to which it shall adapt. Advanced adaptivity requires learning on the system’s side. Under realistic application conditions, information about a human user available to a computerized system usually is highly incomplete. Therefore, the system’s learning process is unavoidably error-prone and the knowledge on which the system’s adaptive behavior has to rely is hypothetical by nature. Adaptive system behavior is improved by the system’s ability to reflect on the reliability of its current hypothetical knowledge.

I. INTRODUCTION & MOTIVATION

Expanding on [23], the present contribution develops some original approach to the refinement of system adaptivity by means of reflection—the digital system’s ability to ponder its own power and limitations.

The approach covers theory development and conceptual design, generic tool implementations, and some prototypical applications.

The work is motivated by the following chain of thoughts.

- The contemporary development of information and communication technologies results in the technology pervading increasingly more areas of human life, thus reaching an increasing percentage of naïve users unlikely to become IT specialists ever.
- An efficient and effective use of information and communication technologies requires an increasing ability of the digital systems to adapt autonomously to the needs and desires of their human users.
- Any adaptivity of a digital system needs to be built upon knowledge about what to adapt to.
- Adaptive systems need to learn about the possibly changing needs and desires of their possibly varying human users based on the interaction with them.
- The information extractable from human-computer interaction is usually incomplete and often uncertain. Therefore, the system’s learning is usually error-prone.
- Because a system’s adaptivity is unavoidably based on incomplete and uncertain information, adaptivity must be realized very cautiously.

Reflection on the reliability and on the appropriateness of its knowledge is key to a system’s cautious adaptation.

II. ADAPTIVITY IN LEARNING

There is no doubt at all among educators that learning and teaching, resp., is more successful when the particular needs of the learners are taken into account [15], [21], [39].

What this might mean in more practical details is discussed controversially (see, e.g., [11] and the debate between [33], [52] and [37]). Contemporary educational technology is still far from dealing with issues as focused in the latter four references.

Adaptivity dates back to work as early as [12], [10], and [45]. It reached practical computer science first in the area of natural language processing [43] almost exactly 35 years ago. Nowadays, adaptivity seems to be a standard requirement in technology enhanced learning, at least, as [3], [4], [5], [13], [19], [29], [34], [38], [50], [51], and others nicely demonstrate. Disciplines like software ergonomics are affected as well [42]. Nowadays, adaptivity goes far beyond the limits of technology enhanced learning being relevant to the majority of customer-oriented system developments.

However voluminous the academic literature, conferences, case studies, and evaluations, the adaptivity of an educational system still mostly reduces to varying navigation, content, and content presentation [50], [51]. This, essentially, requires suitably flexible building blocks, so-called learning objects [41], to built upon presentations adaptive to every learner [40].

But what to adapt to? What might a digital system possibly know about its human user?

Carl Gustav Jung proposed “psychological types” [30]. The Myers Briggs Type Indicator (MBTI) [2] reflects this early approach and is still rather successful in business practice. It is complemented by a large number of different approaches [4], [32], [35], [36].

Whatever approach one assumes, a digital system to adapt effectively to human users with different needs and desires has to learn about them. This is the crux.

As we will see in the following chapter very explicitly, one can hardly guarantee that a system’s belief about its human users is always sufficiently correct. Adaptivity based on misunderstanding the addressee’s needs and desires may go terribly awry.

Adaptivity may be improved if a system before adaptation ponders its knowledge base, a behavior we call reflection.
III. ADAPTIVITY AS LEARNING

This section is a condensed version of the first author’s book chapter [27], but goes clearly beyond the limits of this chapter which addresses teachers more narrowly discussing the assistance for technology enhanced learning in school\(^1\). The introduced terminology serves a concise treatment in the following sections IV and VI.

For simplicity, we assume an interaction scenario stripped down to the essentials. There is some system of interest, some particular human user, and–within the current context of application such as, e.g., some learning task or some information gathering–some target characteristics of the user. Under these conditions, we investigate the system’s knowledge acquisition and processing toward adaptively serving the user.

The subsequent sections V, VI, and VII deal with different aspects of making adaptivity come true.

Let us briefly put the cart before the horse. If system adaptivity is assumed to work ultimately, there must be some ultimate goal to be learned which describes the human user within the given context sufficiently well. If such a target would not exist, there were no way to adapt to it.

We denote the ultimate target by \( T \). Readers may imagine some particular user profile within the MBTI formalism [2], within the David Kolb Learning Style Inventory (LSI) [36], according to Felder and Silverman [14], or anything like that.

For the purpose of adaptivity, it is the digital system’s task to learn \( T \) from data collected during human-machine interaction.

Initially, the system has some user profile denoted by \( T_0 \). This may be either empty or contain certain information based on the user’s login data and, thus, on former sessions.

Based on data collected throughout human-system interaction, the system updates the user profile from time to time. In this way, there emerges some sequence of hypothetical user profiles \( T_1, T_2, T_3, T_4, \ldots \).

Due to the unavoidable incompleteness of interaction data, the scenario sketched above coincides with learning models of inductive inference dating back to Solomonoff’s early work [48],[49] and, in particular, to Gold’s seminal paper [16]. Nowadays, [20] is the standard reference.

The basic learning scenario is illustrated by means of fig. 1. The scenario may be refined in manifold ways including some varying specifications of concepts of convergence [28].

At any point \( n \) in time, the current user profile \( T_n \) is used to determine the way in which the system attempts to adapt to its user according to the beliefs formally represented by \( T_n \).

In a large number of application contexts, it is an urgent requirement to have declaratively represented rules of adaptive behavior. In approaches to technology enhanced learning that are learner-centered, for instance, it is a must to allow the learner to check the current learner profile and to understand the digital system’s behavior. Furthermore, declarative representations of the mechanisms of adaptivity are a necessary prerequisite of debating variants of media didactics.

To keep it short, adaptivity is specified in a rule based form (see section V) with preconditions allowing for adaptive choices.

Assume \( \alpha \) to be some adaptation rule, e.g., deciding about the appearance of some content presentation. \( \alpha \) seen as a rule has the form \( c \rightarrow d \) meaning that under some constraint \( c \) there is made some decision \( d \) about the system’s behavior.

When at some point in time the system needs to decide about how to serve the user’s needs best, it has to check its rules for adaptation. Which one does apply?

In formal terms, we are facing a problem of automated deduction (see [46] and [47] for a birds-eye perspective and [1] for a particularly reader-friendly introduction). For every relevant rule of adaptation \( \alpha \), the system needs to find out whether or not the precondition \( c \) holds under the beliefs summarized in \( T_n \). In logical terminology, this is the question for entailment written as \( T_n \models c \).

In case \( T_n \models c \) holds, the adaptation rule \( \alpha \) may be applied. Otherwise, written as \( T_n \not\models c \), the system needs to look for another rule or for some default.

In practice, drawing conclusions of the form \( T_n \models c \) is frequently very simple. If constraints just compare values, e.g., such as an achieved score or a percentage of solved exercises, entailment mostly means looking into the database for the bookkeeping of the corresponding parameters. There is no need here to go into further details.

The logical terminology introduced leads to a particular perspective at learning. For any target profile \( T \), it is very unlikely that a system’s learning process ultimately leads to some sequence of hypotheses \( T_1, T_2, T_3, T_4, \ldots \) where at some time point \( n \) the target \( T \) occurs syntactically correct, i.e., \( T_n = T \).

For a sufficiently correct adaptive system behavior, it is not necessary that hypotheses represent the target of learning syntactically correct. Instead, sooner or later some hypothesis should be semantically correct, i.e., it should allow for the same conclusions as \( T \) does. Then, \( T \) has been learned.

\(^1\)The publication [27] is a chapter of the 2013 almanac for teachers of the German federal state of Thuringia. For this reason, it is written in German.
IV. LEARNING & REFLECTION

One of the crucial problems of learning from incomplete information consists in the difficulty to find out whether or not some currently adopted hypothesis is already correct or, at least, sufficiently correct for some purpose such as drawing conclusions on its basis. The question is known to be generally undecidable [20], [28].

What does that mean in practice? What does that mean to computerized systems aiming at an adaptive behavior by means of learning about the human users?

Let us assume any digital system aiming at adaptivity which is inductively learning about some user’s needs and desires by constructing a sequence of user profiles $T_0$, $T_1$, $T_2$, $T_3$, ... as introduced in section III before.

When at time point $n$ opportunity is knocking for some adaptive system behavior by applying some rule $\alpha = c \rightarrow d$, there arises the crucial question whether $T_n \models c$ holds or not. If $T_n \models c$, why not adapting as determined by $d$ ...?

The answer depends on the reliability of the intermediately hypothesized user model $T_n$. In case the system is convinced, so to speak, that its understanding of the human user is justified, it should adapt accordingly. In contrast, if the system has its doubts about $T_n$, to use an anthropomorphic circumscription again, it should better stick with behavior based on $T_{n-1}$.

This requires an expression for an adaptive computerized system’s “trust in its own thoughts”.

Apparently, the most simple approach is to label hypothesized user profiles as either “trustworthy” or “untrustworthy”. This elementary approach to reflection about learning systems’ hypotheses has been introduced by the first author in [22]. Subsequently, Gunter Grieser has investigated the issue of reflection in inductive learning in much detail and came up with the largest number of results world-wide [18].

The present section IV will only focus on the essentials. Formalisms are introduced in as much detail and depth, only as necessary for subsequent elaborations. From the large variety of possible formalizations, the authors confine themselves to the most simple one. For illustration, we refrain from studying degrees of trustworthiness as in fuzzy logics, e.g., and simply stick with binary values of classical logics opposing trustworthiness to untrustworthiness. A few selected results are discussed for the purpose of explicating potentials and limitations.

In accordance with the original sources of reflection in inductive learning [22], [18], any computerized system that is capable of learning from incomplete information, for reflecting its hypotheses, it needs to be able to mark every hypothesis either by 1 (meaning trustworthy) or by 0 (untrustworthy). Moreover, such a labeling needs to be meaningful with respect to the overall learning process. When learning does ultimately succeed, the reflection should “say 1 in the end”. Otherwise, if learning fails, it should say no. This needs some explanation.

The crux is, as briefly mentioned above, that there is no end that can be recognized.

The general algorithmic undecidability of knowing when an inductive learning process arrives at its ultimate goal—recall that in the domain under consideration, arriving at the goal means that the system understands its human user sufficiently well—leads to particular conceptualizations as follows.

Any learning system labeling its hypotheses by 0 or 1, respectively, is said to learn inductively with reflection, exactly if the following two conditions hold. On any sequence of hypotheses that converges correctly to a learning target, the corresponding sequence of labels converges to 1. On any sequence of hypotheses that does not converge successfully, the sequence of labels converges to 0. Note that these conditions may by paraphrased in a quite different way, because there are only the two alternative reflection values 0 and 1 available. When a learning system succeeds in learning, the values of reflection become constantly equal to 1 past some time point. In contrast, when the system fails in learning, the values of reflection become constantly equal to 0 past some time point.

For completeness, we mention some refinement due to [18]. Grieser has introduced so-called optimistic, pessimistic, and exact reflection. For illustration, some reflection is said to be optimistic, exactly if its labels of untrustworthiness are always definitely justified.

The figures 2 and 3 survey more than two dozens of results about possibilities of making learning algorithms reflective. Terms such as CONS, CONS$^{arb}$, T-CONS, T-CONS$^{arb}$ have been studied by the two first authors of the present paper in [28]. Every term denotes a class of solvable problems of learning from incomplete information. Extensions such as, for instance, -Refl and -$arb$Refl express the additional requirement of providing some reflection or some optimistic reflection, resp.

![Fig. 2. Survey of results on reflection in inductive learning from [18], p. 64](image)

Fig. 2 states that there are learning problems which can be equipped with any form of reflection without any loss of learnability. In contrast, fig. 3 shows that in certain conditions almost every requirement of reflection is problematic, because it reduces the amount of solvable learning problems. Interested readers may consult [18] for explanations and even for proofs.

Intuitively speaking, it is frequently possible to enrich learning algorithms including those constructing user profiles for the sake of adaptivity by the additional ability to reflect about the trustworthiness of hypotheses.

But it is a rather involved issue whether one can get the reflection for free (see fig. 2) or has to accept some reduction of learning power (see fig. 3).
V. TECHNOLOGIES FOR ADAPTIVITY

In [24], some program has been sketched for generic technology development aiming at tool support for the design and implementation of adaptive systems. The present section surveys the essentials of the work.

Before going into the details revealing what functionality we provide in the GERA authoring tools, the high level architecture will be described together with the challenges from the development point of view.

During the early design phase, decisions had to be made concerning what type of software should be developed, what programming language will be used and what part of the system should provide which functionalities. Fig. 4 shows the system’s software architecture supporting the unified, generic, flexible and extendable approach necessary to allow for a multitude of target applications.

A web based application suite comprised of three main parts is the result. We chose to develop in Java, because it is a multi-platform solution and widely used.

The heart of the GERA authoring tools, the user kernel (Nutzerkern), is the central component for storing, processing and distributing data. It can be implemented as part of the target application and on a server in the intranet or internet.

The communication layer is responsible for connecting the target applications to the system. See section VII of this paper to learn more about adGERA, one of the target applications.

What is called presentation layer reflects the applications to work with in the end. The presentation layer consists of three main editors, each focusing on one specific task necessary to implement adaptivity: (i) user modeling, (ii) system learning and (iii) system behavior.

(i) In order to adapt to users, a system needs a conceptual user model to be filled with information based on a user’s interactions over time. Our User Model Editor allows to create, change and delete user models. They can be searched and exported into XML files.

(ii) Our System Learning Editor provides the necessary tools to analyze and interpret a user’s interactions and to fill a user’s profile by writing the findings into it. A requirement for this process to work is to get sensor data from the target application.

(iii) The final step toward a complete adaptive system is to be able to define adaptive reactions, to specify the behavior of a program after a certain amount of knowledge about a user has been gained. Here, our System Behavior Editor comes into play and allows to define appropriate measures, the adaptivity rules.

![GERA User Model Editor](image)

Fig. 5. The GERA user model editor providing a list of user models

To build user models out of components means to provide predefined versions for rules, patterns, variables and the like. Edit, copy, paste, search and more advanced functions such as, e.g., defining relations conveniently allow to create proper user models. System information is available in a console at any time providing vital feedback to all workflow steps.

The next chapters aim at briefly explaining this adaptivity workflow inspired by a real use case implemented in adGERA (see section VII). Such planning software should differentiate between users who prefer different ways of data presentations such as, e.g., in tabular or graphical form.

The user model needs to be able to store the number of times a user clicks on the buttons for a tabular or a graphical view, respectively. To achieve this, all counter and parameter components must be added in the User Model Editor.

When a user starts to interact with the target application, at first the personal user profile will be generated based on the user model. In some cases, it is necessary to store the user profile for some comparison in the future. This can be achieved by defining the circumstances when the storage should be triggered.

After some time of recording, interaction data recorded in the user profile are increasingly likely to characterize the user’s preferences. The system is learning (section III). Some threshold may be set to determine when to adapt accordingly.

Once such conclusion about the specific users’ preference is reached, the planning software is able to take it into account by adaptivity rules written in the System Behavior Editor by means of JavaScript. They will be processed and the newly learned preferred type of data presentation will be transmitted to the planning software. The outcome would be to show the data to the user by default in the preferred way.

Right now, we are working on a variety of improvements, for instance, the integration of a portal framework and optimizing usability. In future versions, we expect to add storyboarding in order to provide additional workflow capabilities.
VI. REFINEMENT OF ADAPTIVITY BY REFLECTION CONCEPTUALLY

The previous section has been very briefly surveying the authors’ generic tool development toward system adaptivity in practice. This section together with the following one is intended to illuminate the refinement of adaptivity by reflection.

It is obvious both from practically oriented thoughts as well as from theoretical viewpoints [18] that there are infinitely many ways of specifying reflection, even on a conceptual level. This section does not deal with all of them.

First of all, there is a large variety of approaches which do not take any deeper reasoning into account. For simple cases, they might do. Think, for instance, of just looking at changes of hypotheses in the sequence $T_0, T_1, T_2, T_3, T_4, T_5, T_6, \ldots$. One may understand it as an evidence of uncertainty if some hypothesis has been newly generated different from the one before, i.e., if $T_n \neq T_{n-1}$ holds at the point of time $n$. The most trivial idea of reflection is to label hypotheses such that $\lambda(T_0) = 0$ and $\lambda(T_n) = 1$ only if $T_n = T_{n-1}$ (for any $n > 0$). ($\lambda$ is shorthand for indicating the label of trustworthiness.)

Several variations of this really trivial approach are obvious. For example, one may require hypotheses to stay stable for a longer time before they are considered trustworthy, formalized by expressions such as $\lambda^k(T_n) = 1$ if and only if the chain of equalities $T_n = T_{n-1} = T_{n-2} = T_{n-3} = \ldots = T_{n-k}$ is valid. However, the authors are not following those ideas any longer.

Far beyond the limits of syntactic comparison, promising reflection needs to consider what hypotheses are expressing.

Pondering the question for trustworthiness of hypotheses, one needs to clarify relevant relations between the hypotheses. The key approach derives from the usage of user profiles in the process of reasoning. As introduced above, user profiles are deployed for checking the validity of constraints of adaptive behavior formally expressed in the form $T_n \models c$. Consequently, one may compare hypotheses with respect to the amount of constraints which may be derived.

We define the consequence operator $C$ assigning to every hypothesis $T_n$ the amount of its logical consequences $C(T_n)$.

Formally, $C(T_n) = \{ \varphi \mid T_n \models \varphi \}$, where $\varphi$ is a variable for formulas. The family of all those sets forms a lower semi lattice [17]. This semi lattice is usually infinite.

If an adaptive system has been specified according to the methodology and, perhaps, using the authors’ tools sketched in section V, there is implicitly given the amount of all constraints $c$ occurring in any rules of adaptivity $\alpha$ of the form $c \rightarrow \alpha$. For any given adaptive system (possibly under development), we denote by $C$ the set of all those constraints $c$. $C$ is finite.

For the comparison of two different hypotheses $T_m$ and $T_n$, only $C(T_m) \cap C$ and $C(T_n) \cap C$ are relevant. In other words, for the comparison of human user profiles, it is relevant which of the constraints may be derived2. If for two different hypotheses these sets of constraints are identical, the hypotheses, at least from the viewpoint of adaptivity, are semantically equivalent.

There are several rather simple, but practically important consequences directly derived from the fact that the original semi lattice is abstracted to a finite semi lattice.

- Many (from a theoretical perspective, even infinitely many) syntactically different user profiles are semantically equivalent.
- Under the assumption that user profiles are growing by adding data from time to time, semantical changes of hypotheses are irreversible.
- All sequences of hypotheses converge semantically, i.e. after some point in time all user profiles imply exactly the same constraints.

The last point above is decisive, because it is providing some evidence for the present approach to ultimately succeed.

Let us now carry over the present insights to reflection. The authors’ semantic approach to reflection—on the conceptual level—allows for the introduction of a larger number of reflection concepts $\lambda$ as exemplified above. For the purpose of shorter formal expressions, we introduce $C^\lambda(T_n) = C(T_n) \cap C$.

$$\lambda(T_n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n > 0 \text{ and } C^\lambda(T_{n+1}) = \ldots = C^\lambda(T_{n+1/2}) \\ 0 & \text{otherwise} \end{cases}$$

Here, $[\ldots]$ means the integer part of the number in brackets.

Intuitively speaking, this type of reflection says that some hypothesis is considered trustworthy, exactly if for at least half of the learning history the most recent hypotheses have been semantically equivalent.

Like all other approaches to adaptivity refined by reflection, this approach has advantages and disadvantages. First of all, it does converge to 1 in case of learning. But the response rate of adaptivity is slowing down over time.

Using labellings $\lambda$ as introduced and discussed above, one may easily refine adaptivity by reflection. For determining a system’s adaptive response at time point $n$, choose rules $\alpha$ of adaptivity and check the validity of their constraints $c$.

Reasoning is based on the largest time point $m$ with $m \leq n$ such that $C^\lambda(T_m)$ holds.

[2] This points to some highly interesting area of investigation unfortunately beyond the limits of this submission. There is some feedback from the design of adaptivity to the design of user models, because the—conceptually earlier—design of adaptivity including the specification of constraints has some impact on classifying the—conceptually earlier—designed user profiles.

Fig. 6. The space of potential user profiles seen as a semi lattice (see [17])

But in practice, there are only finitely many constraints of interest—those introduced in rules of adaptivity (section V).
VII. REFINEMENT OF ADAPTIVITY BY REFLECTION PRACTICALLY

Within the project consortium, ADISY Consulting is using adaptivity refined by reflection. The present application case is ADISY’s tool for the optimized detailed process planning in manufacturing enterprises. The technology is promising. Thus, it shall be carried over to tools for business analytics, in the near future. This section demonstrates the enterprise’s view at the approach and its potentials developed in previous chapters. The focus is on the plan optimization application, exclusively.

Detailed planning of manufacturing processes establishes so-called scheduling problems³. A technical term in use is job shop scheduling.

Given any job shop scheduling problem, the computation of a detailed optimal plan to resolve the scheduling problem is algorithmically infeasible, in general. In mathematical terms, it is known to be NP-complete.

A manufacturing plan consists of elementary steps of manufacturing which are allocated to resources such as machinery. Within some detailed plan, those elementary activities are partially ordered and assigned to time intervals. There are constraints to be satisfied. In a correct plan, there shall be no conflicts of resource allocations as well as no violations of technologically necessary successions.

Due to the larger number of possible combinations which grow exponentially in dependence on the number of elementary steps to be taken into account, there is no feasible method for getting optimal plans. Even to get sufficiently good plans, only, turns out to be impossible, in practice.

In response, contemporary planning tools comprise a lot of different methods for constructing and for improving plans. Practical planning starts with the construction of some plan which, ideally, does not violate any constraint. Plan generation proceeds by plan modification and improvement according to target functions of mathematical optimization. A predefined mix of some measures defines the plan quality.

Planning is an interactive process. The human planner is exploring the huge space of potential solutions. A frequent act of interactive plan generation is trying out different measures, i.e., different target functions, and comparatively evaluating the intermediate results.

A careful analysis of contemporary planning practice in the manufacturing industries reveals that human planners rarely use the potentials of the planning tools. In the majority of small and medium size enterprises, there is a lack of knowledge and skills for exploitation of the available planning possibilities. Human planners prefer to choose predefined and accustomed ways of planning.

As a result, although a good detailed plan is a crucial basis for effective manufacturing and, thus, the human planner’s ultimate goal, really good plans are rarely found.

³There is no space to introduce the background of the application domain systematically and to refer to sufficiently comprehensive literature resources. This is a large field of research and development in its own right. It relies on mathematical optimization and on computational complexity which, in turn, is built upon the theory of effective computability aka the theory of algorithms. In particular, an understanding of the theory of NP-completeness is essential.

The only way out consists in a better assistance for the user and in better possibilities for doing experiments. Planning should become a bit like playing a digital game. The planner tries different possibilities, experiences more or less successful steps, can always go back, and has always an overview about the quality of the temporarily generated plans. The measures appear just like scored points in a game.

This is the basic scenario to proceed from planning tools to planning assistants. But assistants should be adaptive [31].

The high degree of interaction—according to the metaphor of seeing planning as playing—allows for the collection of larger amounts of data about the human planner and about the planner’s preferences when planning. Planner modeling reflects the way of working, preferences for actions, skills, and expertise.

ADISY Consulting adopts the present view at adaptivity. Inspired by the pheromone principle in ant colony optimization, there is a certain preference for some gradual oblivion in planner modeling. More recent planner decisions shall have a higher impact on the profile than earlier decisions. As a case study, let us consider the question for the planner’s choice between tabular or graphical visualizations of plan evaluations as illustrated by means of fig. 7.

![Fig. 7. The tabular and the graphical alternative displayed in parallel](image)

Planner modeling of the preference for tabular vs. graphical presentations depends on some factor \( f \) (some figure between 0 and 1). There are internally stored evidences \( e \) for the one decision and \( \bar{e} \) for its counterpart. When the planner makes some decision sound with \( e \), the evidence value is updated by calculating \( e := f \cdot e + 1 \). In case of the opposite decision, the value of \( \bar{e} \) is updated accordingly. The rules of adaptivity compare \( e \) and \( \bar{e} \) and choose a presentation according to the higher evidence value for the human planner’s preference.

To avoid a flickering behavior of adaptivity, reflection is adopted as well. There is some threshold \( t \) between 0 and 1. For any hypothetical planner profile \( T_n \) with the evidence values \( e \) and \( \bar{e} \), \( \lambda^{[1]}(T_n) = 1 \), if and only if \( |e - \bar{e}| > t \).

The big advantage of using adaptivity refined by reflection is a higher user acceptance of the offered system assistance. Changing presentations do not confuse the human planner, when they occur moderately delayed by reflection.

In future further elaborated applications, adaptivity refined by reflection is intended to contribute to the acquisition of knowledge about the domain, because planning is extremely context-sensitive and differs from branch to branch, from enterprise to enterprise. Planning strategies shall be learned by recording successful adaptation.
VIII. Summary, Conclusions & Outlook

A. Summary

Ideas of adaptivity in information and communication technology are known for about 35 years and are no novelty at all. But the crucial relationship between the ultimate goal of adaptive behavior, on the one hand, and the necessary prerequisite of learning about what to adapt to, on the other hand, has been rarely discussed in sufficient scientific depth. Consequently, the problem of incompleteness and uncertainty of an adaptive system’s basis of reasoning and behavior has not yet found much attention.

The authors’ answer to the problem is to refine adaptivity by reflection.

A systematic treatment does inevitably require to understand user modeling as learning (see section III above) and to identify the limits of learning from incomplete information what directly leads to concepts of reflection (see section IV).

The first author’s paper [23] in introducing the basic ideas and discussing the needs and potentials has set the stage for the present application-oriented work. As reported in the chapters V and VII above, there are tools available and applications in progress, respectively.

The authors’ present approach of “understanding humans and their needs and desires”, of “learning about other humans”, and of “worrying about ones hypotheses and pondering the suitability of the hypothesized knowledge for generating upon effective adaptive responses” might sound somehow ambitious. Indeed, for a human being to understand another one and to behave accordingly is an art.

This contribution may be seen as an attempt to transfer the art into a science, very much in the sense of Donald Knuth\(^4\) who discussed the question whether programming is more an art or a science. Knuth advocated that we should always strive hard to transform the art into a science. Even if we do not fully succeed, both art and science will draw some benefit.

B. Conclusions

More widespread modern information and communication technologies require an increasing ability to adapt to human users, instead of expecting human users to continuously adapt to technology. Adaptation of computerized systems necessarily requires those systems’ ability to learn about their users and, perhaps, about varying context conditions of their application. Learning based on unavoidably incomplete information results in hypothetical knowledge, only. Reasoning and behavior based on knowledge just hypothesized has to be treated very cautiously. All this bears abundant evidence for the need of reflecting about an adaptive system’s internal basis of behavior.

In the authors’ opinion, the present approach leads to a revised perspective at adaptive systems and how to design and implement them. Designing an adaptive system means to design a learning system, more precisely, to design a reflective learning assistant (see also [31] and the chapters therein).

To equip adaptive systems with reflective learning abilities is a great challenge which requires appropriate tool support.

\(^4\)TURING Award Lecture 1974

C. Outlook

The present submission is necessarily much too short to reflect all the authors’ recent work on adaptivity and reflection including the ongoing applications in rather different areas. Beyond the application case sketched in section VII above, there is another case study within the same project focusing adaptivity in some digital game on tablet PCs developed for children’s learning of some German language grammar issues. Other cases of developing and implementing adaptivity focus on learning about history or training for disaster management. Reporting this related work of the authors is beyond the limits of the present contribution.

Beyond the state of the art and, thus, more interestingly, there are several extensions and generalizations of the present approach. Four of them will be briefly introduced in the sequel.

a) Expressive Theories: Work in cognitive psychology such as on theories of mind [8] and on conceptual change [6], [7], [53], for instance, suggests to model human learners in ways quite different from what is current practice. Hypotheses according to those future approaches are much more expressive than user profiles, e.g., à la David Kolb, à la Myers Briggs, or à la Felder and Silverman.

User profiles may become theories that really tell something about the human user, about needs and desires in a declarative way [25], [26].

b) Types of Reflection: Although the work on reflective learning on which the present paper is based [22], [18] helps to go far beyond the current theory and practice of adaptivity, options for extensions and refinements are obvious.

At least, one should formalize, investigate in depth, and explore practically labels of hypotheses finer than just 0 and 1, somehow in the sense of fuzzy logics. Furthermore, one may prefer values of reflection that are not just numerical, but provide reasons for trust or doubts, respectively. This closely relates to more expressive theories (see above).

c) From Reflection to Empathy: According to Popper’s logic of scientific discovery [44], learning and theory formation can hardly come up with provably correct results. All one can possibly do is disproving incorrect hypotheses. This insight leads to specific approaches to learning from incomplete information [9].

Carried over to user modeling as learning, that means that changes of hypotheses represent moments when an adaptive system has understood, so to speak, that it treated its human user incorrectly before. In terms of this paper’s formalisms, think of some \(T_n\) and \(T_{n+1}\) with \(\lambda(T_n) = 1\) and, perhaps, \(\lambda(T_{n+1}) = 1\), but \(C^T(T_n) \neq C^T(T_{n+1})\). The computerized system needs to change the way it is treating its human user, but carefully and with some empathy.

IX. Acknowledgement

The work reported here has been supported within the joint project GERA, subproject E1E of IDMT and subproject IABS of ADISY, under the contracts 03WK27A and 03WK27B, respectively.
REFERENCES