Fourth International Conference on

User Modeling

Proceedings of the Conference

15-19 August 1994

Tara Hyannis Hotel

Hyannis, Massachusetts, USA

Published by The MITRE Corporation

© 1994, User Modeling Inc.

Order copies of proceedings from: Bradley A. Goodman, The MITRE Corporation, Artificial Intelligence Center, 202 Burlington Road, Bedford, MA 01730-1420, USA

Student model centered architecture for intelligent learning environments

Peter Brusilovsky
International Center for Scientific and Technical Information
Kuusinen str. 21b, Moscow 125252, Russia
plb@plb.icsti.su

Abstract

We discuss architectural problems of the student model centered approach to building intelligent learning environments (ILE). By this approach different components of ILE including tutoring, coaching, environment, and manual components use the central student model to adapt its behavior to the given student. The implementation of this approach is based upon ideas from the fields of intelligent tutoring systems, adaptive interfaces and intelligent help systems. We introduce a simple student model centered architecture for ILE which we have applied in several implemented systems, report some problems and limitations of our original simple architecture, and present an advanced student model centered open architecture for ILE.

Introduction

An intelligent learning environment is a relatively new kind of intelligent educational system which combines the features of traditional Intelligent Tutoring Systems (ITS) and learning environments. Traditional ITS are able to support and control student's learning on several levels but doesn't provide space for student-driven learning and knowledge acquisition. An intelligent learning environment (ILE) includes special component to support student-driven learning, the environment module. "The term *environment* is used to refer to that part of the system specifying or supporting the activities that the student does and the methods available to the student to do those activities." (Burton, 1988, p.109). Some recent ITS and ILE include also a special component (we call it as "manual") which provides an access to structured instructional material. The student can work with the manual via help requests or via special browsing tools exploring the instructional material on her own. An *integrated* ILE which includes the environment and the manual components in addition to regular tutoring component can support learning both procedural and declarative knowledge and provide both system-controlled and student-driven styles of learning.

Our research at the Moscow State University and International Center for Scientific and Technical Information (ICSTI), are centered around two problems of creating integrated ILE: the problem of adaptation and the problem of integration. As for *adaptation*, the problem is to make all the components of an integrated ILE adaptive. Most of ITS and tutoring components of ILE can adapt its work (tutoring) to the given student, however very few environment and manual component can do that. It was one of our goals to build adaptive environment and manual components of ILE. As for *integration*, our position is that an integrated system should be not just a sum but a real integration of its components. In particular, it requires the *continuity* of student work in an integrated ILE. The results of students' work with any of the components during the session should be taken into account by other components to adapt their performance to the changed knowledge level and current interest of the particular student.

As a solution for the above two problems we suggested a student model centered approach to building integrated ILE. By this approach all the components of ILE including the environment and the manual use the same central student model, a traditional part of ITS, to adapt its behavior to the given student. We also designed a simple student model centered architecture of an ILE. Since 1985 we applied the approach and the student model centered architecture in several ILE designed by our group for different domains. We use these ILE to investigate various aspects and problems of integrated ILE. We consider our approach as fruitful and effective, however years of experience enable us to find the limitations of our original architecture. Recently we have adopted an advanced student model centered architecture for ILE which we use in our most recent ILE.

In this paper we present our original approach and the simple student model centered architecture for ILE, report some problems and limitations of our original architecture, and present an advanced student model centered open architecture for ILE.

Background

First goal of the approach is to build a *really adaptive* ILE where *all the components* can adapt dynamically to the student. level. There is a range of characteristics that can be taken into account by ILE components to tailor its behavior to the given student: personal factors, cognitive styles, strategies, personal knowledge (van der Veer 1990). The key (and the most changeable) characteristic of the student from educational point of view is the student's knowledge on the subject. In our work we concern primarily the component of the general student model which represents student knowledge.

Here a good background is provided by the research on **intelligent tutoring systems** (Wenger 1987). A characteristic shared by many ITSs is that they infer a model of the student's current understanding of the subject matter and use this individualized model to adapt the instruction to the student's needs. The domain of ITS is a good source of ideas how to design student models and how to use it by the tutor or coach components of ILE (Self 1987; VanLehn 1988). In the same time it provides very few ideas how to use it by the environment and manual components.

The ideas about creating an adaptive manual can be found in the field of **intelligent help systems** (IHS) which has deep roots in ITS research (Breuker 1990). An IHS aims to support a user working with an application system. An IHS provides the user with passive assistance (answering the student's questions) and active assistance (revealing wrong and suboptimal behavior and incrementally extending the student's knowledge). An IHS uses overlay *user models* to tailor the answers and the explanations to each individual user's knowledge level.

The ideas how to use the student model by the environment component of ILE can be found in the domain of **adaptive user interfaces**[1] (Dieterich et al. 1993). This relatively new field (compared to that of ITSs) studies interfaces that adapt themselves to suit the characteristics of the user. A key part of an adaptive interface is the user model which represents those characteristics of the user that are important for the purpose of adaptation. The model of user knowledge about the domain is an important part of the general user model. Because of the similarity of this data to that held by an ITS, Benyon and Murray (1993) refer to this portion of the user model as the student model.

The ideas from the domain of adaptive interfaces can be applied to make an adaptive environment component for an ILE. To do that we have consider the environment as a regular application system and the student as a user of this system (Brusilovsky 1993). One can argue, that an environment component based on adaptive interface ideas can adapt to the student knowledge about *the environment*, not about *the domain* being learned. Note however, that any essential feature of an educational environment represents some knowledge about the domain.

Taking altogether, the domains of ITS, IHS and adaptive interfaces form a good background to achieve the first goal of our approach, i.e. to build an adaptive ILE where all the components can adapt dynamically to the changing student knowledge. The second goal of the approach is to have single representation of student's knowledge in the student model of ILE, which can be used by all the components of ILE. This feature provides continuity: the results of students' work with any of the components which may influence student's knowledge level are immediately reflected in the student model and can be taken into account by other components which adapt their performance to the changed student knowledge. To achieve this goal we have to design a unified student-user model and, more generally, to design a student model centered architecture for an ILE supporting student modeling and cooperative use of the student model. Next section presents simple student model centered architecture which we have used formerly in our several ILE. The subsequent section presents an advanced student model centered architecture which we are using now.

[1]Some special areas of research within the area of adaptive interfaces provide also a source of ideas for the adaptive manual component of ILE.

Simple student model centered architecture

In our work on student model centered ILE we were going from the ITS side, thus we adopted traditional ITS architecture as a basis for the architecture of student model centered ILE. The traditional ITS architecture includes three main components: the expertise component, the tutoring component, and the student modeling component. Each of the components localizes one of the three kinds of knowledge important for intelligent tutoring: knowledge about domain, knowledge about tutoring, and knowledge about student and student modeling (Wenger 1987). According to this architecture, the student model represents the student's understanding of material to be taught. The student model is used by the tutoring component to provide adaptive tutoring on various levels. The results of student's work with teaching operations are returned back to student modeling (diagnostic) component and used to update the student model. This is called the student modelling loop.

To use ITS experience in student modelling we decide to apply the regular student model being used by the tutoring component of an ILE as the central student-user model for overall ILE. In our first systems the tutoring component provides regular student modelling loop, while other components of the ILE just use this central student model for adaptation. The only problem was to choose the kind of student model which can be used by all the components.

As we can see from the previous chapter, ITS, IHS and adaptive interfaces use student or user models for the same purpose of adaptation, what also results in the similarity of the models applied. If we consider the student knowledge component of student or user models, we will find similar *overlay* models based on the structure model of the domain, where the domain is either the subject to be taught or the application system. For each element of the domain knowledge the student (user) model stores some data about the student's (user's) competence and previous experience with this element.

In our student model centered architecture the domain model is a network whose nodes correspond to elements of subject knowledge (depended from the subject) and whose links reflect several kinds of relationships between nodes. We use overlay model, which contains one integer counter for each subject knowledge element measuring student's understanding of this element. This kind of overlay model is powerful and general enough to be used by different components of ILE. The student model is kept updated by the special evaluation module which analyzes the results of student' problem solving activity. If an ILE contains the coach component which can follow student step-by-step problem solving, then a kind of model tracing technology can be applied (Corbett & Anderson 1992), otherwise a kind of differential modelling (Wenger 1987) is used to update the counters of concepts related to the problem. The changes are propagated along the network links.

The above overlay model is accessible for all the modules of ILE and can be used by each of them to adapt its behavior to the student knowledge. However to avoid using senseless numbers and to provide more flexibility we suggested a *threshold* technique. Each of the ILE components can distinguish several distinct knowledge states for each knowledge elements. Each of these states has special meaning for the module from the adaptation point of view. The more states a module can take into account the more complex adaptation it can provide. Simple modules can distinguish only two states - for example *unknown* and *known*, while the most adaptive tutoring module can distinguish six states (Brusilovsky 1992a). To map a particular integer value of the overlay model into a set of states each module use integer thresholds which divide the possible range of values of the counter into required number of intervals corresponding to knowledge states recognizable by the module. Thus simple modules use one threshold only, while the tutoring module uses five thresholds. Each module use own set of thresholds over the central student model. These thresholds can be different for different knowledge elements and different students. The threshold technique provides a good flexibility, giving the way to adapt the student modelling mechanism to the knowledge elements of different difficultly and to different classes of students.

We applied the above student model centered architecture in several ILE for different domains. These ILE have the same overall architecture, but use different sets of modules and demonstrates several possible ways of applying the overlay student model for adaptation. Below we briefly describe in this context some ILE designed by our group along the simple student model centered architecture.

ITEM/IP is an ILE for learning introductory programming (Brusilovsky 1992b). The domain knowledge elements in ITEM/IP are general programming concepts and constructs of the programming language being learned. ITEM/IP contains the following adaptive modules: the strategy module which supports adaptive sequencing of teaching operations, the visual interpreter which uses the student's current knowledge level to provide adaptive error handling and adaptive visualization, and the presentation module which generates an adaptive description of a concept or a construct when introducing or repeating it. All these modules refer to the same six knowledge states (five thresholds) for each domain knowledge element in its adaptation rules[2]. More details about these components can be found in (Brusilovsky 1992a,1993).

ILED is an ILE for acquiring the skill of derivation in calculus (Brusilovsky V. 1993). The elements of the domain knowledge in ILED are rules and malrules of derivation. ILED includes the following adaptive modules: the structural formula editor which plays the role of exploratory environment, the tutor which can suggest the best teaching operation (problem or example) to the student, and the coach which follows the student actions step-by-step, diagnosing errors and updating the student model. New features of ILED comparing to ITEM/IP are adaptive structure editor[3], adaptive coach and the tutor's ability to *generate* (vs to select) the best teaching operation on the base of the student model. The structure editor distinguish two states for the derivation rules – *not-acquired* and *acquired*. The tutor and the coach distinguish four states for the derivation rules - *unknown*, *introduced*, *known*, and *acquired*.

ISIS-Tutor (Brusilovsky & Pesin 1994) is an ILE to support learning the print formatting language of an information retrieval system CDS/ISIS. ISIS-Tutor resembles the architecture of ITEM/IP in many details. A new adaptive component of ISIS-Tutor is a hypermedia manual which originates from the presentation module of ITEM/IP. This component supports both adaptive concept presentation and adaptive hypermedia navigation. The hypermedia component of ISIS-Tutor distinguish three knowledge states for each concept: *not-ready-to-be-learned*, *ready-to-be-learned*, and *known* (a concept is ready to be learned if all the preceding concepts are known to the student).

The methods of adaptation used in the above projects are rather simple. The goal was not to improve the known methods of adaptation of various components, but to build a system where most of the modules can use the same student model to adapt their performance, in various ways, to the knowledge of the given user. On the further steps some simple methods of adaptation can be replaced by more sophisticated technologies developed in the fields of intelligent interfaces and intelligent help systems. Some examples are user–adapted natural language explanations (Paris 1988), strategy–based intelligent help (Breuker 1990), and adaptive hypermedia help (Böcker, Hohl & Schwab 1990).

Lessons learned

Our experience with several ILE designed along the student model centered approach proves that it is a generally good way for building integrated ILE. We feel now that the student model can play the role of a kernel of an ILE. We demonstrated that in several domains it's possible to build ILE where most of the modules can use the central student model for adaptation.

On another side, our experience showed us serious problems and limitations of our simple student model centered architecture. This limitations become clear when we start working on our recent ILE - ISIS-Tutor (Brusilovsky & Pesin 1994) and ITEM/PG (Brusilovsky & Zyryanov 1993). Both systems applies the hypermedia manual as a component for student-driven browsing of domain knowledge. The hypermedia provides new quality and the students working

[2]It is the experience of ITEM/IP which force us to use different thresholds for different modules.

[3]ITEM/IP also contains a structure editor, but it was not adaptive

with these systems (unlike with original ITEM/IP) spend serious amount of time learning with hypermedia on their own. It was obvious that the results of student's work in hypermedia have to be reflected in the student model. The problem is more general: in an adaptive learning environment not only can each module of an ILE use the student model for the purpose of adaptation, but also each module can influence the student model, reflecting an experience that the student has demonstrated while working with this module. Thus the diagnostic component have to loose the traditional ITS monopoly for student model updating. Unfortunately, it appears to be quite difficult to coordinate several sources of student model update in the simple architecture. We tried to do that in ISIS-Tutor, but was not satisfied with the result.

Another problem is that the student model of a classic ITS which we inherited with the simple approach was designed to accumulate and process the information about the student according to the needs of the tutoring module. The information stored in the central model is relevant for the purposes of a tutor or coach, but our experience showed that other modules of an ILE can need quite different information about the student according to the kind of adaptation they provide. A part of this problem can be solved by our threshold technique. However the main problem is that processing the information about the student into a form oriented to one of the modules often leads to the loss of information important to some other module. For example, the hypermedia component needs the information how often the tutoring component presents a particular hypermedia page to the student. This information was used to update some counter in the student model and then erased. Now it can not be reconstructed from the student model.

The third problem was encountered when we tried to make the student model changeable by the student. The reason for the student to change the student model is to customize the adaptation of a particular component. However, any changes in the student model result in changed behavior of all the components, what is not the goal of the student.

The above considerations lead us to revise the simple student model centered architecture. We have done it in several steps, immediately applying and checking new decisions in our ITEM/PG and ITEM/IP-II environments. Next section presents an advanced student model centered architecture for ILE where each of the components (modules) can use and/or update the student model without the loss of information.

Advanced student model centered architecture

The student model centered architecture separates the student model into two parts, the main student model and the projections (figure 1). The main student model (we will keep the name *student model* for it) stands in the center of the environment and collects the information about the given student from different sources. Student interaction with any of the system components is reported to the student model. Examples: "at time T_1 the student visits the hypernode for the concept C_1 for S_1 seconds", "at time T_2 the student was presented with the problem solving example which concerns concepts $C_1, \ldots, C_n$", "working with the editor at time T_3 the student used successfully concepts $C_1, \ldots, C_m$". These reports are time stamped and stored in the form of *standard events* directly related to domain model nodes. No further processing is performed in order to avoid the loss of important information. The main student model unifies all the information about the student which can be used for the purpose of adaptation.

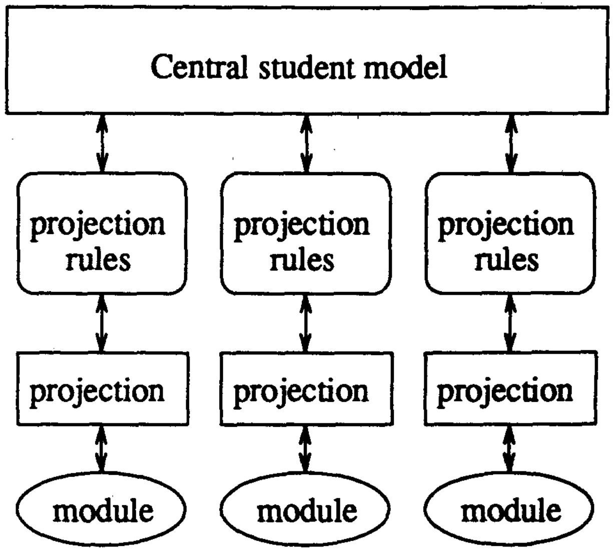

Figure 1: Advanced student model centered architecture

The components of an ILE do not use the student model directly, but instead use local views on the student that we call *projections*. A projection represents that information about the student, which is essential for the component to adapt its work to the student. A component has as rich and wide projection as it needs for the purpose of adaptation. A projection is built and updated from the main student model by a special set of rules called a projector. Each component has its own projection and projector, which provides the interface between the component and the main student model. One part of the projector rules is used to project the main student model into the local projection. These rules refer to the student model in their left hand sides and contain commands to update the projection in their right hand sides. Example: "if the student *read the description of the construct C_i* and the student *watch the work of the construct C_i with first level of visualization* more then 15 times then *set second level of visualization for the construct C_i*". Another part of the projector rules is used to provide reverse projection: to project if required the results of the student's work with the component into the form of standard events used by the main student model. For example: "if *at time T_1 the student visits the hypernode for the concept C_i* for more then 30 seconds then *at time T_1 the student read the description of the concept C_i*". More examples of using projections in real a ILE can be found in (Brusilovsky & Zyryanov 1993).

>From our point of view, the student model of a classic ITS is just one of the local projections: the one used by the tutoring component. Other components of the system (such as the microworld) can use quite different projections. The main student model stores partly processed information about the student, because further processing can lose information important to one of the components. The student model is more than a traditional "history", but is less formalized than a classic overlay model. Rather, it is a *structured history*. Further processing and projecting to more traditional overlay form is made separately by the projectors according to the requirements of different component.

We think that the proposed student model centered architecture is a good basis for creating an integrated learning environment, or any other integrated adaptive system which consists of a set of different components. The use of projections and rules provides the open architecture with a good degree of flexibility. Since the performance of the component depends on the projection, we can tune the performance for a particular use by changing the projector rules (or even the projection itself) without influencing other components. A new component can be easily integrated into the environment by designing a set of rules which connect the central model with the given component and its local view on the student. If a new component requires new forms of interaction which cannot be projected into the existing set of standard events, this set can be extended. For example, an event "at time T_1 the student *heard* the presentation of the concept C_i from a multimedia record" can be projected into event "at time T_1 the student read the description of the concept C_i" or can be recorded as a new kind of event. If a particular module needs to take into account a new kind of events for better adaptation, its projector can be updated. Thus the open architecture localizes and minimizes changes in a developing system.

We should note that similar architectures was suggested by other authors for user modeling in quite different domains (Kay 1990; Sukaviriya & Foley 1993; Kobsa, Müller, & Nill 1994). It gives us an implicit confidence that our advanced architecture is general enough to be used in a number of domains.

Discussion

An important problem which has to be discussed in the context of the proposed student model centered architecture is the relevance of adaptation. The system can use very elaborate strategies to provide the student with the "optimal" next teaching operation, level of visualization, or help detail. The problem is whether the student agrees with the choice. The student could prefer a different operation, more (less) concise visualization, or more (less) detailed help. To deal with this problem, we think, the adaptation should not be intrusive, and the student at least should be provided with a choice: to accept the system-provided automated adaptation or to switch the adaptation off. Our experience with task sequencing (Brusilovsky 1992a) shows that novices tend to agree with the system choice, while experienced students often prefer to make their own choice from the complete list of relevant teaching operations. In ITEM/IP the student has a choice between adaptive and detailed visualization, between adapted and complete on-line help.

Next step is to provide the student with the possibility to "adapt the adaptation" or to customize the adaptation mechanism if she is not satisfied with the adaptation. The student model provides a good high level tool for the student to control the adaptation. This leads us to the area of viewable (or inspectable) and changeable student models. These ideas become more and more popular in ITS field now (Self 1988; Corbett & Anderson 1992). A number of work towards this direction have beed done in the field of adaptive interfaces (Kay 1990; Böcker, Hohl & Schwab 1990). The book (Schneider-Hufschmidt, Kuehme, & Malinowski 1993) provides a good generalization of these efforts and suggest a taxonomy of different kinds of cooperative adaptation, where the part of the job is done by the system and another part by the student.

We think that the proposed advanced student model centered architecture provides a good basis for different kinds of cooperative adaptation. This architecture enable the student to control separately the adaptation mechanisms of different system components. The tuning of a particular projection or adaptation mechanism will not influence other parts of the system. Some good ideas about the tuning of adaptation mechanism for the case of hypermedia can be found in (Kaplan 1993)

References

Benyon, D.R. and Murray, D.M. 1993. Applying user modeling to human-computer interaction design. *AI Review* 6: 43–69.

Böcker, H.-D.; Hohl, H.; and Schwab, T. 1990. $\Upsilon\pi$*Adapt*$\epsilon\rho$ - Individualizing Hypertext. In Proceedings of the the Third International Conference on Human-Computer Interaction, INTERACT'90, 931–936. Amsterdam: North-Holland.

Breuker, J. ed. 1990. *EUROHELP: Developing intelligent help systems. Final Report on the P280 ESPRIT Project EUROHELP*. Brussles: EC.

Brusilovsky, P.L. 1992a. A framework for intelligent knowledge sequencing and task sequencing. In Proceedings of the Second International Conference on Intelligent Tutoring Systems, ITS'92, 499–506. Berlin: Springer-Verlag.

Brusilovsky, P.L. 1992b. Intelligent Tutor, Environment and Manual for Introductory Programming. *Educational and Training Technology International* 29(1): 26-34.

Brusilovsky, P. 1993. Student as user: Towards an adaptive interface for an intelligent learning environment. In Proceedings of World Conference on Artificial Intelligence and Education, AI-ED'93, 386–393. Charlottesville: AACE

Brusilovsky, P.; Pesin, L.; and Zyryanov, M. 1993. Towards an adaptive hypermedia component for an intelligent learning environment. In Bass, L.J; Gornostaev, J; and Unger, C. eds. *Human-Computer Interaction.*, 348–358.

Berlin: Springer-Verlag.

Brusilovsky, P. and Zyryanov, M. 1993. Intelligent Tutor, Environment and Manual for Physical Geography. In Proceedings of the Seventh International PEG Conference, PEG'93, 63–73. Edinburgh.

Brusilovsky, V. 1993. Task sequencing in an intelligent learning environment for calculus. In Proceedings of the Seventh International PEG Conference, PEG'93, 57–62. Edinburgh.

Brusilovsky, P. and Pesin, L. 1994. ISIS-Tutor: An adaptive hypertext learning environments: In Proceedings of Japan-CIS Symposium on knowledge based software engineering, 83-87. Tokyo: Isshinsa, Ltd.

Burton, R.R. 1988. The environment module of intelligent tutoring systems. In Polson, M.C. and Richardson, J.J. eds. *Foundations of intelligent tutoring systems*. Hillsdale: Lawrence Erlbaum Associates.

Corbett, A.T. and Anderson, J.R. 1992. Student modeling and mastery learning in a computer–based programming tutor. In Proceedings of the Second International Conference on Intelligent Tutoring Systems, ITS'92, 413–420. Berlin: Springer–Verlag.

Schneider-Hufschmidt, M.; Kuehme, T.; and Malinowski, U. eds. 1993 *Adaptive User Interfaces: Principles and Practice*. Amsterdam: North Holland Elsevier.

Kaplan, C.; Fenwick, J; and Chen, J. 1993. Adaptive Hypertext Navigation Based on User Goals and Context. *User Modeling and User Adapter Interaction* 3(3): 193–220.

Kay, J. 1990. um: A toolkit for user modelling. In Proceedings of the second international workshop on user modelling, 1-11. Honolulu.

Kobsa, A.; Müller, D.; and Nill, A. 1994. KN-AHS: An adaptive hypertext client of the user modeling system BGP-MS. This volume.

Paris, C.L. 1988. Tailoring object description to a user's level of expertise. *Computational Linguistics* 14(3): 64–78.

Self, J. 1987. Student Models: What Use are they? In Ercoli, P. and Lewis R. eds. *Artificial Intelligence tools in education*. Amsterdam: North-Holland.

Self, J. 1988. Bypassing the intractable problem of student modelling. In Proceedings of the Intelligent Tutoring Systems conference, ITS'88, 18–24. Montreal.

Sukaviriya, P. and Foley, D. 1993. A built in provision for collection individual task usage information in UIDE: the User Interface Design Environment. In Schneider-Hufschmidt, M.; Kuehme, T.; and Malinowski, U. eds. *Adaptive User Interfaces: Principles and Practice*, 225-240. Amsterdam: North Holland Elsevier.

van der Veer, G.C. 1990. *Human–computer interaction: learning, individual differences, and design recommendations*. Alblasserdam: Haveka.

VanLehn, K. 1988. Student models. In Polson, M.C. and Richardson, J.J. eds. *Foundations of intelligent tutoring systems*. Hillsdale: Lawrence Erlbaum Associates.

Wenger, E. 1987. *Artificial intelligence and tutoring systems. Computational approaches to the communication of knowledge*. Los Altos: Morgan Kaufmann.