An Adaptive Time Management System for Student Learning

Andrew M. Gravell
School of Electronics & Computer Science, University of Southampton, UK, amg@ecs.soton.ac.uk

Till Rebenich
School of Electronics & Computer Science, University of Southampton, UK, tr306@zepler.net

Abstract: We present a modular framework for an adaptive and position-aware student time management system, and a prototype implementation distributed between a desktop PC and a mobile device (PDA). The system uses an adapted version of Soloman & Felder's Index of Learning Styles questionnaire to determine the student's learning style. This is matched with the teaching style of module, acquired by using a complementary teaching style questionnaire, to create an individual study plan for a user-defined learning task hierarchy. Microsoft Outlook and Pocket Outlook are used as a user frontend for this plan. Based on user feedback the schedule is continually adapted using a multi-layered neural network in combination with an iterative back-propagation learning algorithm to adjust the time devoted to a particular task. Experiments have shown that the system adapts satisfactorily to individual student requirements. The neural network error threshold and an additional boosting factor control its adaptation behaviour. The mobile part of the system uses GPS data to track the current user position and launches position-related reminders. In practice, this mechanism was found to be unreliable in high-density areas, and very power intensive. The novelty of our approach is its comprehensive character, combining aspects of education theory, time management, machine learning, and position-awareness technology in a single framework. Our prototype shows its practical applicability. Remaining work includes the integration into the university computing infrastructure and a thorough evaluation by a representative group of students.

1 Introduction

This paper presents the prototype of an adaptive software system, focussed on the university education domain, aimed at supporting students in planning study activities, and addressing two of the main student issues: time-management and study organisation.

Universities require students to learn independently. This involves coping with course objectives, planning learning tasks (Marshall & Rowland, 1998), and sustaining a workload necessary to achieve learning goals. To meet these requirements, students need to acquire a certain degree of self-organisation driven by "inner motivation" and self-discipline (Payne & Whittaker, 2000). The flexible nature of university courses leaves a great deal of responsibility to the student (Moore, 1973). Universities merely set learning goals, but are also support systems offering learning materials and employing teaching staff. It is therefore more appropriate to speak of students as "autonomous learners" (Broad, 2006). Although they are dependent on lectures, information sources, and university facilities, the teacher's role moves from that of a "director of learning" towards "resource" and "helper". An autonomous student is thus "responsible for his own learning" (Moore, 1973) and shows a number of traits.

These are (1) the inclination towards planning ahead, (2) the ability to commit to a plan and to adapt it, (3) effective time management, (4) open-mindedness towards new things, reading, writing, listening, and discussing, (5) questioning, testing, analysing, and abstraction of basic ideas, (6) note-taking, remembering, and establishing relationships between materials, and finally (7) cooperation with other learners while learning on their own (Moore, 1973).

In contrast, the most frequent difficulties encountered by students are their inability to organise and plan work properly, using study time effectively, handing in assignments on time, and structuring material (Main, 1980). Our software supports students in adopting traits 1 to 3 by creating a personalised study schedule, while traits 4 to 6 are not addressed explicitly. The system framework is extensible to support collaboration and cooperation, and is not only aimed at students lacking self-organisation but also at experienced time managers. To achieve this objective, we use machine learning to adapt to the user's individual learning progress as the scheduled plan is being followed.

But how do we measure the time for a particular learning task and how do we determine whether it was spent effectively?

When coping with learning tasks, students pursue two phases. In the first phase, they approach them dependent on their curiosity/interest in the subject, and in the second phase they deal with them over time. With regard to the first phase, (Prosser & Trigwell, 1999) mention the *deep* and the *surface approach*, i.e. a qualitative definition of how intensively students deal with learning material presented to them. (Richardson, 1994) later added the *strategic approach*, which characterises the degree to which marks influence this process. Other sources (Saunders, 1994) focus on temporal aspects: *reactive* students are prone to procrastination, whereas *proactive* students do things earlier rather than later.

In the second phase, learning styles come into play. Again, we find many competing or supplementary learning style models, focussing on three aspects: the learning environment, the structure of learning material, and individual learning skills based on past experience. Regarding the latter aspect, researchers came up with dimensional models. A simple, one-dimensional model by (Main, 1980) distinguishes *syllabus-bound* and *syllabus-free* students. The former require learning incentives derived from structural/temporal regulations (e.g. hand-in dates), the latter prefer to be more adventurous and experimental.

(Kolb, 1984) presents a two-dimensional model, yielding four learning styles of experiential learning. Convergers focus on active experimentation and abstract conceptualisation, divergers use a combination of concrete experience and reflective observation, assimilators apply reflective observation and abstract conceptualisation, and accommodators active experimentation in combination with concrete experience. Kolb's learning style model is widely used in study skills literature, e.g. (Payne & Whittaker, 2000) and (Cottrell, 2003), whereby the latter source uses more figurative terms (diver, dreamer, logician, and searchlight).

The Felder & Silverman learning style model (Felder & Silverman, 1988) adds another perspective. In addition to the way a learner learns, it focuses on the learning environment and its impact on the learner, which can be more or less stimulating. The model comprises five dimensions, namely processing, perception, input, understanding, and organisation. Its key feature is the incorporation of teaching *and* learning style. A learner's performance is the result of a comparison between these two styles: the more discrepancies the poorer the learning outcome. Felder & Silverman's model is also the foundation of the Index of Learning Styles (ILS) (Soloman & Felder, 2001), which determines a user's learning style from a questionnaire comprising 44 questions, 11 for each of four dimensions; the organisation dimension is not included (Zywno, 2003). In cases where learning and teaching style in a subject do not match, the poor learning outcome can be compensated by devoting more time to it. Conversely, less time must be spent on subjects where both styles match closely. Thus, learning style models can help us measure the amount of time, and they are explicit indicators of effectiveness.

Apart from learning and teaching, there are other aspects influencing time: the location, the individual "body-clock" (Payne & Whittaker, 2000), and motivation (Main, 1980). Study guides (Marshall & Rowland, 1998) suggest that learning activities should be performed at the "right" location, e.g. the library for literature-intensive tasks. Moreover, the best outcome is achieved when they are performed at peak energy times, and motivation is needed to start on a task and to move on the next one.

2 A Framework for Student Time Management

We then derived requirements from the above findings and applied an object-oriented and architecture-driven design using UML 2.0 as a modelling language.

2.1 Requirements

Being an autonomous learner means defining personal study objectives, deadlines, and associating study material and other resources with the subject so that they can be accessed easily and quickly. The system should endeavour to prevent the procrastination of study activities, especially if they are perceived difficult or complex.

Such activities are the result of a divide-and-conquer approach applied to the predefined university course structure. We distinguish structural elements of organisational/hierarchical nature, and teaching units embedded in this organisational apparatus. A module provides one or more learning units of weight w $(0\% < w \leq 100\% \wedge \Sigma w_i = 100\%)$, e.g. examinations or assignments, broken down into several tasks and sub-tasks which are to be performed to meet a learning unit goal. (Cottrell, 2003) calls these sub-tasks "mini-goals". All tasks are integrated (relating to a larger plan, project or subject), manageable and realistic, specific (containing description of the expected outcome), measurable, and flexible (allowing for unforeseen circumstances). We adopt the time management cycle shown in **Figure 1**.

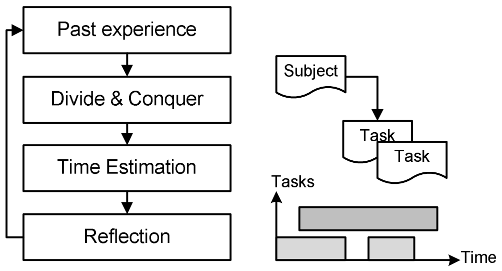


Figure 1: Time management cycle (based on (Cottrell, 2003; Drew & Bingman, 2001))

Each task is assigned a specified time to target, dependent on the student's learning style and the teaching style in the corresponding module. Here the system must maintain the work-life-balance, i.e. tasks are not scheduled at times devoted to leisure activities. Moreover, individual time patterns such as the preferred time of day for study, learning slot length, and breaks between slots must be considered.

The scheduled plan should be accessible to the user at all times, preferably on a mobile device (PDA), integrated into existing personal information manager (PIM) architectures. This enables the use of position-awareness technology to ensure that learning tasks are performed at the "ideal" location.

2.2 Architecture

The system comprises five basic components (**Figure 2**), three of which are deployed to a mobile device. The master data library handles non-volatile data, e.g. faculties, schools, courses etc., manageable through a graphical user interface (GUI), which is also used for triggering planning-related functions located in the planning library. The main components here are the planner, creating the study plan, and the synchroniser synchronising it with a PIM. All data is stored in databases.

On the mobile device, the study plan is used together with position-awareness technology (GPS) to introduce a location dimension into the reminder mechanism, i.e. students are notified *when* and *where* to start on a task. The framework is extensible by other plug-ins, e.g. navigation systems.

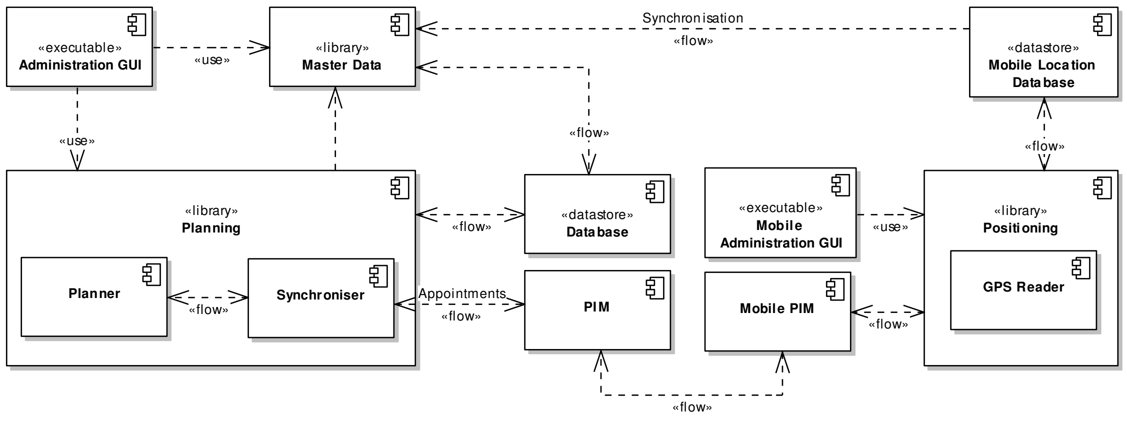


Figure 2: System architecture (as component diagram)

2.3 System Design

The master data library contains classes representing the university's organisational structure, i.e. multiple faculties containing multiple schools. The latter provide different types of courses, which again hold multiple modules offered in a particular semester. A module provides learning materials covered during a semester. For each association between a module and a semester, classes (e.g. lectures) take place periodically at a specified location.

Besides, we define multiple learning units to be performed per module. They are assessed by examination or coursework, and broken down into a nested task hierarchy consisting of tasks and sub-tasks. We follow the recommendations made in (Cottrell, 2003; Leung & Li, 2003; Vavoula & Sharples, 2002), which use different terms for these elements. We also introduce actors, namely students and members of staff. A student is associated with exactly one user account, augmented by personal planning preferences. This data structure contains the individual learning time patterns, slot and break lengths, and system-related preferences controlling the planning behaviour.

2.3.1 Planning

The planning library handles variable data, including the user's module selections and its difficulty assessment, the learning tasks, and the calendar entries scheduled for each of them. In this context we distinguish predefined from adopted or user-defined tasks. The university defines a set of recommended tasks for each learning unit, and the user can define their own personal tasks either from scratch or by adopting predefined tasks. A task is also assigned a priority, "composed of two elements in various mixtures: urgency and importance" (Adair, 1982), a complexity level, and an order criterion determining its position in the task list, implying that it is always dependent on the completion of another task. Learning and teaching styles are four-dimensional data structures holding a numeric value for every dimension (Soloman & Felder, 2001).

Moreover, controller classes hold complex algorithms for planning and synchronisation. We use a Strategy design pattern (Shalloway & Trott, 2005) to separate the abstract algorithm definition from its implementation.

The *synchroniser* controls the synchronisation of calendar entries between PIM and system database, whereas the *planning algorithm* schedules calendar entries for each personal task. A calendar entry has a start and an end date, and can be associated with a location specified in the task. We distinguish personal calendar entries, imported from an existing PIM, module calendar entries (classes or lectures), and task calendar entries. For planning and performance reasons, a *slot manager* divides a planning period into a sequence of uniform slots $P=(S_1,S_2,S_3,...,S_n)$ where n denotes the number of slots contained in the period and S_i the slots. n is defined as $\frac{l}{d}$ with l being the length of the period in minutes subject to the constraint $d-3\cdot\max\{n\in \mathrm{Z} \mid n\leq\frac{d}{3}\}=0$ (cf. **Figure 3**).

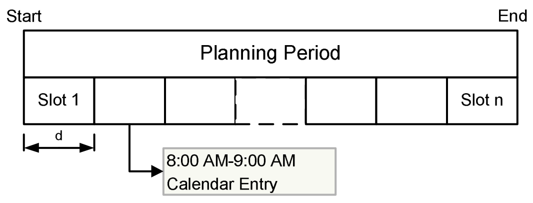


Figure 3: Planning slot structure

A slot can have one of two states $\{\text{free}, \text{occupied}\}$ and is only occupied by one calendar entry at a time.

2.3.2 Neural Network

The adaptive system behaviour is controlled by the *neural network manager*, a proxy for the neural network library (Fleurey, 2002). It invokes the learning algorithm with training samples created from user feedback gathered after each passed calendar entry, which is the task's overall percentage of completion. We use a multi-layered neural network comprising 7 inputs and 6 layers including the output layer, and a back-propagation learning algorithm (Nilsson, 1998, p. p. 46). Its input vector

$$\vec{x}=(u_1,u_3,u_3,u_4,m,c,n)^T$$

consists of

- The differences $u_1...u_4$ between learning and teaching style of the task's module, gathered by using a simplified version of the ILS (Soloman & Felder, 2001) comprising only a random selection of 5 out of 11 questions per dimension. We had to conceive a complementary teaching style questionnaire, taking the learning style questions as templates. This yields a total of 20 questions per questionnaire and values ranging from 0 to 10 for the differences per dimension, such that $u_i=|p_i-q_i|$ with p_i being the learning and q_i the teaching style value of the i th dimension.
- A module difficulty m between 0 and 4 as assessed by the user when choosing their modules
- A task complexity c between 0 and 4
- The task importance n, which is either 0 (not important) or 1 (important)

This results in a total of $11^4 \times 5^2 \times 2 = 732{,}050$ different input vectors. The output y is a floating point value between 0 and 1. After applying the linear scaling $b = 2y$, we receive values between 0 and 2, which are used directly to determine the scheduled task time $t = t_0 \cdot b$ with t_0 being the base task time excluding the adaptive factor.

Initially, we take an artificial set of training samples containing only significant input vectors with $u_i = \{0,5,10\}$, $m, c = \{0,2,4\}$, and $n = \{0,1\}$, restricting the set of possible vectors to 1,458, which are then presented to the network. The desired output is:

$$y' = 0.5 + \left(0.015\sum_{i=1}^{4} u_i + 0.007m + 0.012c + 0.025n\right)$$

2.3.3 *Positioning*

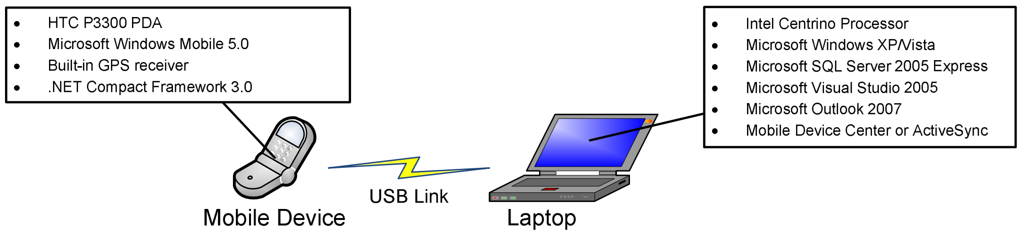
The GPS positioning component uses a polling algorithm to read GPRMC sentences from the serial port of the mobile device. A location is identified by its name and refers to a position, comprising a latitude p_t and a longitude p_l. The *reminder manager* detects whether an appointment C contains a reference to a known location. If the current position does not match the position specified for this location, the device launches a reminder following the function

$$R(C) = \begin{cases} 1 & p_t \in \left[(p_t(C) - \Delta p), (p_t(C) + \Delta p)\right] \wedge p_l \in \left[(p_l(C) - \Delta p), (p_l(C) + \Delta p)\right] \\ 0 & \text{else} \end{cases}$$

where Δp is a positional tolerance specified by the user. The precision can be configured by evaluating the mean dilusion of precision (DOP) of GPGSA sentences.

2.4 Implementation

Our prototype is a Microsoft .NET application developed in C# with Visual Studio 2005 and executed on a Windows XP/Vista platform. We use Windows Mobile 5.0 on a HTC P3300 PDA with a built-in GPS receiver. Outlook 2007 serves as personal information manager, and its counterpart on the PDA is Pocket Outlook. The synchronisation between desktop and PDA is completely opaque to our system since ActiveSync or the Mobile Device Center deals with this issue. Application-specific data is stored in SQL Server 2005 Express Edition. We chose these technologies to provide an execution environment as homogeneous as possible (cf. **Figure 4**).



HTC P3300 PDA
Microsoft Windows Mobile 5.0
Built-in GPS receiver
.NET Compact Framework 3.0
Intel Centrino Processor
Microsoft Windows XP/Vista
Microsoft SQL Server 2005 Express
Microsoft Visual Studio 2005
Microsoft Outlook 2007
Mobile Device Center or ActiveSync
USB Link
Mobile Device
Laptop

Figure 4: System hardware setup

The core system component is the *planner*. During a *planning cycle*, the following steps are performed:

1. Initialisation (preparing neural network and planning period)
2. Outlook synchronisation (taking stock of existing appointments)
3. Scheduling of module calendar entries (creating periodical appointments for classes, lectures etc.)
4. Grouping of leaf tasks
5. Feedback processing
6. Neural network training
7. Creation of planning items

8. Scheduling of task calendar entries (the appointments created for leaf tasks)
9. Outlook synchronisation (submitting changes)

2.4.1 *Grouping of Leaf Tasks*

Principally, only leaf tasks, i.e. tasks at the bottom level of the hierarchy, are considered during the planning process. They are grouped according to the associated learning unit and their position in the task list. For this purpose, we define a task order criterion denoting the position in the sub-task list of its parent relative to the hierarchy.

2.4.2 *Feedback Processing and Neural Network Training*

The adaptation mechanism is dependent on user feedback provided after each passed calendar entry. It consists of a real number defining the task's percentage of completion. A 100% complete task is no longer considered by the planner and all associated calendar entries are removed. A feedback is a training sample

$$H = (\bar{x}, y, y')$$

with $\bar{x}$ being the input vector, y the output before training, and y' the desired output. The latter two values are real numbers determining the factor applied to the planned task time t_0. y' is defined as

$$y' = y - g(k_{i+1} - (k - k_{i-1}))$$

where k_{i+1} is the planned percentage of completion after a calendar entry has passed, k_{i-1} the old percentage of completion, and g a boosting factor expediting the neural network learning behaviour. Values between 3 and 5 for g were found to be most effective.

2.4.3 *Creation of Planning Items*

For each grouped leaf task on the list, a so called planning item is created. It consists of the task itself and the time t_0, which is made up as follows:

- The minimal time t_m (in minutes) to be spent on a module, dependent on the number of credit points a gained for the module: $t_m = 10 \cdot 60 \cdot a$.
- The weight w (in percent) of a learning unit is used to distribute t_m over the units associated with a module, such that $t_u = t_m \frac{w}{100}$ with $\sum w_i = 100$.
- The task time t_0 is determined by setting the task complexity c in relation with n complexity values of all other leaf tasks of the same learning unit:

$$t_0 = \frac{c+1}{5\sum_{i=1}^{n} \frac{c_i+1}{5}} t_u$$

We further define

- The remaining time $t_r = t_0 \frac{100-k}{100}$ after reducing the task time by the percentage k already completed.
- The total time $t_t = t_r \cdot b$ including the neural network output factor b.

2.4.4 *Scheduling of Task Calendar Entries*

Once the total time is known for a task, it is split into calendar entries of user-defined length, leaving a break at the end. It is scheduled with the help of the slot manager, which holds a bitmap of all slots in the specified planning period, complying with the user's body clock time patterns and preserving the task grouping.

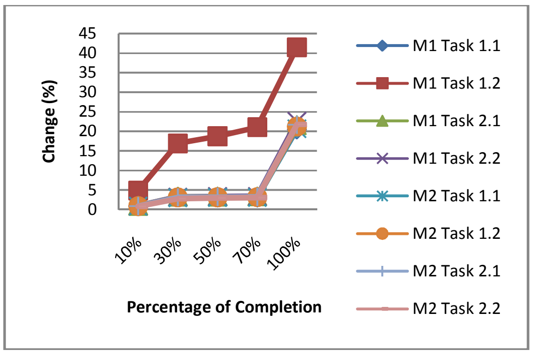
3 Evaluation

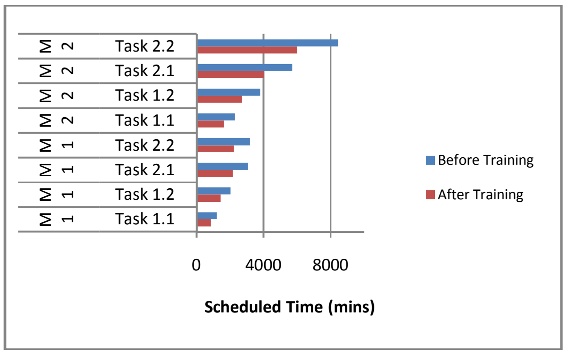
To evaluate the planning accuracy, we create two sample modules “M1” and “M2” of difficulty 0 and 2, and with 10 and 20 credit points, respectively. Both modules contain a 100% coursework-only learning unit holding two main tasks with two sub-tasks of increasing complexity each. Moreover, two student profiles with differing learning styles are created. The calendar entry length is 120 minutes, and there is a 30 minutes break between subsequent entries, scheduled between 8:00-10:00 and 12:00-22:00 on the day. Both profiles use the same neural network training configuration with an error threshold of 0.01, 3,000 training iterations, and a boosting factor of 4.

Each profile uses its own neural network instance, and we perform an initial planning for all tasks. Afterwards, we run up to 10 successive cycles. Before each cycle, one of the scheduled appointments in Outlook is manually moved into the past, causing the system to request a feedback. The data output (total task time and training behaviour) is collected for statistical purposes.

We conduct two scenarios per user, one simulating a swift downward trend, i.e. the user completes the task earlier than expected, and one assuming a more realistic, fluctuating trend with no, very small, or negative progress feedback.

The time change $\Delta t = \left|\frac{t_o - t_n}{t_o}\right| \cdot 100$ in percent, with t_o being the old and t_n the new time, is then plotted as shown in **Figure 5**a, here for task 1.2 of module “M1”.

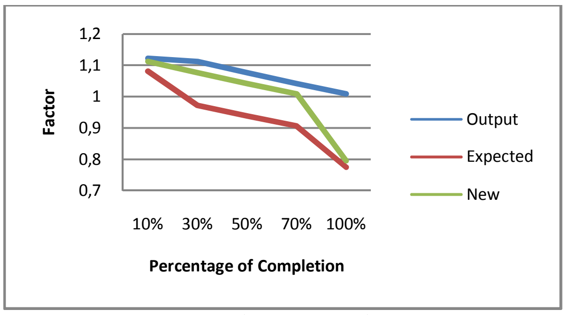


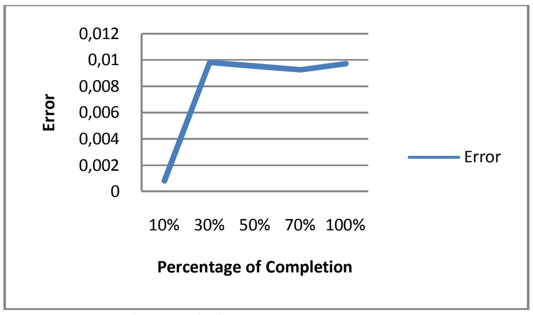


(a) Changes between successive runs (b) Scheduled times before and after training

Figure 5: Scheduling results of the first scenario

The change for task 1.2 is steeper than that of the other tasks since it already includes the reductions caused by its partial completion. However, the system decreases the time for other tasks fairly proportionally. If we repeated the same scenario using the trained neural network, less time would be scheduled for all tasks (cf. **Figure 5**b). Due to the “high” error threshold, the neural network reacts reluctantly to user feedback (cf. **Figure 6**a). The graph (**Figure 6**b) starts off with a low training error, which is rising in the course of the scenario, reaching a maximum in the second run. In the third run, the network already needed six iterations as opposed to just one in previous runs to stay under the threshold.





(a) Output, expected output, and new output (b) Training error

Figure 6: Neural network training results of the first scenario

Since all other scenarios yield similar results, the algorithm copes equally well with differing input vectors. In conclusion, the training error threshold can serve as a factor for the adaptation quality, and the boosting factor controls the adaptation steepness.

The positioning component was tested directly on the mobile device by conducting a series of steps: First, we scheduled a calendar entry C in the near future, associated with a location whose position was known to the system, and set the reminder. We then walked to a position different from the specified one, started the application, and approached the original position while observing the system behaviour. For all tests, a detection interval of 2 minutes was used. We tried different values for the positional tolerance Δp (between 0.01 and 0.02) and the maximum DOP.

We found that a high maximum DOP leads to false reminders although a position was reached, or no reminders at all when it was not reached. Moreover, the weather had a greater influence on the precision than expected. On cloudy days it took up to 3 minutes until valid positions were available. It was also impossible to get fixed positions with an acceptable precision at the inside of buildings and in high-density areas.

4 Conclusions and Future Work

We developed a framework and prototype of an adaptive time management system, making use of sophisticated methodologies presented in contemporary research literature on learning styles (Felder & Spurlin, 2005). The novelty of our approach is its comprehensive character, combining advanced Windows programming, education theory, time management, and machine learning. Furthermore, GPS is utilised to introduce a positional dimension to time management and study organisation.

Our main objective was to provide a tool supporting students in structuring their study activities, assessing their learning capabilities, and planning the resulting learning tasks over time. The system thus deals with a variety of different types of learners, whose learning and time management capabilities change over time. Giving a typical application scenario, a student would first define the structure of their course of study. This includes both organisational and temporal data known to the student, e.g. the modules and dates of lectures etc. The system then uses a wizard-like approach to assess the learning capabilities and to gather data influencing the amount of time to devote to a module. This typically yields data the student is not consciously aware of. After they have identified and entered learning tasks, the time for a module is distributed over its task hierarchy and continually adapted as the resulting schedule is being followed. For this purpose, the system expects feedback from the user and utilises it to re-schedule existing or plan new learning tasks. This iterative process should lead towards a personalised study time planner which can improve the individual learning performance and mitigate existing time management deficits. The use of positioning technology can help disorganised students to optimise their self-management skills and improve the outcome of learning activities. Furthermore, our framework can serve as a foundation for additional tools aimed at enhancing a student's learning experience, e.g. by integrating social network components.

We note akin methods for improving student learning and study planning in (Vavoula & Sharples, 2002), (Brown, Ryu, & Parsons, 2006), (Ab Hamid, Chuan, & Kasirun, 2006), (Kelly & Tagney, 2005), and (Parvez & Blank, 2007), which may contribute to a better use of time, but are not specifically aimed at time management and do not cover all the aspects mentioned above.

Positioning: Due to technical limitations, GPS is not the best option in view of the way it is used in our system. The device cannot react intelligently on positional changes due to the low signal strength inside of buildings and its higher-than-average power consumption. We use a periodical polling strategy, switching the receiver off after each valid position. Consequently, the system is far less flexible with respect to position changes. However, we refer to other useful applications of GPS technology, e.g. guiding students to the location of their next appointment as presented in (Brown, Ryu, & Parsons, 2006). For our purpose, the use of alternative positioning technologies (wireless network or mobile telephone signals) seems more sensible.

System Integration: In its current state, master data must be entered manually into the system. In the next version of the system, we would use a server providing an interface for retrieving such data, and the client would only download a subset dependent on the user's study profile. Other issues are user-friendliness (the user interface is too simplistic) and the integration into the university's computing infrastructure. The confinement to Microsoft technologies is also problematic in view of inhomogeneous environments. Other promising ideas for future versions are social networking and cooperation modules (e.g. for groupwork assignments), a better integration of learning materials, and a component combining training samples of multiple students to produce more accurate planning outputs.

Planning: Our planning algorithm is robust, but not very flexible. It uses a forward planning strategy, scheduling calendar entries at the start of the planning period and leaving leeway at the end. This makes sense in training mode when it is likely that this time will be needed for plan adjustments. In non-training mode we would expect more evenly distributed entries. Furthermore, the current version of the algorithm is very performance-intensive with more than a third of the time spent on Outlook synchronisation. In future versions, the planner could also include a risk management module to further improve the time estimation process.

Adaptation: The use of a neural network for system adaptation is feasible, produces acceptable results, and is relatively easy to implement. Its main drawback is the limited range of output values, so they need to be scaled. We note that there are other approaches such as Bayesian networks (Jensen, 2001) or Fuzzy systems (Kruse, Gebhardt, & Klawonn, 1994), which were not evaluated in this project. However, Fuzzy sets are complex in view of seven input vector coordinates, and a good deal of experimentation is needed to find an appropriate fuzzy set function $\mu(\bar{x})$. Similarly, the use of Bayesian networks requires the definition of a graphical decision model, entailing additional complexity.

We use a simplistic feedback model, consisting only of the overall percentage of task completion. We are not sure whether all users are able to estimate this value accurately and if the adaptation mechanism compensates such inaccuracies. In future versions, feedback could be provided from within Outlook – possibly by using another feedback mechanism – or directly on the mobile device.

System Evaluation: Finally, the system has not yet been evaluated by a representative group of students, which is mainly due to the short time scope of the project. A thorough evaluation means using the system prototype over an adequate period of time. Moreover, it requires specific hardware (PDA) and software (Windows environment).

References

Ab Hamid, H. S., Chuan, T. H., & Kasirun, Z. M. (2006). L3OP: Learning Styles Application Using Learing Objects Approach. *The Sixth IEEE Conference on Computer and Information Technology, 2006. CIT '06*, (pp. 255-255).

Adair, J. (1982). *Effective Time Management.* Pan Books.

Broad, J. (2006). Interpretations of Independent Learning in Further Education. *Journal of Further and Higher Education , 30* (2), 119-143.

Brown, R., Ryu, H., & Parsons, D. (2006). Mobile Helper for University Students. *OZCHI '06: Proceedings of the 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on computer-human interaction, design, activities, artefacts and environments* (pp. 297-300). New York, NY, USA: ACM Press.

Cottrell, S. (2003). *The Study Skills Handbook* (2nd Edition ed.). Palgrave McMillan.

Drew, S., & Bingman, R. (2001). *The Student Skills Guide* (2nd Edition ed.). Gower.

Felder, R. M., & Silverman, L. K. (1988). Learning and Teaching Styles in Engineering Education. *Engineering Education , 78* (7), 674-681.

Felder, R. M., & Spurlin, J. (2005). Applications, Reliability and Validity of the Index of Learning Styles. *International Journal of Engineering Education , 21* (1), 103-112.

Fleurey, F. (2002, April). *CSharp Neural Network Library.* Retrieved September 2007, from http://franck.fleurey.free.fr/NeuralNetwork/index.htm

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs.* Springer.

Kelly, D., & Tagney, B. (2005). 'First Aid for You': Getting to Know your Learning Style Using Machine Learning. *Fifth IEEE International Conference on Advanced Learning Technologies, 2005. ICALT 2005*, (pp. 1-3).

Kolb, D. A. (1984). *Experiential Learning: Experience as the Source of Learning and Development.* Prentice-Hall.

Kruse, R., Gebhardt, J., & Klawonn, F. (1994). *Foundations of Fuzzy Systems.* John Wiley & Sons.

Leung, E. W., & Li, Q. (2003). A Dynamic Conceptual Network Mechanism for Personalized Study Plan Generation. In W. Zhou, P. Nicholson, B. Corbitt, & J. Fong (Ed.), *ICWL. 2783*, pp. 69-80. Springer.

Main, A. (1980). *Encouraging Effective Learning.* Scottish Academic Press.

Marshall, L. A., & Rowland, F. (1998). *A Guide to Learning Independently* (3rd Edition ed.). Longman.

Moore, M. G. (1973). Towards a Theory of Independent Learning and Teaching. *The Journal of Higher Education , 44* (9), 661-679.

Nilsson, N. J. (1998). *Artificial Intelligence: A new Synthesis.* Morgan Kaufmann Publishers.

Parvez, S. M., & Blank, G. D. (2007). A Pedagogical Framework to Integrate Learning Style into Intelligent Tutoring Systems. *J. Comput. Small Coll. , 22* (3), 183-189.

Payne, E., & Whittaker, L. (2000). *Developing Essential Study Skills.* Pearson Education.

Prosser, M., & Trigwell, K. (1999). *Understanding Learning and Teaching: The Experience in Higher Education.* The Society for Research into Higher Education & Open University Press.

Richardson, J. T. (1994). Using Questionnaires to Evaluate Student Learning: Some Health Warnings. In G. Gibbs (Ed.), *Improving Student Learning: Theory and Practice* (pp. 73-88). The Oxford Centre for Staff Development.

Saunders, D. (Ed.). (1994). *The Complete Student Handbook.* Blackwell.

Shalloway, A., & Trott, J. R. (2005). *Design Patterns Explained: A New Perspective on Object-Oriented Design* (2nd Edition ed.). Addison-Wesley.

Soloman, B. A., & Felder, R. M. (2001). *Index of Learning Styles Questionnaire*. Retrieved September 2007, from http://www.engr.ncsu.edu/learningstyles/ilsweb.html

Vavoula, G. N., & Sharples, M. (2002). KLeOS: A Personal, Mobile, Knowledge and Learning Organisation System. *Proceedings of the 2002 IEEE International Workshop on Wireless and Mobile Technologies in Education*, (pp. 152-156).

Zywno, M. S. (2003). A Contribution to Validation of Score Meaning for Felder-Soloman's Index of Learning Styles. *Proceedings of the 2003 American Society for Engineering Education Annual Conference & Exposition.*