De Bra, P., Aerts, A., Smits, D., Stash, N.,
AHA! Version 2.0, More Adaptation Flexibility for Authors. Proceedings of the AACE ELearn'2002 conference,
October 2002, pp. 240-246.

AHA! Версия 2.0
Более гибкая реализация адаптации для авторов (курсов)

Paul De Bra*, Ad Aerts, David Smits, Natalia Stash
Department of Computing Science Eindhoven University of Technology (TUE) PO Box 513, Eindhoven, The Netherlands
debra@win.tue.nl

Аннотация: AHA! – это простая адаптивная гипермедиа система для Web. Из-за своей простоты она изучалась и использовалась в практике несколькими исследовательскими группами, см. например (Cini & Valdeni de Lima, 2002), (Calvi & Cristea, 2002), (Romero et al., 2002). Эта статья показывает недостатки в AHA! и представляет AHA! версии 2.0 которая пытается преодолеть известные проблемы AHA! продолжая поддерживать ее наиболее ценное качество: простату. Мы иллюстрируем как различное моделирование пользователя и аспекты адаптации могут быть выражены в новой системе AHA! (что трудно было сделать в старой системе, и невозможно в большинстве других систем).

Вступление, предпосылки

Разработка системы AHA! началась в 1996 году, когда создали серверное программное обеспечение (CGI-скрипты на то время) чтобы добавить адаптацию в on-line курс по теме гипермедиа. (Для введения в область адаптивной гипермедиа см. (Brusilovsky, 2001)). Программное обеспечение AHA! изменялось и расширялось с течением времени, в конечном итоге став тем, что мы называем AHA! Версия 1.0 в этой статье, и что описано, например, в (De Bra et al., 2000) и (De Bra & Ruiter, 2001). AHA! 1.0 основана на простой архитектуре адаптивных Web систем:

- Приложение состоит из некоторого количества *концептов (concept)*. Некоторые концепты представляют Web страницы, в то время как другие представляют высокоуровневые или абстрактные сущности.
- Модель пользователя (для каждого пользователя) состоит из *значений (value)* для каждого *концепта*. Наиболее общая интерпретация этих значений заключается в том, что значение определяет *уровень знаний* пользователем этого концепта.
- Каждый раз, когда пользователь посещает страницу, *значение* соответствующее этой странице *увеличивается*. В AHA! Автор может определить правила распространения, вызванные этим увеличением, влияющие на другие концепты. Поэтому легко определить структуру страниц, секций и глав, означающую, что чтение страницы вносит вклад в знания по секции и увеличение знаний по секции означает (небольшое) увеличение знаний по главе. Распространяемые изменения произвольны. Также возможно регистрировать *уменьшение* знаний концепта, если пользователь посетил определенную страницу.
- Автор может определить *требования* (*requirements)* для фрагмента страницы и для страницы в целом. Требования являются Булевскими выражениями над значениями концептов. Когда требования для фрагмента страницы принимают значение *истина*, тогда фрагмент включается в презентацию. (В ином случае фрагмент исключается.) Когда требования для страницы выполнены все указатели (anchor) на ссылки на эту страницу показываются явным видимым цветом (обычно голубым или пурпурным в AHA!). В противном случае, указатели показываются *черным*, что означает что они будут спрятаны в тексте, который также является черным. (Черные ссылки также *не подчеркнуты*, чтобы

* Также работает в “Centrum voor Wiskunde en Informatica” в Амстердаме, и университете Антверпена, Бельгия.

сделать их скрытыми более эффективно.)

Как указывало несколько исследователей, включая (Calvi & Cristea, 2002), следствием простоты AHA! является то, что просто создавать только простые, прямолинейные (straightforward) приложения. В этом смысле AHA! по существу не очень отличается от других адаптивных систем, как Interbook (Brusilovsky et al, 1998), которые делают простым создание одного типа приложений, называемых on-line курсами. В (De Bra et al., 2000) мы показали, что *возможно* разрабатывать различные типы приложений в AHA! или основывать адаптацию не только на знаниях, а также на иных аспектах поведения или просмотра пользователем страниц. Тем не менее, примеры которые были предоставлены показали, что эти разработки трудны и было абсолютно очевидно, что моделирование пользователя и особенности адаптации AHA! "тяготили", когда требовалось реализовать альтернативные приложения.

В AHA! Версии 2.0 мы направили силы на решение этой проблемы для чего создали более богатую структуру модели пользователя, предметной области и модели адаптации с более развитыми *требованиями* и правилами *генерации*. Новые особенности основаны на ссылочной (reference) модели AHAM (De Bra et al., 1999), которая является расширением хорошо известной ссылочной модели Dexter для гипермедиа (Halasz & Schwartz, 1994).

Модель предметной области / Модель адаптации в AHA! Версии 2.0

AHA! отличается от ссылочной модели AHAM (De Bra et al., 1999) тем, что в ней не отделена *модель предметной области* от *модели адаптации*. В AHA! автор определяет *концепты*, наряду с *требованиями*, которые определяют условия, когда пользователь "готов" перейти к концепту, и *правилами генерации*, которые определяют как поведение просмотра (browsing behavior) пользователя интерпретируется при обновлении модели пользователя. *Правила генерации* являются *продукционными правилами (condition-action rules)*, как принято в теории активных баз данных (Baralis & Widom, 2000). Мы иллюстрируем структуру концептов AHA! продолжая пример о шоколаде и пиве из (De Bra & Ruiter, 2001). AHA! 2.0 поддерживает хранение структуры концептов в XML файлах или в базе данных mySQL. В данной статье мы используем XML представление. (Есть несколько незначительных отличий между синтаксисом в этой статье и действительным синтаксисом принятым в AHA! 2.0. Это сделано для улучшения читаемости и простоты изложения.)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE conceptList SYSTEM 'concept.dtd'>
<conceptList>
<name>Belgian Beer Example</name>
<concept>
   <name>de_koninck</name>
   <desc>first author's favorite beer</desc>
   <resource>de_koninck.xhtml</resource>
   <req>beer.interest>50</req>
   <attribute name="access" type="bool"
      isPersistent="false" isSystem="true"
      isChangeable="false">
      <default>false</default>
      <generateListItem isPropagating="true">
         <req>beer.interest<100</req>
         <trueActions>
            <action>
               <concept>de_koninck</concept>
               <attribute>knowledge</attribute>
               <expr>100</expr>
            </action>
         </trueActions>
         <falseActions>
            <action>
               <concept>de_konink</concept>
               <attribute>knowledge</attribute>
               <expr>35</expr>
            </action>
         </falseActions>
      </generateListItem>
   </attribute>
   <attribute name="knowledge" type="bool"
```

```
        <action>
            <concept>beer</concept>
            <attribute>interest</attribute>
            <expr>beer.interest+10</expr>
        </action>
    </trueActions>
</generateListItem>
<generateListItem isPropagating="true">
    <req>chocolate.interest<50 and
        chocolate.interest>4</req>
    <trueActions>
        <action>
            <concept>chocolate</concept>
            <attribute>interest</attribute>
            <expr>chocolate.interest–5</expr>
        </action>
    </trueActions>
</generateListItem>
<generateListItem isPropagating="true">
    <req>beer.interest>50</req>
    <trueActions>
```

```
isPersistent="true" isSystem="false"
isChangeable="true">
<default>0</default>
<generateListItem isPropagating="true">
    <req>true</req>
    <trueActions>
        <action>
            <concept>beer</concept>
            <attribute>knowledge</attribute>
            <expr>beer.knowledge+0.2*
                _de_koninck.knowledge</expr>
        </action>
    </trueActions>
</generateListItem>
</attribute>
</concept>
<concept>
    ...next concept definition here...
</concept>
    ...more concept definitions...
</conceptList>
```

Вышеприведенное определение концепта связано со страницей "de_koninck.xhtml". Эта страница рассматривается системой как "желательная", если требование "beer.interest>50" (пиво.интерес>50) выполнено. Для адаптации это означает, что ссылки на страницу de_koninck будут иметь голубой или пурпурный цвет указателя только для пользователей с высоким интересом к пиву. (Ссылка будет голубой до того как пользователь первый раз посетит ее и пурпурной после.) Для незаинтересованных пользователей будет не просто увидеть ссылку на определенное пиво такое, как "De Koninck" потому что указатель на ссылку спрятан (сделан черным, как и обычный текст, и не подчеркнут). Когда пользователь получает доступ к странице de_koninck атрибут доступа *временно* становится истиной. Атрибут – это "системный" атрибут, что значит, что его значение изменяется системными событиями, в данном случае доступом к странице. Атрибут не является неизменным, т.е. его значение возвращается к исходному после того, как событие состоящее в доступе к странице закончилось. Событие вызывает на выполнение три действия (или "generateListItem"s) ассоциированных с этим атрибутом. Первое действие увеличивает интерес к пиву, но только если этот интерес уже не находится на уровне 100. Второе действие уменьшает интерес к шоколаду, но только если этот интерес уже достаточно высок (достаточно высок, чтобы не стать отрицательным после уменьшения). Третье действие, регистрирует, что для заинтересованных пользователей знание о de_koninck стало 100, а для незаинтересованных пользователей (которые случайно нашли эту страницу несмотря на спрятанную ссылку) знание становится только 35. "Знание" является постоянным атрибутом de_koninck, что значит, что оно хранится в модели пользователя. Первоначально атрибут имеет значение 0 ("значение по умолчанию "). Правило ассоциированное с атрибутом знания означает, что любые изменения этого знания также воздействуют на знания о концепте более высокого уровня, как "пиво". 20% *изменения в знаниях* о de_koninck переносится на "пиво". "_" применяется чтобы показать, что используется *изменение* в значении атрибута, а не само *значение* атрибута. Как можно видеть из примера, *доступ* к странице de_koninck изменяет *интерес* к пиву и шоколаду и увеличивает *знание* о de_koninck; это знание меняет пусковые схемы правил атрибута знания de_koninck, которое затем увеличивает *знание* о пиве. Изменение интереса к

пиву и шоколаду и изменение знаний о пиве также могут запускать правила ассоциированные с этими концептами и атрибутами, но этого нельзя увидеть из данного примера, т.к. не приведена структура концептов пива и шоколада.

Вышеприведенный пример может показаться чрезмерно многословным для выражения нескольких вполне простых правил для обновления модели пользователя. Представленный XML файл демонстрирует только один концепт, тогда как даже маленькое приложение уже содержит 100 или более. К счастью простой интерфейс (Java апплет) прячет этот многословный синтаксис от автора. Также большинство приложений требует только нескольких простых типов правил, таких как, пребывание на странице увеличивает знания, и наличие знаний распространяется через иерархию концептов.

Адаптивные приложения иногда сталкиваются с различными сторонами пользователя, включая *знания*, *интересы*, *цели*, *стили изучения*, и т.п.. Их авторы также могут пожелать осуществлять адаптацию исходя из других условий таких, как различия при просмотре, размер окна, пропускная способность сети, расположение, все, что связано с жизнью, образованием, связями и т. п. человека и опыта работы с Web и т.п.. Большим преимуществом AHA! 2.0 над его предшественниками является то, что каждый аспект может быть представлен через различные концепты или через различные атрибуты концептов. Это делает возможным четко различать аспекты адаптации, которые не имеют отношения друг к другу. Использование атрибутов также делает возможным “комбинировать” связанные аспекты адаптации, такие как *знания о* концепте и *интерес к* тому же самому концепту.

AHA! позволяет каждому концепту иметь различные наборы атрибутов, ассоциированные с ним, и различные продукционные правила. Атрибуты могут быть булевского, целого или строчного типов. Каждое правило может запускать другие правила, как было показано на примере выше. Вообще трудно предсказать как поведет себя машина управления правилами (машина вывода) в целом. Мы уже изучали в прошлом проблемы *завершения* и *слияния (confluence)* (Wu et al., 2001). Тем не менее, мы уверены, что самым простым способом разрешить авторам следить за потенциальными проблемами является наличие атрибута “isPropagating” для каждого действия, который указывает разрешено или нет этому действию запускать другие действия. Также это было предложено в модели AHAM (De Bra et al., 1999). Как только выбранное действие (из назначенных) указывает который атрибут какого концепта обновляется, сразу становится ясно распространение которого именно правила будет происходить. Как и во всех такого рода системах основанных на правилах (Baralis & Widom, 2000) существует опасность, что процесс запуска правил посредством этого механизма распространения не будет закончен. В предыдущей версии AHA! было введено арбитражное ограничивающее условие распространения (De Bra et al, 2000), которое гарантирует завершение процесса. К сожалению, эти ограничения могут стать источником непредвиденных для автора последствий. В AHA! 2.0 это ограничение снято. Правила всегда могут запускать друг друга, устанавливая атрибуты “isPropagating” в значение истина. Более того, автор может более точно контролировать условия по которым правило может быть запущено. В (Wu et al, 2001) мы указали как могут быть установлены потенциальные бесконечные циклы на этапе проектирования.

На рисунке 1 мы показываем два образа экрана с авторскими средствами, написанными на Java, для ввода структуры концептов и правила для обновления модели пользователя, которые используются для создания адаптивной презентации (адаптивного представления информации). Первое изображение показывает, что концепт “de_koninck” имеет атрибут “доступ” и “знания”, и что три действия или generateListItems ассоциированы с этим атрибутом. Интерфейс показывает условия (<req>) каждого правила. Второе изображение показывает как определяется третий generateListItem (список элементов генерации). Пока выражения вводятся (такие как “beer.interest>50”) авторские средства показывают выражения красным цветом до тех пор пока они синтаксически корректны и их атрибуты предварительно определены. Это гарантирует то,

что будет отображена полная структура концептов, атрибутов и generateListItems (элементов генерации).

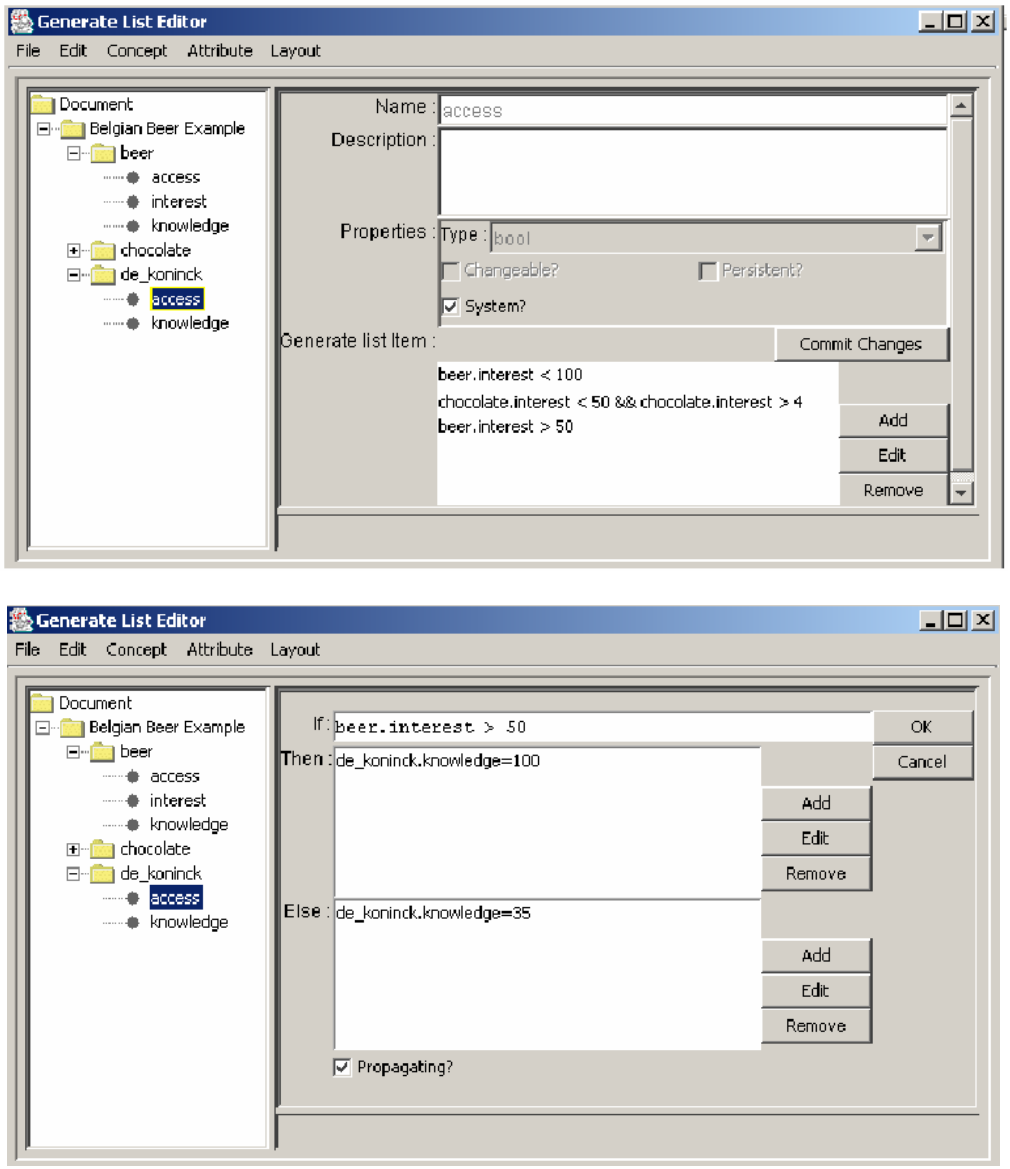

Рис. 1: авторские средства AHA 2.0

Модель пользователя в AHA! Версии 2.0

Наиболее распространенным типом модели пользователя в адаптивных гипермедиа приложениях (Brusilovsky, 2001) является *оверлейная модель*. Каждый концепт и атрибут из модели предметной области приложения содержится в модели пользователя. AHA! 2.0 следует этой же философии, хотя и с двумя исключениями:

- Атрибуты (концептов) в модели предметной области/адаптации могут быть определены не постоянными. Атрибут доступа является одним из типовых атрибутов. Когда запускается механизм адаптации (когда осуществлен доступ к странице) эти переменные атрибуты инициализируются значением по умолчанию (“default”) и любые обновления этих значений теряются, когда механизм завершает работу. Атрибут доступа, например, для

всех концептов первоначально установлен в “ложь”, но он будет временно установлен в “истина” для концепта (страницы), который в данный момент просматривается.

- Когда пользователь входит в систему первый раз создается специальный “концепт”, называемый личный (personal), который используется для хранения аспектов, связанных с пользователем, которые не касаются предметной области приложения. Эта информация включает имя, адрес электронной почты, идентификатор для входа в систему (login-id), пароль и любую другую информацию, которую запрашивает приложение в течение входа в систему. Для нашего on-line курса по гипермедиа эта информация также включает университет, в котором обучается студент, и его специальность. Так как эта информация представлена через концепт, она может использоваться для адаптации. Поэтому, например мы можем предоставлять различные задания студентам из различных университетов. (В AHA! 1.0 информация для входа в систему не была доступна для целей адаптации.)

Ниже мы приводим небольшую часть модели пользователя для воображаемого приложения о пиве.

```
<!DOCTYPE profile SYSTEM 'profile.dtd'>
<profile>
   <record>
      <key>personal.id</key>
      <type>string</type>
      <persistent>true</persistent>
      <value>debra</value>
   </record>
   <record>
      <key>personal.email</key>
      <type>string</type>
      <persistent>true</persistent>
      <value>debra@win.tue.nl</value>
   </record>
   <record>
      <key>personal.goodlink</key>
      <type>string</type>
      <persistent>true</persistent>
      <value>0000ff</value>
   </record>
   <record>
      <key>personal.badlink</key>
      <type>string</type>
      <persistent>true</persistent>
      <value>000000</value>
   </record>
   ...
   <record>
      <key>de_koninck.knowledge</key>
      <type>int</type>
      <persistent>true</persistent>
      <value>35</value>
   </record>
   <record>
      <key>beer.knowledge</key>
      <type>int</type>
      <persistent>true</persistent>
      <value>20</value>
   </record>
   <record>
      <key>beer.interest</key>
      <type>int</type>
      <persistent>true</persistent>
      <value>40</value>
   </record>
   <record>
      <key>chocolate.knowledge</key>
      <type>int</type>
      <persistent>true</persistent>
      <value>15</value>
   </record>
   <record>
      <key>chocolate.interest</key>
      <type>int</type>
      <persistent>true</persistent>
      <value>20</value>
   </record>
   ...
</profile>
```

Форматы документов (страниц)

Приложения AHA! состоят из набора *страниц* со *ссылками* между ними. В настоящее время (в обеих версиях 1.0 и 2.0) основным форматом является HTML, потому что именно его понимают браузеры. В AHA! 2.0 есть три "варианта" в которых может быть записана исходная страница (и которые AHA! транслирует в HTML перед тем как послать их браузеру):

- Есть некоторые XML теги, которые имеют определенное значение для AHA!. Первый формат AHA! состоит из XML страниц с этими тегами, сгруппированными с HTML. Наиболее важными для AHA! являются теги <if>, которые служат для вставки фрагментов по условиям, и тег <a> для ссылок. На выходе AHA! выдает HTML страницу в которой некоторые фрагменты могут быть удалены и в которой ссылки показываются разными цветами в зависимости от "желательности" той страницы на которую эта ссылка указывает.
- Обычные (plain) HTML документы также могут использоваться в AHA!. В таких документах ссылки адаптированы также как и в первом формате. Хотя HTML документы не получают полного преимущества от возможностей AHA! эта функция является важной для включения внешних документов в адаптивное приложение. (AHA! 2.0 может быть полезным для оперирования страницами доставляемыми с других серверов.)
- Используя моделированные (modularized) DTD (и Xerces XML парзер для их поддержки), страницы теперь также могут быть записаны в формате, который теги AHA! с XHTML. Средства AHA! смотрят только на теги AHA! чтобы выполнять их для обеспечения адаптации. В результате становится достаточно просто расширять AHA! возможностями обрабатывать другие форматы документов используя при этом теги AHA!. (SMIL является одним из форматов, который мы хотели бы попробовать в будущем.)

AHA! не навязывает и не предоставляет никаких специальных форматов страниц. Мы используем AHA! в on-line курсах без HTML фреймов и в других курсах с использованием фреймов, а также с Javascript.

Примеры приложения AHA!

Использование AHA! для моделирования и адаптации к *знаниям* и *интересам* было продемонстрировано в предыдущем разделе. Гибкость в применении AHA! становится несомненной, если рассмотреть следующие две небольшие иллюстрации совершенно разного применения адаптации:

- В одном on-line курсе мы хотели иметь два способа представления информации: лаконичный (как схемы) и многословный (полный текст курса). Давая возможность пользователю установить словесно наполненный уровень (что может быть реализовано как атрибут в "личном" концепте) мы можем организовать систему выбора фрагментов для показа и сокрытия. В настоящее время мы предоставляем только два уровня (краткий и многословный) в этих курсах, но используя целочисленные атрибуты мы можем предоставлять больше уровней, показывая все больше и больше подробностей и давая все более длинные объяснения.
- Наш курс, использующий фреймы, имеет "панель навигации" в левом фрейме и отображает страницы в (большем) правом фрейме. Панель навигации отображает темы курса и может по выбору показывать подменю для каждой темы. При показе страницы из определенной темы соответствующее меню будет открыто, а другие меню закрыты. Мы реализовали данную функцию заключив все меню в условные операторы. Каждый раз, когда страница просматривается активируется соответствующее меню. Активация осуществляется посредствам программы на Javascript, которая говорит браузеру, что

необходимо перезагрузить панель навигации. Существует два способа реализации системы меню в AHA!. Первый способ это создать концепт “меню” с булевым атрибутом для каждого меню. Каждый доступ к странице запускает набор правил, которые раскрывают необходимое(ые) меню и закрывают остальные. При этой схеме возможно оставить открытым более одного меню. Недостатком этого решения является то, что когда добавляется новая тема в курс необходимо добавить большое количество правил для закрытия меню этой темы каждый раз, когда открывается любая страница из любой другой темы. Второй способ заключается в создании целочисленного или строкового атрибута меню (или для личного концепта или концепта меню). Каждому меню ставится в соответствие число или имя, и каждая страница устанавливает атрибут меню в то, значение, которое соответствует тому меню, которое должно быть раскрыто. Данное решение не ведет к проблемам при добавлении новых тем. Тем не менее, недостатком этого решения является то, что в один момент может быть открыто только одно меню.

Возможности адаптации такие, как эти две, уже могли быть выражены в AHA! 1.0, но это требовало применения уловок к концептам и правилам адаптации. В AHA! 2.0 можно определить концепты и атрибуты для вполне определенной цели и ассоциированные правила не смогут быть просто “спутаны” с концептами и атрибутами, которые используются для представления знаний и заинтересованности. Мы рассматриваем способность системы четко различать различные подчасти модели предметной области/адаптации, каждая из которых имеет различные цели, как самое большое преимущество нашей новой версии AHA!.

Выводы и планы на будущее

AHA! 2.0 предоставляет универсальный механизм моделирования пользователя и средства адаптации. Мы надеемся, что AHA! будет удобной для многих различных типов приложений. Наш опыт использования AHA! для on-line курсов показал, что помимо моделирования знаний и адаптации предоставляемой на этой основе, AHA! может также использоваться для создания адаптивных приложений на основе личностных аспектов пользователя, включая его предпочтения (такие как цвета, или полный текст против сжатого текста).

Формат презентации в AHA! полностью зависит от автора. Структура с концептами и атрибутами также может быть использована для помощи в создании презентации. Мы, например, показали как создать навигационный фрейм с меню и подменю.

Пока мы еще приобретаем опыт использования AHA! 2.0, но мы уже имеем список пожеланий для будущей версии. Связывание в AHA! осуществляется постранично. Мы планируем в будущем позволить ссылаться на концепты более высокого уровня, наряду с автоматической выборкой лучших страниц для показа. Мы также планируем позволить более гибкое конструирование страниц. Сейчас страница состоит из последовательности фрагментов, которые условно включены только в установленном порядке. Для приложений поиска информации мы, как минимум, должны добавить сортировку фрагментов.

Благодарности

Разработка AHA! Стала возможной благодаря гранту NLnet Foundation. Кроме него мы также хотим высказать слова благодарности многим ученым, которые изучали наши предыдущие исследования и разработки по AHA! И некоторые из них даже создавали приложения с помощью AHA!. Их исследовательские и практические отчеты были для нас очень значимыми, когда решалось, что следует расширить и улучшить в AHA!, чтобы больше людей смогло использовать

ее при разработке своих адаптивных курсов, а также и других адаптивных приложений.

Ссылки

Baralis, E., Widom, J. (2000). *An algebraic approach to static analysis of active database rules*. ACM Transactions on Database Systems, 20:1, pp. 269-332.

Brusilovsky, P. (2001). *Adaptive hypermedia*. User Modeling and User Adapted Interaction, 11 (1/2) pp. 87-110.

Brusilovsky, P., Eklund, J., Schwarz, E. (1998). *Web-based Education for All: A Tool for Developing Adaptive Courseware*. Computer Networks and ISDN Systems (Seventh International World Wide Web Conference), 30, 1-7, 291-300.

Calvi, L., Cristea, A. (2002). *Towards Generic Adaptive Systems: Analysis of a Case Study*. Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Springer Verlag, LNCS 2347, pp. 77-87.

Cini, A., Valdeni de Lima, J. (2002). *Adaptivity Conditions Evaluation for the User of Hypermedia Presentations Built with AHA!*. Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Springer Verlag, LNCS 2347, pp. 490-493.

De Bra, P., Brusilovsky, P., Houben, G.J. (1999). *Adaptive Hypermedia: From Systems to Framework*. ACM Computing Surveys 31:4. (URL: http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/25.html)

De Bra, P. & Calvi, L. (1998). *AHA: a Generic Adaptive Hypermedia System*. 2nd Workshop on Adaptive Hypertext and Hypermedia, pp. 1-10. (URL: http://wwwis.win.tue.nl/ah98/DeBra.html)

De Bra, P., Aerts, A., Houben, G.J., Wu, H. (2000). *Making General-Purpose Adaptive Hypermedia Work*. Proceedings of the AACE WebNet 2000 Conference, pp. 117-123.

De Bra, P., Ruiter, J.P. (2001). *AHA! Adaptive Hypermedia for All.* Proceedings of the AACE WebNet Conference, pp. 262-268.

De Bra, P., Houben, G.J. & Wu, H. (1999). *AHAM, A Dexter-based Reference Model for Adaptive Hypermedia*, Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 147-156.

Halasz, F., Schwartz, M. (1994). *The Dexter Hypertext Reference Model*. Communications of the ACM, 37:2, pp. 30-39.

Romero, C., De Bra, P., Ventura, S., de Castro, C. (2002). *Using Knowledge Levels with AHA! for Discovering Interesting Relationships*. Proceedings of the AACE ELearn'2002 Conference.

Wu, H., De Kort, E., De Bra, P. (2001). *Design Issues for General-Purpose Adaptive Hypermedia Systems*. Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 141-150.