Declarative To Procedural Manual
Jeremiah W. Hiam, Frank E. Ritter, Jonathan H. Morgan, Christopher L. Dancy II
jwhiam@psu.edu, frank.ritter@psu.edu, jonathan.h.morgan@gmail.com, cld5070@psu.edu
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
17 October 2013

Abstract
This documents how to create tutors using the Declarative to Procedural (D2P) tutoring architecture. Tutors built in this architecture are designed to move declarative knowledge into procedural knowledge through practice in applied training systems. There are two example tutors so far, the Moving Target Tutor (D2P/MTT) and the Combat Lifesaver Tutor (D2P/CLS). This manual describes how to get started, how to create a simple tutor, the types of pages, the possible contents of pages, and how to adjust the automatic/adaptive tutoring algorithms included.

Acknowledgements
Tatum Software (Ian Schenk and Jon Staszeiasc) created the first version of D2P in collaboration with us. ONR grants have supported creating the Moving Target Tutor (N00014-11-1-0275), the Combat Lifesaver Tutor (N091-086/P10008), and D2P (N00014-11-M-0342). Mark Cohen created the instructional designer’s tool set. Martin Yeh, and colleagues at Charles River have provided useful comments. Joseph Sanford contributed to source code development and provided useful comments as well.
Table of Contents

Abstract .......................................................................................................................... ii
Acknowledgements........................................................................................................ ii
Table of Contents .......................................................................................................... iii

1. Introduction .............................................................................................................. 1

2. Design and Overview of D2P .................................................................................. 1
   2.1 The Use of Java in D2P ...................................................................................... 2
   2.2 Why Java? .......................................................................................................... 3
   2.3 Java GUI ............................................................................................................ 3
   2.4 Record Keeping .................................................................................................. 3
   2.5 How to Begin Creating a Tutor ......................................................................... 3
   2.6 System Requirements to Run D2P .................................................................. 4
   2.7 Control Flow in the Tutor ................................................................................ 5
   2.8 Types of Pages .................................................................................................. 5
   2.6 Media Use in Pages ........................................................................................... 6

3. XML Tags for Creating Pages ................................................................................. 6
   3.1 The D2P Page Editor ........................................................................................ 7
   3.2 Overview of Standard Page Tags ...................................................................... 7
   3.3 Type of Pages (Beans) ...................................................................................... 8
   3.4 Menu Pages ....................................................................................................... 8
      3.4.1 Menu Items ................................................................................................ 8
      3.4.2 Title ............................................................................................................ 9
      3.4.3 Jumps to Index ........................................................................................... 9
      3.4.4 Thumb Path ............................................................................................... 9
      3.4.5 Description ............................................................................................... 9
   3.5 Test Pages .......................................................................................................... 9
      3.5.1 Questions .................................................................................................. 9
      3.5.2 Example Set ............................................................................................... 10
      3.5.3 Timer Delay .............................................................................................. 10
      3.5.4 Filters ....................................................................................................... 10
      3.5.5 Entry Key ................................................................................................ 10
      3.5.6 Value ........................................................................................................ 11
      3.5.7 Media in Test Pages .................................................................................. 11
   3.6 Video Test Pages .................................................................................................. 11
   3.7 Image Test Pages ................................................................................................ 11
   3.8 Results Page ...................................................................................................... 12
   3.9 The Question Builder ......................................................................................... 12
   3.10 Multiple Choice Question Pages ..................................................................... 12
   3.11 Fill in the Blank Questions.............................................................................. 13
   3.12 Declarative Question Tags .............................................................................. 13
      3.12.1 Category .................................................................................................. 13
      3.12.2 Number of Questions .............................................................................. 13
   3.13 Simulation Pages ............................................................................................... 13

4. XML Structures (beans) ......................................................................................... 13
   4.1 Bean Class ........................................................................................................ 14
      4.1.1 Controller ................................................................................................. 14
      4.1.2 Parent Controller ..................................................................................... 14
1. Introduction

Declarative to Procedural (D2P) is a language for creating tutoring systems that teach declarative knowledge and procedural knowledge, and also to then provide the tutee an opportunity to practice this knowledge so that the declarative knowledge is proceduralized\(^1\), thus, Declarative to Procedural or D2P. Through this practice, the declarative knowledge may be proceduralized and applied quickly. A theoretical account of this is available (Kim, Ritter, & Koubek, 2013), and this approach of practice to proceduralize memories is based on previous work and summaries of learning (Anderson, 1982, 2000, 2004; Kim, Ritter, & Koubek, 2013; Ritter & Schooler, 2001).

This document describes how to create a tutor, including the numerous options and definitions available to create a tutor from the D2P tutoring language. These specifications will prove useful for readers interested in providing content to a tutor created in the D2P tutoring language, or for those looking to gain an understanding of the language’s capabilities. Further, it gives an abbreviated relationship between the Java language that serves as the Graphical User Interface (GUI) generator and the XML (Extensible Markup Language) that specifies content including pages, menus, and interactive items within the tutor.

We created D2P initially to teach a perceptual motor skill that needed each of its components practiced by transitioning the learner’s declarative understanding of the steps involved into a procedural knowledge. Procedural knowledge (or very rapid declarative retrieval) is required as the learner has to be able to execute this task in a very limited time frame: approximate 2 to 3 seconds.

From developing this custom complete tutoring system came the understanding that other available tutoring languages were not sufficient either through lacking compatibility, source code distribution, or the ability to be extended, although we are continuing to review these other systems.

The manual provides an overview of the tutoring approach, and then describes how to start to create a tutor including how to create pages and then how to create different pages, how to create and use the logs that can be created, and how to adjust the tutoring apparatus.

2. Design and Overview of D2P

D2P creates a tutoring session by reading in an XML specification of a tutor using a Java parser that then passes the information to an architecture that renders the information and displays it in and as a GUI. This separation between knowledge and architecture was envisioned and initially

\(^1\) When we say *proceduralized*, we mean the knowledge is moved from a *fact* that one must recall to a *condition-action pair(s)* that constitutes a procedure.
created by our early collaborators Tatum Interactive, and was created to ease the process of designing or extending tutor content.

The instructional designer does not need to have familiarity with Java or object oriented programming to create pages, sections, or a tutor’s overall design. Instead, this separation of knowledge and architecture allows subject matter experts and instructional designers to contribute meaningfully with less demand on learning a new language (now through the Page Builder interface) and it supports reuse and automatic instruction. XML was chosen as the implementation language for its versatility and similarities to HTML. The structure of the tutoring approach is shown in Figure 1.

![Diagram of D2P design architecture](image)

**Figure 1. The D2P design architecture.**

### 2.1 The Use of Java in D2P

A Java system is used in D2P to parse and visualize the XML pages written by an instructional designer and present the pages in the form of the tutor. Further, the Java D2P code creates database records as the learner interacts through a connection with an included, portable SQL server and logs user behavior in a plain text format. For more information on logging and the
SQL information stored, refer to the following sections. For information on how logs are able to be sent to a remote server, view the setting up a receiving server topic.

2.2 Why Java?
The Java programming language was chosen to accommodate instructional designers to potentially make use of a D2P tutor in a variety of locations: be it during deployment in a foreign environment, or a student using a tutor on their home computer. Because Java boasts compatibility with a wide range of operating systems and hardware specifications, as well as providing its own virtual machine for code compiling instead of relying upon an outside compiler, it seemed the perfect candidate for a tutoring language with mobility as a core concern.

2.3 Java GUI
The tutor works through a combination of XML and the Java programming language. XML is used by content designers to specify the layout, media, and other content of pages while the engine built in Java remains unmodified by content designers. Instead, the Java system is a means of interpreting the designer’s XML input and displaying the information to learners through a constructed GUI. Where the XML references and indicates media or text, the Java parses and combines the media into the final presentation.

2.4 Record Keeping
The D2P language stores information about the tutor, the learner, and the manner in which the tutor was used in three formats. First, the learner sees most of their information displayed in the tutor through a connection to a local SQL database (on the learner’s machine). This database stores a wide range of information, from basic items such as their login name, to more detailed information such as speed and accuracy on examination questions in an aggregate or detailed manner. Second, D2P retains information through the use of a log file system. This system records the learner’s behavior, progress, and relative score while using the tutor, as well as time spent by section, question asked, and overall time spent. Third, when the learner completes a tutor session, the log file is then transferred via SFTP (secure FTP) to a remote server for data collection and analysis.

2.5 How to Begin Creating a Tutor
To create a tutor, the first step is to do a task analysis of what you are tutoring. This analysis can be done in plain text or graphic editor. In the next release we anticipate supporting using Herbal (Cohen, Ritter, & Haynes, 2010), a high-level behavior representation language to create this task analysis and have it directly import into D2P. Currently, Herbal models can be imported into the tutor, but it is not as easy as we would like.

The task analysis can be done in a hierarchical way, and can be done with reference to the Problem space computational model (Newell, Yost, Laird, Rosenbloom, & Altmann, 1991; Yost & Newell, 1989) or other hierarchical task analysis method. We suggest using the PSCM.
With the task analysis in hand you should prepare the instructional material. We typically wire frame the tutor in Power Point, but we have also used Microsoft Word. The point is to edit the text and prepare figures and pointers to media such as movies using a system that is fluid and familiar. This step can include preparing images, video, and text. D2P/MTT uses all of these types as does the D2P/CLS system.

You might also prepare a simulation. CLS uses a simulation. D2P allows pages to include simulations, and can pass questions into the simulation to ask the user (Ritter et al., accepted).

As you prepare this material, you should check that (a) the pages teach all the tasks in the task analysis. (b) The questions (realized as video questions, text questions, or simulation questions) are often enough and close enough to the task analysis that they give the user practice with the tasks, and that all objects in the task analysis are tested so that the learner has a chance to practice retrieving/executing/proceduralizing them. In the MTT, learners practice judging speed and lead quite often. How often is still specified by the instructional designer. In the CLS tutor, learners practice applying tourniquets frequently.

With all this in hand, then, acquire the D2P architecture as a software system. You can download the D2P software here. Next, you need software for formatting and reading XML page types: a textual XML editor. Examples of such editors include: Eclipse, Emacs, XML Notepad, and the Serna XML editor.

Using the D2P page editor, you may then create your first page, which will appear directly after the login screen in the resulting tutor; you can also download the page-editor using the link above. We recommend you examine the appendix to gather an idea on how the various XML tags and pages are structured, as well as a glimpse at the fundamental properties of varying page types. You can also edit a copy of our stub example (pages.xml.sample) included in the D2P distribution.

You can then create each page that is in your wireframe using the D2P page builder system, and prepare the questions to the learner in the question editor. Cleanup and light editing used to be done in an XML editor, but now these edits can also be done in the D2P page builder.

2.6 System Requirements to Run D2P

For authors of a tutor, check the version of Java you run and make sure you have a reliable XML editor. We developed D2P on Java version 1.6.26, and for compatibility reasons suggest you use at least this version or higher. This Java requirement applies to both development and distribution machines.

While we hope to have future versions of the D2P language Mac compatible, at this time not all D2P tutors work on Mac OS due to an incompatibility with video rendering.
2.7 Control Flow in the Tutor

As described in detail in the XML tags section of this document, the instructional designer is able to determine the order of pages and pace of testing pages. As such, they have a great deal of control over how the learner interacts with testing material. The instructional pages do not have any preset viewing duration limit at this time, and are advanced through the learner’s use of the ‘Next’ button presented in the Java GUI.

We urge tutor authors to carefully evaluate the length of time that pages with questions are given based on the needs of their learner. If too little time is provided to answer a question page, the learner will feel overwhelmed by the rapidity of the questions, and if too much time is offered, the learner will not experience any pressure to proceduralize the knowledge being presented.

2.8 Types of Pages

D2P provides 10 different page types. These are noted in Table 1. These page types serve different purposes in the creation of the tutor: instruction, practice, testing retention or proceduralization, navigation, and reviewing performance over the content of the tutor. The tutor designer may use media as a means of instruction or as part of review and testing. The currently supported image files are .jpeg and .png, while the movie format is .mpg.

<table>
<thead>
<tr>
<th>Page Type</th>
<th>Page Purpose</th>
<th>Notes on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction page</td>
<td>Gives textual information to the learner.</td>
<td>Supports standard tags, HTML text formatting.</td>
</tr>
<tr>
<td>Menu Page</td>
<td>Allow the learner to navigate about the tutor</td>
<td>Supports images, jumps to index, and standard tags.</td>
</tr>
<tr>
<td>Image instruction page</td>
<td>Use text and images to teach the learner</td>
<td>Supports images, HTML text formatting, and standard tags.</td>
</tr>
<tr>
<td>Video instruction page</td>
<td>Use a video and text to teach the learner</td>
<td>Supports videos, HTML text formatting, and standard tags.</td>
</tr>
<tr>
<td>Image test page</td>
<td>Test the knowledge of the learner using an image</td>
<td>Supports images, and standard radio button answer input, displays feedback on input.</td>
</tr>
<tr>
<td>Video test page</td>
<td>Test the knowledge of the learner using a video</td>
<td>Supports videos, radio button input, and keystroke input, displays feedback on input.</td>
</tr>
<tr>
<td>Video pre test page</td>
<td>Gauge the learner’s knowledge before they begin use of the tutor.</td>
<td>Supports videos, radio button input, and keystroke input, records to database/log files distinctly, does not display feedback on input.</td>
</tr>
<tr>
<td>Page Type</td>
<td>Description</td>
<td>Supports Features</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Video post test page</td>
<td>Final test of the learner’s cumulative knowledge after using the tutor.</td>
<td>Supports videos, radio button input, keystroke input, and records to database/log files distinctly.</td>
</tr>
<tr>
<td>Image pre test page</td>
<td>Gauge the learner’s knowledge before they begin use of the tutor.</td>
<td>Supports images, and standard radio button answer input, records to database/log files distinctly.</td>
</tr>
<tr>
<td>Image post test page</td>
<td>Final test of the learner’s cumulative knowledge after using the tutor.</td>
<td>Supports videos, radio button input, keystroke input, and records to database/log files distinctly.</td>
</tr>
<tr>
<td>Results Page</td>
<td>Review and show user progress on current test and over time.</td>
<td>Supports standard tags, content is automatically generated.</td>
</tr>
<tr>
<td>Multiple Choice Question Page</td>
<td>Test the user with a declarative question.</td>
<td>Supports standard tags, question categories specified by IA and then populated through the database.</td>
</tr>
<tr>
<td>Fill in the Blank Question Page</td>
<td>Test the user with a declarative question.</td>
<td>Supports standard tags, question categories specified by IA and then populated through the database.</td>
</tr>
<tr>
<td>Simulation Page</td>
<td>Creates an interactive simulation for the learner to manipulate.</td>
<td>Uses standard tags, and the instructional designer may specify a simulation to use.</td>
</tr>
</tbody>
</table>

### 2.6 Media Use in Pages

The inclusion of media (such as a video or picture) on a page is a part of the page design; one simply places the media in an identifying subdirectory (images or videos), in the same directory as the executable D2P.jar with associated XML file or D2P with an embedded XML file. Following this, when creating any of the appropriate page types (image and video pages for either tests or instruction) the link to the media is established on the XML for that page. With this linking capability described in XML, the author is able to partition and organize their various media resources.

### 3. XML Tags for Creating Pages

Using XML tags, the Instructional designer creates an outline and source material for the D2P engine to display to the learner in the GUI. All pages have a set of required tags that define their basic behaviors, including their view type, their controller, the parent controller, and finally the model. In addition to these standard tags, pages with specific display media or input types have
additional tags that are used to expand and define their functionality. This section of the manual explains the syntax, purpose, and nature of these tags for the Instructional designer.

While the Instructional designer should have access to the D2P Page Editor tool (below), understanding the purpose of the tags used in the D2P architecture will improve the instructional designer’s productivity, debugging, and error handling capabilities.

3.1 The D2P Page Editor
The page editor software is provided to ease the construction of XML pages by providing a GUI-driven tool that semi-automates page creation. Currently, the editor will handle the number of pages automatically to create smooth <jumpsToIndex> tag use in menu pages (for jumping to pages), and provides a list of additional properties per page type that the author may specify the value of. Further, the consistency of a code driven tool for the creation of pages helps prevent syntax errors.

Additionally, the properties file associated with the page editor may be modified by the Instructional designers to help automate and expedite the use of tags as attributed to certain page types.

3.2 Overview of Standard Page Tags
The XML portion of D2P serves as the means in which a tutor page is specified and populated with media and textual information. All page types may include images or video media, as well as the ability to designation sections, titles, and links to other pages. XML shares a formatting that is similar to the HTML language (both are markup languages), using encapsulated tags to indentify components of D2P for the java code to parse and represent. Additionally, you are able to create tests and practice exercises. In practice exercises you describe stimuli and provide the scope of possible answers as well as the correct answers, done as a set. It is possible to give feedback and record performance on individual selections or question sets as the whole, including the time to answer.

This manual will at times reference standard tags. The definition of a standard tag is a required part of the XML structure for all page types. All pages require the following definitions as shown on Table 2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Type</td>
<td>Links to Java class Page</td>
<td><code>&lt;bean class=&quot;com.tatuminteractive.mtt.view.pages.ImageInstructionPageView&quot;&gt;</code></td>
</tr>
<tr>
<td>Controller Type</td>
<td>Links to Java class Controller</td>
<td><code>&lt;bean class=&quot;com.tatuminteractive.mtt.control.PageController&quot;&gt;</code></td>
</tr>
</tbody>
</table>
In short, each page must have the type of page, its controller, and its model set. Though it is not required, we highly recommend that the user also include at least a title tag.

### 3.3 Type of Pages (Beans)

While many pages have an overlapping use of standard, structural XML beans, each page type typically comes with unique or presentation type focused tags that allow a greater degree of control over the realization of the tutor. It is important for authors to familiarize themselves with these beans to fully use the capabilities of the D2P language.

### 3.4 Menu Pages

Menu pages in the D2P language allow the user to navigate to portions of the tutor specified by the instructional designer by clicking upon the appropriate linked icons. Using the jumps to index functionality of the tutoring language, menu pages allow a learner to move from a menu page to another page (properly referenced) by clicking on the icon that represents that new page. Further, short descriptions of sections may be shown in this type of page, and previously visited sections of the tutor may be presented in a different manner to avoid unneeded overlapping (but, this feature remains to be implemented). A full example is included in Appendix A1.

Menu pages are organized in a two column approach, with each row of a column containing a picture, a section title, and a brief, short item description. There is a finite limit to the length of descriptions and headings only in the sense of aesthetics. Larger pictures that are linked as section icons will be automatically scaled down to fit the expected icon size.

Menu pages contain standard tags such as page type, model, controller, title, sub-title. However, they also have unique interactions and unique tags with such items as jumps to index, thumb path, and menu items. The following sections describe the important XML tags involved with the creation of a menu page, and an example menu page may be found in appendix one.

#### 3.4.1 Menu Items

```xml
<property name="menuItems">
  <list>
    
  </list>
</property>
```

This tag precedes the listing of all the potential displayed icons to the tutor learner. Because it is interpreted as a list by the Java side of the tutoring language, the list tag must follow.
3.4.2 Title

An enclosed tag that will display the name of the section the learner may move to by clicking this icon. In this case, next to the image display will be the larger title.

1) First menu item

3.4.3 Jumps to Index

This enclosed tag sets the page number that the learner will move to when clicking this icon. In the case given, the learner will be moved to the second page of the tutor. Tutor page numbers are assigned automatically by the architecture at run time, the first created page of the IDer being 1, and incrementing from there on.

3.4.4 Thumb Path

This enclosed tag informs the tutor of the location of the image to display for this menu item. It is a relative path, meaning that it will start at the directory the tutor is run in, and then move down the linkage path indicated from there. In this case, the tutoring language starts in the current directory the .jar file was run from, then looks for a directory named ‘images’, and then the file named ‘rcoSIGHT.png’ and displays this image. The Instructional designer may reference .png or .jpeg files.

3.4.5 Description

This enclosed tag contains the subtext under the title text, used to further describe the section of the tutor indicated by the image. In this case, next to the image and under the title text, the following would be shown: “Test your existing knowledge about engaging moving targets.”

3.5 Test Pages

Test Pages allow the learner to apply/practice/test their knowledge by answering questions based on presented media. The two types of applicable media currently are images and videos using the jpeg and mpg formats respectively. The answers are processed automatically by the Java side of the tutoring language by reading the name of the indicated media. It would make sense in the future to specify simulation exercises in this way as well.

Questions are answered by the learner by clicking radio buttons containing potential answers, or by pressing a corresponding button on their keyboard associated with the correct value.

3.5.1 Questions

This open tag encapsulates all of the prospective questions to be asked by the tutor on video or image test pages. Finished by using:
Currently, there are four values that can go into a question set: Distance, Speed, Angle, and Points_of_Aim. You can use any combination of these four values in your question set. Below is an example of a set of question values.

<set>
  <value>DISTANCE</value>
  <value>SPEED</value>
  <value>ANGLE</value>
  <value>POINTS_OF_AIM</value>
</set>

The set tag is an XML tag that specifies a group of information. For question pages, this is used to indicate the answer titles. As shown below, it also is used to enter the radio button options for each answer category.

<entry key="DISTANCE">
  <set>
    <value>25</value>
    <value>50</value>
    <value>100</value>
  </set>
</entry>

### 3.5.3 Timer Delay

This optional enclosed integer value tag sets the length of time in milliseconds the learner has to answer the questions presented on this page.

Authors should be careful when setting this value, as too little time may not give the learner sufficient ability to interact with the Java GUI. If this tag is not used, D2P puts a default time of thirty seconds (30,000 milliseconds).

### 3.5.4 Filters

This open tag is used to offer the possible answers to the information presented in the testing media. Each specific answer is set with the 'value' tag. This tag is required for classification page types, without at least one, there will be no answer option to questions.

### 3.5.5 Entry Key

This enclosed tag sets the group name of the possible answers. This will be displayed above the grouping of radio button inputs described by the value tag. This tag is required for classification page types, without at least one value given, there will be nothing for the learner to classify their responses.
3.5.6 Value

<value>25</value>

This enclosed tag gives a possible answer, the value of which is placed beside the corresponding radio button presented in the Java GUI. This tag is required, as without at least one value there will be no options for the learner to answer with.

3.5.7 Media in Test Pages

Any media used by the test page (e.g., image for an image test page) must be named a certain way to be correctly recognized by the page; **incorrect naming can result in errors when the tutor is run by the D2P architecture.** The required prefix for the image and video media is `DISTANCE_SPEED_ANGLE_POINTS-OF-AIM` with (distance, speed, etc.) corresponding to the values from the question set.

3.6 Video Test Pages

Video test pages are tutor pages that test the learner’s knowledge gained in the tutor by presenting a video to them, and then having them provide a response about that media as it plays. Like all test pages, the length of time to respond and questions are described and set by the tutor author.

It uses the standard tags in Table 2. There is one additional tag that this page type can use:

**Number of Videos**

<property name="numberOfVideos" value="10" />

This enclosed tag allows the instructional designer to set the total number of video questions that will be asked on this page. If no number of videos is given, the D2P language will use its default value of 10.

3.7 Image Test Pages

Image Test pages show the user an image to a specified number of images from which a question may be asked of the user, and answer categories and options are then presented, as specified by the instructional designer. The default number of pictures the architecture will attempt to show is 10, but will not repeat an image on the same page, and will not cause an out of bounds error to generate by only showing the total number of pictures in a directory if more than that are specified by the instructional designer to be displayed.

Tags in these pages allow the instructional designer to specify the number of pictures, the sub directory where these pictures are located, the answer categories(using filter, map) and answer radio button options (using set).

It uses the standard tags in Table 2. There is one additional tag that this page type can use:
Number of Pictures

<property name="numberOfImages" value="2" />

This tag informs the architecture of how many image tests to present to the learner. If the tag is omitted, the default number is 10.

Images for these test pages must be in the `images/exerciseimages` directory to be used by these pages. Including an Image Test (or Pretest) page without this folder and images named as indicated in section 3.5.7 will result in an error within the D2P architecture.

3.8 Results Page

Results pages allow the learner to observe their progress, both on immediate tests completed and their progress over time. It is recommended these pages be shown to the learner after tests have been taken on a test page.

By placing a Results page directly after a Pretest page (e.g., Video Pretest), one can allow the user to review their scores directly after taking a pretest. If the Results page is placed anywhere but directly after pretest pages, the user will see general test information from the previous tutoring sessions with the most recent tutoring session appearing first.

3.9 The Question Builder

The Question Builder is a Java based tool used by instructional designer to populate questions for later use into the architecture. To do so, one must run the questionbuilder.jar file, select an appropriate database when prompted, and then begin filling in questions by category and type. When a category is requested in a later question page, any questions belonging to that category and type of question may be asked of the learner.

The method for selecting which question is displayed is based upon the instructional designer’s selection of a category. Categories of questions belong to Question Sets, which represent the largest block of information categorization used by the question pages and tutor structure.

As an example, an instructional designer may create a Question Set titled “Fruit” that contains all questions he will ask the learner about fruit. However, the Fruit Question Set may contain several categories, such as strawberries, grapes, or tomatoes for each specific topic of interest under that Question Set. When creating a question page, that page will pull up one of X questions from the category, where X is the size of all questions in that category.

3.10 Multiple Choice Question Pages

Multiple Choice pages allow the instructional designer to test the learner’s knowledge by presenting them with a series of questions from the database supporting D2P. The learner reads the question, and then selects the best answer of up to four possible choices.
3.11 Fill in the Blank Questions
Fill in the blank question pages function much as the multiple choice question pages do. The difference is in the nature of how the learner answers. Instead of being prompted with four possible choices, the learner instead must type in what they believe is the correct answer. At this time, D2P will accept any answer that matches the one in the database, and is not case sensitive.

3.12 Declarative Question Tags
In this section the distinct tags to declarative question type pages are shown.

3.12.1 Category
<property name="Category" value="SomeCategory" />
The category tag is the means by which the Instructional Designer tells the architecture which group of questions to use on a particular given question page. The category of questions is set by the Instructional designer when using the D2P Question Builder.

3.12.2 Number of Questions
<property name="NumberOfQuestions" value="1" />
This Tag determines how many questions will be asked of the user when visiting the given question page. If this tag is omitted, the default is 10. The architecture will not display more questions than are in a category, however, asking for more than exist will not cause the tutor to error.

3.13 Simulation Pages
Simulation pages present the learner with an interactive environment and display, allowing them to manipulate objects or persons as they practice skills taught by the tutor through the application of them.

The properties Tag allows an instructional designer to set the scenario they wish to have the page display to the user. The list of acceptable scenarios is also generated by the instructional designer through use of the simulation’s extensibility options.

4. XML Structures (beans)
The basic way to create D2P tutors is by creating pages. Pages are represented with XML. These pages include D2P’s core functionality lies in the creation of pages through XML. The types of pages include an introductory Page, Instruction on how to use the tutor, to the core tutor pages, of the tutor including the instructional material itself.

Learning the format and structure of these XML tags and Java beans will empower the Instructional designer to fashion a tutoring system, and help use a tool for building pages called the D2P Page Builder.
4.1 Bean Class
The various tags that are presented here are Java beans, read in by the architecture and applied to the system in an appropriate manner to tune the output in desired ways. Through these tags, we are able to extend the Instructional designer greater flexibility in their design process, modify recording of learner information, and flag various other options.

The following section lists all used XML tags/Java Beans employed by the D2P language. Additionally each tag is explained and an example of that tag properly employed is demonstrated.

4.1.1 Controller
   <property name="controller">
   <bean class="com.tatuminteractive.mtt.control.PageController">
This tag designates the Java class responsible for ‘controlling’ the page that will be displayed. Page controllers contain functionality that allows the tutor to move pages, login, logout, record learner input to the local database and local text log files. The controllers have specific functions tied to the information type presented, so linking the correct controller is essential.

4.1.2 Parent Controller
   <property name="controller">
   <property name="parentController" ref="mttMain" />
This tag simply tells the D2P what class should be considered the program’s beginning. For all intents and purposes, it will always remain as shown here; there is no need to alter it.

4.1.3 Page Model
   <property name="model">
   <bean class="com.tatuminteractive.mtt.model.ImageInstructionPageModel" lazy-init="default" autowire="default" dependency-check="default">
Models act as the template for a page to control information management and data presentation for the page. There is one model per type. In the future these could be merged into having a single page type and model type per page. The page builder currently provides this linkage to instructional designers.

4.1.4 Title
The title tag is used to create a bold heading at the top of any tutor page. There are no restrictions on what a title may be, however, it is advised that for organizational purposes, the author keeps a consistent methodology for organization and presentation of information. Figure 2 shows how a title and subtitle appear when the tutor is running.
4.1.5 **Subtitle**
Subtitles appear beneath the title and are used for clarification on page purpose. They are optional; there is no default value.

4.1.6 **Value**
Value is a generic tag that assigns a value to a listed tag type before. What the value of this tag should be is dependent on what it expands.

An example of a value tag expecting a path to media:

```xml
<property name="imagePath" value="images/rcoSight.png"/>
```

An example of a value tag expecting text input:

```xml
<property name="title" value="Introduction to the Moving Target Tutor (MTT)"/>
```

4.1.7 **Content**
```xml
<property name="content">
  <![[CDATA[ <html> Text and HTML tags to present to the user.
  </html> ]]>}
</value>
</property>
```

This tag allows the user to add textual content to the page they are creating, using standard HTML tags to format the text. The `<!CDATA` tag acts as an escape character, causing XML not to parse the text in the brackets following normally. The `<html>` tag allows the instructional designer to present the page format using HTML within the escape tags.

Notably, this tag applies to menu pages and instruction pages, but is not meant for use with classification (testing) pages. An example is available in appendix A2.

4.2 **Media Inclusions**
This section explains the use of specific tags related to the presentation of media to the learner. These tags are only applicable to image and video type pages.

4.2.1 **Image Path**
```xml
<property name="imagePath" value="images/good_picture.jpg"/>
```
This tag is used in image instruction pages, setting the relative path of the picture to be displayed. Relative path means that the D2P language will begin its search for the media at the directory it was run from. So if it was run from the c:/D2P folder, it would look in the absolute path of c:/D2P/Images/good_picture.jpg.

### 4.2.2 Relative Video Path

<property name="relativeVideoPath" value="videos" />

In using this tag, the Instructional designer tells the D2P language where to pull out a subset of the media to use for testing in video classification pages. If we set the value to “videos”, D2P will use relative path as described in the previous tag explanation to look for the directory /videos under where it was initiated from.

### 4.2.3 D2P Default behavior for Playing Images and Videos

When selecting a video or image for a test page, the D2P architecture will attempt to pull one of the media files pseudo-randomly from the subset of media in the indicated media path. If multiple media displays are requested (NumberOfImages, NumberOfVideos tags) D2P will pull up to the amount requested, or the total subset of the media available in the designated path, whichever is smaller. Additionally, the media pages implement a shuffling algorithm that attempts to randomize the display of media to the user for each page. In this way, there is not a predictable sampling of media for the learner to grow accustomed to. What is chosen for display is put into the log for later analyses.

## 5. Care and Feeding of Logs

Log files are instantiated on the local machine running the D2P language and are expanded with each use session. At the termination of a session, the D2P tutor will attempt to send the local log file to a remote repository that the author may indicate in the tutor specification (see below).

The population of information in the log takes place over real time, handled by the Java portion of the D2P language as the learner interacts with the tutoring software. Further, on the local machine, the log file is always identified by the name “mtt-trace.log”. When sent to a remote server, the file is changed to another format, to make it a unique file name. The name of the remote log file will appear as the learner’s log in name, followed by the date and time stamp of their completion of the session. An example of this naming behavior would be “foo-11-1-20011-7:20am”.

### 5.1 What’s in a Log?

Log files contain detailed information on the learner behavior, including the time spent per page for non-testing pages, and for testing pages a question-by-question analysis of correct, incorrect, time to answer, and summary of performance. The following is an example of the mtt-trace.log from a development machine.
Table 3. An example D2P log taken from D2P/MTT.

MTT, using D2P Architecture Version: 1.10.0 Last modified on: November 27, 2012
Beginning at Fri Feb 22 13:24:12 EST 2013
Screen resolution: 1366x768
Logged in as Foo.
Going to next page. Time spent on page 1 6.90 seconds.
Going to next page. Time spent on page 2 8.43 seconds.
Going to next page. Time spent on page 3 1.10 seconds.
Going to next page. Time spent on page 4 8.70 seconds.
Going to next page. Time spent on page 5 5.26 seconds.
Video length: 11.68 seconds.
Incorrect. Time to answer: 2.591 seconds.
Incorrect answer for value ANGLE: 90; Correct value: 45
Incorrect answer for value SPEED: 10; Correct value: 5
Incorrect answer for value POINTS_OF_AIM: 4; Correct value: 1
Showing next video, video is: 100_5_45_1_1.mpg
Video length: 7.24 seconds.
Incorrect. Time to answer: 3.744 seconds.
Incorrect answer for value DISTANCE: 50; Correct value: 100
Incorrect answer for value POINTS_OF_AIM: 1; Correct value: 3
Showing next video, video is: 100_10_135_3_3.mpg
Video length: 9.96 seconds.
Incorrect. Time to answer: 2.818 seconds.
Incorrect answer for value ANGLE: 45; Correct value: 90
Showing next video, video is: 50_5_315_1_3.mpg
Video length: 10.66 seconds.
Incorrect. Time to answer: 3.261 seconds.
Incorrect answer for value DISTANCE: 50; Correct value: 100
Incorrect answer for value POINTS_OF_AIM: 2; Correct value: 1
Showing next video, video is: 100_5_315_1_0.mpg
Going to next page. Time spent on page 6 6532.636 seconds.
Upload of log successful.

In the log file in Table 3 we see that the system starts up without issue, the version number of the D2P software and its release date are posted. After this, we see a date and time stamp for the application’s start. Next, we capture the learner’s visual settings on their local machine, in this case, they are using a resolution of 1366x768, a large laptop screen or small monitor display. On the next line, we see that the learner has logged in using the name “Foo”. From there, we see a breakdown on the user’s time spent on each page, and in the case of question pages, how they answered each component question, if they were right or wrong, and how long it took them to complete the question.

5.2 Setting up a Receiving Server (feature not yet implemented)

To receive remote log files from your tutor learners, you must first configure your D2P jar file with the appropriate information. It is imperative that authors understand the means of transferring files is SFTP. This means the receiving machine and network must have this protocol enabled, and that the author will need to provide appropriate account details in order to allow D2P to send its log.

To configure the D2P file, run the .jar file from the command line with the following arguments:

1 <host location> <login name> <password> <directory path>

A fabricated example would be:
java -jar attv.jar 1 aserver.collegetag.collegedomain.edu aname apasswordshouldhavea#
users/logs/d2p

In this case, we’ve instructed the D2P language to send all log files at the completion of a session to the server located at ‘aserver.collegetag.collegedomain.edu’. It will attempt to access this server with the SFTP protocol providing the credentials ‘aname’ (given by the user at login) for the login name, and ‘apasswordshouldhavea#’ for the password. Finally, it will attempt to send the files to the remote directory ‘users/logs/d2p’ of the server located at <aserver.collegetag.collegedomain.edu> (a variable set in the executable).

It is critical that authors understand the limitations of this remote transfer. If one attempts to send a log from behind a heavily restricted network or firewall, it is very likely to fail. However, it is known to work using private computers and public ISP’s, as well as moderately controlled networks such as institutional access at a college campus.

In the case that the transfer fails, the tutoring language will prompt the user to send an e-mail to an address that receives such logs, and move a copy of the trace file to the user’s desktop to ease the process. In the future, we hope to make the address list extensible.

In the future, D2P may offer an alternate means of remote data transfer that will be discussed in detail in this section (HTTP SVN transfer).

5.3 The SQL Database
The SQL database in D2P serves as a means of storing internal states in the language, a redundant storage medium for learner behavior and performance, and a means to communicate with its cognitive architecture. Instructional designers at this time have no need to understand the internal mechanics or the schema for the SQL information. However, as the development of support for cognitive architectures improves and matures, this information will be vital in allowing authors to create models for their own tutor. Analysis of how the tutor works may need to know this information.

A list and examples of the schema and information stored will be available in this section of the document, while the following section will include means of querying the D2P language for learner statistics and behavior.

6. Communication Between Cognitive Models and the D2P Language
This section will in the future contain an overview of the communication model, as well as detailed examples of how to query information and the format in which the information is returned, similar to the XML tags sections.
7. Small Example Tutor

A simple sample tutor may be found for review in the file ‘sample-pages.xml’ that comes with the D2P release. Therein, the basic pages are used to demonstrate the proper management and placement of tags.

8. Extensibility and future changes

D2P Offers the Instructional designer a number of options in extensibility to assist in teaching a variety of subject materials to learners. However, there are a number of features that could be further extended to allow greater customization to the Instructional designer. These future changes may include: adaptation of the sound page or incorporation of sound directly into the tutor’s pages, the inclusion of a Java plug and play style page where it loads Instructional designer created Java applications into the tutor, and expanding the number of properties and internal features of the tutor that may be set with xml tags.

9. Summary

This manual introduced prospective instructional designers to the D2P language. First, we examined the conceptual implementation of the tutoring language, and the reasons that D2P could be used in place of other tutoring languages.

Next, we introduced the high level mechanics of the language, including the separation between the Java and XML components. After this, instructional designers were guided through the various types of pages, their pertinent XML tags, and how to properly create and use these tags to build a tutor.

Next, we gave instruction on information gathering and capturing, and how to transfer this information from the local learner’s machine to their own remote server. Lastly, the instructional designer was guided through the nature of D2P’s use of SQL and how to communicate their own learner models with the language.

References


A1 Example Menu Page

This provides an example menu page.

```xml
<bean class="com.tatuminteractive.mtt.view.pages.MenuPageView">
  <property name="controller">
    <bean class="com.tatuminteractive.mtt.control.PageController">
      <property name="parentController" ref="mttMain"/>
    </bean>
  </property>
  <property name="model">
    <bean class="com.tatuminteractive.mtt.model.MenuPageModel">
      <property name="title" value="A good Title"/>
      <property name="subTitle" value="Relevant subtitle or information."/>
      <property name="content">
        <![CDATA[<html><p><font face="arial">
          Here we put in some HTML formatted text to display to the user.</font></p>
          <p>This is a new paragraph.</p>
          <p>A third paragraph.</p>
          <p>The forth and last paragraph.</p>
          <![html]]>
        </value>
      </property>
    </bean>
    <property name="menuItems">
      <list>
        <bean class="com.tatuminteractive.mtt.model.MenuItem">
          <property name="title" value="1) Introduction"/>
          <property name="description" value="Something descriptive."/>
          <property name="thumbPath" value="images/rcoSIGHT.png"/>
          <property name="jumpsToIndex" value="6"/>
        </bean>
        <bean class="com.tatuminteractive.mtt.model.MenuItem">
          <property name="title" value="2) More Titles"/>
          <property name="description" value="Something descriptive."/>
          <property name="thumbPath" value="images/rcoSIGHT.png"/>
          <property name="jumpsToIndex" value="11"/>
        </bean>
        <bean class="com.tatuminteractive.mtt.model.MenuItem">
          <property name="title" value="3) Another Section Title"/>
          <property name="description" value="Another descriptive batch of text"/>
          <property name="thumbPath" value="images/rcoSIGHT.png"/>
          <property name="jumpsToIndex" value="29"/>
        </bean>
        <bean class="com.tatuminteractive.mtt.model.MenuItem">
          <property name="title" value="4) More Title"/>
        </bean>
      </list>
    </bean>
  </property>
</bean>
```
<property name="thumbPath" value="images/rcoSIGHT.png" />
<property name="jumpsToIndex" value="38" />
<property name="description" value="Something informative here." />
</property>
</list>
</property>
</bean>
A2 Example Image Page

This gives an example of an Image Instruction Page.

```xml
<bean class="com.tatuminteractive.mtt.view.pages.ImageInstructionPageView">
  <property name="controller">
    <bean class="com.tatuminteractive.mtt.control.PageController">
      <property name="parentController" ref="mttMain" />
      <property name="model">
        <bean class="com.tatuminteractive.mtt.model.ImageInstructionPageModel">
          <property name="imageHeight" value="400" />
          <property name="subTitle" value="" />
          <property name="title" value="We Are Under fire: TCCC" />
          <property name="imagePath" value="images/ImageFile18.png" />
          <property name="imageWidth" value="900" />
          <property name="content">
            <value><![CDATA[<html><p><font face="arial" size=5 color="black">Your response is influenced by whether you are taking <b>hostile fire</b>.  We will now discuss Tactical Combat Casualty Care (<font face="arial" size=5 color="green">TCCC</font>).</font></p>]]></value>
        </property>
      </bean>
    </bean>
  </property>
</bean>
```
A3 Example Video Page

A instructional page that uses a video as a means of media content for the learner.

```xml
<bean class="com.tatuminteractive.mtt.view.pages.VideoInstructionPageView">
  <property name="controller">
    <bean class="com.tatuminteractive.mtt.control.PageController">
      <property name="parentController" ref="mttMain" />
      <property name="model">
        <bean class="com.tatuminteractive.mtt.model.VideoInstructionPageModel">
          <property name="subTitle" value="" />
          <property name="videoHeight" value="475" />
          <property name="title" value="Challenges Associated with Moving Targets" />
          <property name="videoPath" value="videos/MTET MPEGs/MTET Instructional Video 1_2.mpg" />
          <property name="content">
            <value><![CDATA[<html><p><font face="arial" size=5>Because the target will most likely continue moving during the bullet's flight, hitting a moving target in many instances requires compensating for that movement.</font></p></html>]]></value>
        </property>
      </bean>
    </bean>
  </property>
</bean>
```
A4 Example Image Test Page

This is an example of an image test page, where the learner is presented with an image and asked to answer a multiple choice question about it.

```xml
<bean class="com.tatuminteractive.mtt.view.pages.ImageClassificationPreTestPageView">
  <property name="controller">
    <bean class="com.tatuminteractive.mtt.control.ImageClassificationPreTestController">
      <property name="parentController" ref="mttMain" />
      <property name="model">
        <bean class="com.tatuminteractive.mtt.model.ImageClassificationPageModel">
          <property name="subTitle" value="Timed Target Classification" />
          <property name="title" value="Pre-Qualification: Target Confirmation with the RCO" />
          <property name="questions">
            <set>
              <value>DISTANCE</value>
              <value>SPEED</value>
              <value>ANGLE</value>
              <value>POINTS_OF_AIM</value>
            </set>
          </property>
          <property name="filters">
            <map>
              <entry key="DISTANCE">
                <set>
                  <value>25</value>
                  <value>50</value>
                  <value>100</value>
                </set>
              </entry>
              <entry key="ANGLE">
                <set>
                  <value>45</value>
                  <value>90</value>
                </set>
              </entry>
              <entry key="SPEED">
                <set>
                  <value>5</value>
                  <value>12</value>
                </set>
              </entry>
              <entry key="POINTS_OF_AIM">
                <set>
                  <value>0</value>
                  <value>1</value>
                  <value>2</value>
                  <value>3</value>
                  <value>4</value>
                </set>
              </entry>
            </map>
          </property>
        </bean>
      </property>
    </bean>
  </property>
</bean>
```
A5 Example Video Test Page

This appendix has an example video-test (videoClassification) page. The learner is given a video when they press a button, and then are asked to classify it with a multiple choice radio button, and then press enter.

```xml
<bean id="rangeClassification"
class="com.tatuminteractive.mtt.view.pages.VideoClassificationPageView">
  <property name="controller">
    <bean class="com.tatuminteractive.mtt.control.VideoClassificationPageController">
      <property name="parentController" ref="mttMain"/>
      <property name="model">
        <bean class="com.tatuminteractive.mtt.model.VideoClassificationPageModel">
          <property name="title" value="Estimating Range"/>
          <property name="subTitle" value="Classification"/>
          <property name="relativeVideoPath" value="videos"/>
          <property name="questions">
            <set>
              <value>DISTANCE</value>
            </set>
          </property>
          <property name="filters">
            <map>
              <entry key="DISTANCE">
                <set>
                  <value>25</value>
                  <value>50</value>
                  <value>100</value>
                </set>
              </entry>
            </map>
          </property>
        </bean>
      </property>
    </bean>
  </property>
</bean>
```
A6  Question Builder Quick Start Information

The D2P Question Builder is a tool that allows an instructional designer to manipulate the database file used by the language. Specifically, this tool allows you to create, add, or remove questions from categories and question sets.

For typical users such as instructional designers, a copy of the D2P Question Builder is included in the D2P release.

For developers, the D2P Question Builder may be acquired from the ACS SVN repository.

Here are some example steps in using the question builder.

1. Double click on the QuestionBuilder.jar. This will display the following dialog:

2. Click on OK and this will prompt you to choose a question set database. Select the test.db database:

3. This will load the questions into the Question Builder GUI. The next step is to open a Question Set using the “Question Set->Open Set…” menu item:
4. For example, select the ‘TestSet’ Question Set:
5. This will load the sample Question Set. Feel free to edit the set as you wish. To edit a question, you select it a part of it by double clicking. It is hard to change a question’s type, so you are better off starting from scratch or duplicating a question if you have to change its type.
A7 Manual for the Page Editor

The D2P Page Editor is a Java-based tool that expedites and automates many portions of building pages within the architecture. Instructional designers use the Page Editor to create content for learners more efficiently and with automated tag handling, to mitigate the risk associated with human managed XML code and typographical errors or omissions.

1. Acquiring and running the Page Editor

Instructional designers can obtain the D2P Page Editor from http://acs.ist.psu.edu/d2p.

Developers may acquire a modifiable version of the Page Editor from the ACS SVN server through Ritter or Hiam. In the mttdev/trunk section, select the Page Editor folder and check out the content of that folder.

Once you have acquired the package, run the D2PPageEditor-v1.1.jar file. This is a Java file and requires a Java runtime environment to execute. The runtime environment can be acquired from Oracle’s web page.

2. Loading a tutor into the Page Editor

With the Page Editor running, select the File option from the drop down menu. In this menu, select Open. This option will create a dialogue box that allows the user to select the appropriate pages.xml file to edit. This XML file is the primary means to create a tutor. It will be located in the directory you have created for your tutor. If there is no pages.xml file, you may create a new one.
3. Adding a Page

After successfully loading a pages.xml file a list of created pages will appear in the leftmost dialogue pane. Buttons on the bottom of this dialogue pane will allow additional pages to be added, or pages to be removed.

To add a page, click on the New button. The addition of a page will take place just after the page selected in the page list. A dialogue box will open, allowing for a choice of page type.
To understand the capabilities and purpose of each page type, please consult the primary sections of this document.

After selecting a type of page, a second dialogue box will open with the required and optional tags for that type of page.

The instructional designer may then fill in the desired values for those tags. It is important to understand that this dialogue box will accept any type of input for these tags that are in a text box format—invalid entries will cause errors when attempting to run the tutor because this dialog box does not do type checking.

4. Deleting a Page

To delete a page, select the desired page for removal from the leftmost dialogue box that shows a list of all pages in the tutor. From this, select the Delete button. A dialogue box will open asking for confirmation. If you select yes, the Page Editor will explain what ramifications the deletion will have. This may include the removal of references to this page from menu pages, and a reevaluation of links throughout the tutor.

5. Move a Page

A page may be moved by selecting the Up or Down buttons. Select the page you wish to move from the leftmost dialogue box. Next, select the button that corresponds to the move you wish to make, and the page will be moved one step closer to the beginning, or one step closer to the end of the tutor. The page number and associated links will be automatically moved to correspond to the new ordering. Note that you may not move pages by simply clicking and dragging the page.
6. Edit a Page

Select the page you wish to edit from the leftmost dialogue box. After selecting the desired page, click on the Edit button at the bottom of this dialogue box. A new dialogue box will open, allowing the instructional designer to modify the tags and content of that page.

In this option, may see how the page will appear to the user by selecting the Preview button. When finished, click on the OK button to finalize your changes. As with page creation, if the Page Editor does not screen for erroneous content, and the tutor will throw errors if invalid information is provided.

7. Save the Tutor

To save the tutor, select the File dropdown menu option. To save your work and replace the version initially loaded, select the Save option from the dropdown menu. To save as a new pages.xml file, select Save As. This will open a dialogue box that allows the instructional designer to select the appropriate directory to save the new file.
8. Export Representation of Tutor to HTML

You can print the tutor to HTML to get a printable version and a version that you can get feedback on more easily and directly than from a running tutor. There are some discrepancies between the tutor and the HTML printout, but it is at least useful for debugging, spell correction, and checking coverage and tone of the material.

To export the tutor to HTML, select the File dropdown menu option. From there, select the “Export As HTML” option. This will open a dialogue box that allows the instructional designer to specify the specific directory they wish to save the html compilation in, as well as name the new file.

The exported HTML will show pictures used in the tutor, but only if the image directory, which contains all images used in the pages, is moved to the same directory as the exported HTML.