Date: January 23th, 2009

To: Dr. Karen Schairer

From: Babel Squad

Subject: Team Standards

Matacies

Requirements and Execution Plan

Richard Lester

Joe Flieger

Travis Hudson

Dean Dobransky

02/03/2009
Rev 1.0

Table of Contents

Introduction

Artificial speech is a way of teaching through a collection of constructed situations with little or no diversity. Early language instruction usually relies on artificial speech because it gives the student consistency. In this way, students learn examples that rarely translate into the lives of native speakers. The key to truly mastering a language lies in being able to formulate and understand natural speech. The problem lies in that natural speech, while identified as an extremely important part of teaching a language, is extremely difficult to convey to a student.

The objective of this project is to create a Web 2.0 portal that will give students the opportunity to learn about naturally spoken languages from a diverse set of people. This is done by using methods centered on a unique collection of videos of native speakers which our client, Dr. Karen Schairer, and will be accompanied by simple fun exercises to check for adequate understanding.

We have divided the requirements for the website into two sections. First, there are those for its users who must be able to create and edit their account, play a game and review their history. Secondly, there are those for its administrators who must be able to manage media content and site settings, create adventures, as well as track student progress.

Team Summary:

- Richard Lester, Team Leader
- Joe Flieger, Multimedia Developer
- Dean Dobransky, Client-side Developer
- Travis Hudson, Client-side Developer

All members of the team are seniors enrolled at Northern Arizona University in the Computer Science bachelors program.

Client Introduction

Dr. Karen Schairer is an Associate Professor of Spanish specializing in Spanish Linguistics for the Modern Language Department at Northern Arizona University. Her collection of oral histories is a unique and powerful tool for teaching natural language. She has undertaken many smaller projects that have shown exciting results and now plans to expand the scope of these successes through this project.

Problem Statement

Teaching natural language has been a problem with which a truly effective solution has evaded instructors for decades. Natural speech operates outside the confines of specific themes and vocabulary limitations. Its unpredictability and spontaneity leads to a language filled with pauses, false starts, blended words, varying velocity of delivery, and loose grammar. Removing these attributes from instructional videos helps to facilitate the understanding of the speakers to less experienced language learners. However, being able to understand these features are essential to a student's understanding of a language.

The client has an archive containing more than 250 videos of Spanish language natural speakers. These videos are the center of learning frameworks for teaching natural language. Through the years, the client has tried multiple technologies, ranging from

HyperCard to the present VISTA system. HyperCard was one of the first hypermedia systems and existed before the launch of the World Wide Web. While easy to use, it has become technologically outdated for today's needs for e-learning. It was withdrawn from sale in 2004 and support had become sparse long before that.

On the contrary, VISTA is a more up to date and versatile system but is cumbersome to use. The client's main objection to the VISTA system is that changing even small details, such as static text, require a lengthy series of steps. Because of this, Dr Schairer now uses PowerPoint to model these frameworks. This allows lessons to be easily designed and widely distributed, but still leaves much to be desired. The purpose of this project is to use modern technology and the web to create a system that can be accessed from anywhere by anyone while still allowing for a large degree of creativity. Furthermore, the product should be easy to use for both administrators and users.

The following are our main problem focus for this project.

- Create a simple yet powerful adventure creation tool.
- Allow for the use of multiple methods of evaluation.
- Translate created adventures into a playable state for the users.
- Create a way to playback a conversation based a previous adventure's history and its recordings.
- Ensure that there is a low overhead for the end-user.
- Ensure that any base language text is systematically replaceable to allow for the addition of other languages.

Solution

To solve this complex problem of language teaching, a solution involving several well-organized components is necessary. These components and the Web 2.0 standard help us in such a way that anyone can simply log on and start learning the language of their choice, without having to install extra software or web browser plug-ins. Given the wide range of people who will need access to this software the web seems to be the perfect platform for our solution.

When a professor is inspired to create a game that students can play to learn a language, the professor can simply log on and use a GUI (Graphical User Interface) to construct a tree-like structure of boxes. These boxes will be color-coded based on content and will include a unique and exciting scene that the player will encounter while playing the game. Scenes designed to challenge the intellect using a variety of predefined templates to create diverse situations. Some of these templates include the items listed below.

- Video questions
- Drag and drop sentence completion
- Drag and drop vocabulary matching

In the real world, native speakers speak very quickly, using slang and informal turns of phrase that can throw students off if they are unaccustomed to it. Video questions will show students how native speakers communicate while exposing them to a more diverse set of ways to communicate.

The drag and drop exercises are aimed at providing a very intuitive way of signaling what words go where, without having to resort to a complicated array of check boxes and radio

buttons. Armed with such a clean, compelling interface, user frustration disappears, leaving behind only the knowledge needed to succeed while traveling abroad in a foreign country.

Each adventure game that the professor wants to build to help students learn a new language is divided up into discrete scenes. Each scene is an individual activity that may involve watching a video or doing a small vocabulary quiz.

When the professor is done gathering together scenes into an adventure, it is uploaded to the Web, where students can eagerly log on and step through the adventure at their own pace and skill level. When a student completes one of their grand adventures, be it hiking along the Andes in Chile, or shopping for knickknacks in a European marketplace, saving a record of this journey allows professors to review and examine a student's strengths and weaknesses.

Using the history of the adventure the entire trip is available for play back by bringing together the videos the student encountered and the audio recordings of the student's responses. This helps the student relive the adventure they just had, as a way of thoroughly cementing the linguistic lessons learned through trial and error.

Not all language-learners are fluent English speakers. Sometimes a native Japanese speaker may want to learn Spanish, for example. To facilitate learning that is not English-based, the professor can specify new text that the user interface will rely on to deliver helpful pointers and directions in the student's native language.

Task Requirements

This product upon completion will be capable of the following set of tasks, in a manner as specified.

Account Creation

The program will allow for the creation of user accounts by following a simple link on the introduction screen or the login screen. Following these links directs them to a form that will collect demographic and academic information from the user. This is only a way to create user accounts where as administrators are the only ones who can create other administrator accounts.

Functional Specification:

- Users must have controls to initiate account creation.
- Administrators must have controls to create other administrators.

Example Scenario:

- A student would enter the site, and see the introduction page.
- The user would then be able to select registration.
- They would fill out a short forum and submit that form.
- The user would see a confirmation page to let them know they have registered successfully.
- The user would then be redirected to the log in page.

- They would then be able to enter their new log in information.

Performance Requirements
- For 80% of users they are able to register themselves within 60 seconds.

Preference Editing

Most preferences specified during creation will be alterable from the web interface. This includes items not limited to their password, their class, and their proficiency level. In the case that a student needs to change their class from a previous semester, or decided they chose too low of a proficiency level, and wish for a more difficult game, they be able to change these items.

Functional Specification:
- Users must have controls to edit their own user accounts.
- Administrators must have controls to edit global site preferences.
- Administrators must have controls to edit their accounts.
- Administrators must have controls to edit user accounts.
- Administrators must have controls to edit other administrators.

Example Scenario:
- A user who signs up under the hard difficulty, and quickly realizes that the difficulty is over their head.
- The user would be able to select from anywhere once logged in that they would like to change their preferences.
- The user would find the difficulty preferences from a list of preferences.
- They would then change their difficulty down to the medium difficulty.

Usability Requirements:
- For 80% of users the process of changing a setting once logged in takes no longer then 20 seconds on a user's first use of the program.

Starting and Resuming Game Play

Users are also able to start games from a list of adventures. Starting a new game allows the user to play through it, abandon it, or quit it at any time. The user also is able to continue a game they have previously quit, unless they have specifically chosen to abandon a game.

Functional Specification:
- Users must have controls to start a new game.
- Users must have controls to select game type.
- Users must have controls to resume previous game.
- Users must have controls to abandon a game.

Example Scenario:
- A player decides to start a game.
- Once logged in, the user would navigate to home from any location.
- The player would be able to select play a new game.
- They would then select the adventure they would like to play.

- The player would enter the game play mode.

Example Scenario:
- After the adventure starts, their computer is accidentally restarted.
- The user would return to the website.
- They would log in.
- They would select the resume game option.
- The user would select the saved game.
- The user is then returned to their previous game.

Example Scenario:
- The user then realizes this is the wrong module.
- The user would then be able to navigate back to home from any location.
- They would then select the abandon game option to delete their current game.
- The user would be prompted before the saved game is deleted.
- The user would be return to home.

Usability Requirements:
- Starting a game takes 80% of users fewer than 10 seconds on their first attempt.
- Resuming a game takes 80% of users fewer than 20 seconds on their first attempt, and fewer than 10 seconds afterward.
- Abandoning a game takes a user less than 30 seconds on their first attempt, and less than 15 seconds afterward.

Reviewing History

Users will be able to view their progress for each of their previous games, including those completed, abandoned, or in progress. A user will be able to inspect these histories, seeing the series of scenes, and the choices they made at those scenes as well as their scores for those games. The student will also be able to hear their recorded audio.

Functional Specifications:
- Users must have controls to review previous games.
- Users must have controls to review previous paths taken in games.
- Users must have controls to review previous scores received in games.
- Users must have controls to review the previous dialog between themselves and videos.

Example Scenario:
- A player wanted to see how they got the high score in the past.
- They would select to view their history.
- They would select that game from a list.
- They would then be able to see the sequence of events that lead to that score.
- They could then listen to the conversation they had over the course of the game.

Usability Requirements:
- For 80% of users they will be able to find a history of a specific game in less than 30 seconds.

Performance Requirements:
- When listening to recorded audio, the audio should buffer in less than three seconds from on the NAU campus.

- When listening to a dialogue of recorded audio, there should be no additional buffer time in between clips.

Management of Media Content

The administrator is able to add, remove, and view items in the media library. This media, pictures and videos, is then available in the adventure design features of the product.

Functional Specification:
- Administrators must be able to add media to the library.
- Administrators must be able to remove media from the library.
- Administrators must be able to view media in the library.

Example Scenario:
- A professor records new footage that they would like to use with the product.
- The professor would navigate to home from any point once logged in.
- The professor would be able to select the media library.
- They would choose to upload new media.
- They would then specify the digital movie file they would like to add.
- They would select to upload it.
- The professor would then be returned to the media library.
- This will make the newly uploaded media available for use in design modules.

Example Scenario:
- The professor wanted to remove a movie.
- The professor would navigate to home from any point once logged in.
- The professor would select the media library.
- They would then select a video from a list.
- They would select remove.
- The professor would be prompted if any adventures were using the video.
- The professor would then be prompted to confirm the removal.
- The video would then be removed, and the administrator would then be returned to the media library.

Usability Requirements:
- For 80% of administrators they will take under 20 seconds to add a new video on their second usage (excluding upload times).
- For 80% of administrators they will take under 10 seconds to remove an existing video on their second usage.

Performance Requirements:
- When viewing video from the media library, video should buffer in less than five seconds from on the NAU campus.

Specify Site Settings

The administrator must be able to set any non-adventure based text displayed on the site. A page will exist allowing the administrator to specify in text all the static text of the page, as well as images. This will allow for both skinning, and translation of the site.

Functional Specification:

- Administrators must be able to edit any non-adventure text.

Example Scenario:

- A professor wanted to adapt an already existing English teaching Spanish site into a French teaching Spanish site.
- The professor would navigate to their home page from any location once logged in.
- The professor would select site settings.
- The professor would select to change static text.
- The professor would change each item to reflect French.

Usability Requirements:

- Altering the site text takes 80% of new users under 20 seconds.

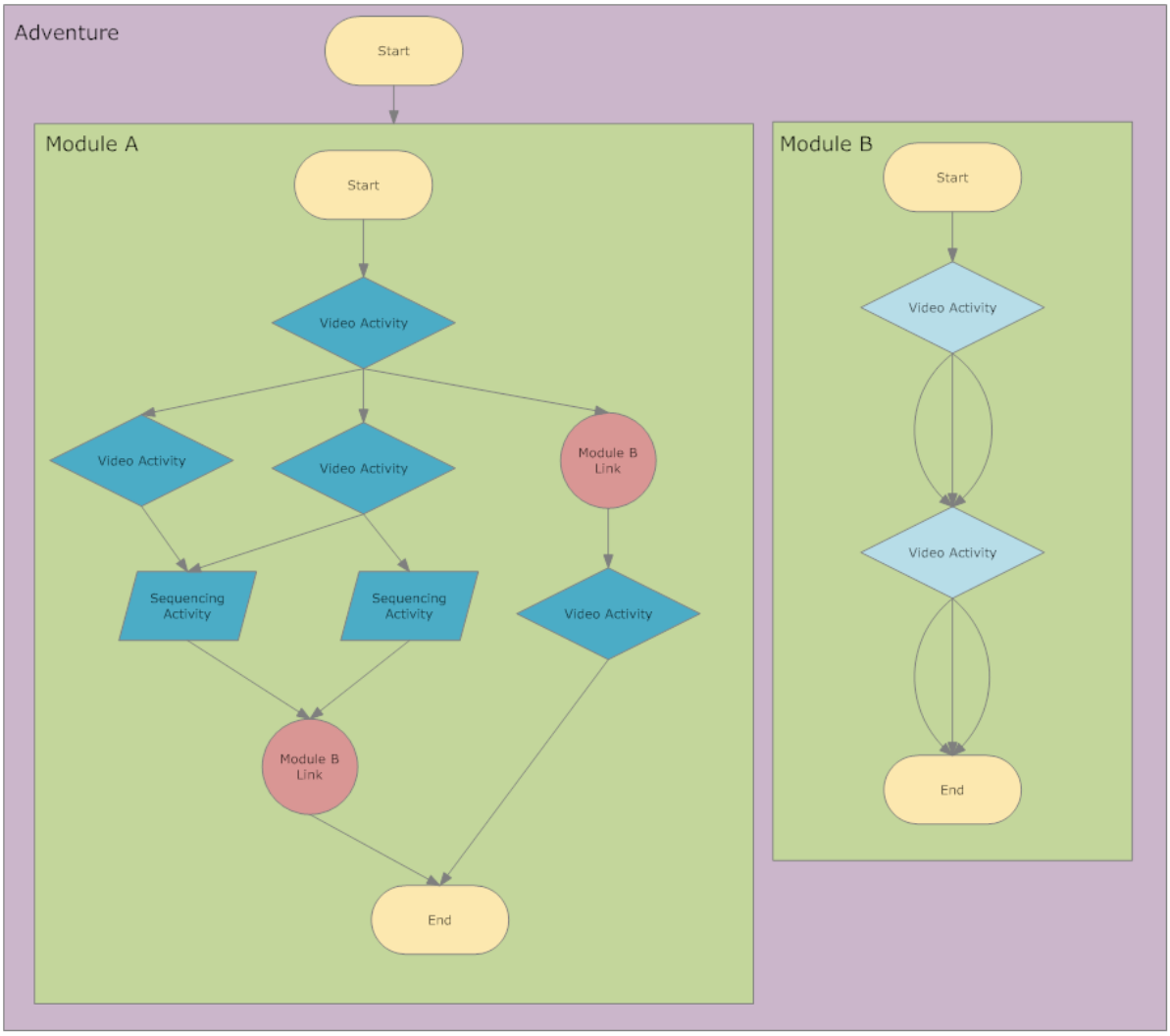


Illustration of the structure of an adventure

Create and Edit Adventures

The administrator will be able to create a new adventure and populate it with modules and activities that link to other modules and other activities. Each adventure, module, or activity will contain a start node and an end node. Adding new nodes to modules and templates is an easy way to diversify its structure. We will use a graphical display in a tree-like format to present the adventure and its components in a friendly manner.

Functional Specification

- Administrators must be able to use default dynamic templates.
- Administrators must be able to create modules.
- Administrators must be able to reuse modules.
- Administrators must be able to link any of the above formats.

Example Scenario:

- A professor decides to make a new adventure.
- The professor would navigate to home from any location once logged in.
- The professor selects a new adventure.
- A new module is then created to start the adventure.
- The professor can add new nodes to the tree.
- The professor can create other modules and link them.
- Once done, the professor would save the adventure.
- The adventure would now be available for students to play through.

Usability Requirements:

- Creating a new, empty adventure will take under 20 seconds for 80% of users.
- Adding new nodes takes under 10 seconds for 80% of users who have performed that operation previously.
- Filling out a new node may take additional time depending on the type of node.
- Specifying a link takes no more than 10 seconds additional time for 80% of users.

Performance Requirements:

- The interface to edit modules should respond to user interaction as a local application.

Edit Accounts and Track Student Progress

Administrators will have access to viewing student accounts while also being able to edit specific information. The same history information available to the student will be viewable by the administrators. Doing so will give administrators a graphical representation of the game with the path taken in a particular instance of the game overlaid on it.

Functional Specification:

- Administrators must be able to edit user accounts.
- Administrators must be able to access student histories.

Example Scenario:

- A professor decides to see how a student did on a travel adventure.
- Once logged in the professor would select to view student progress.
- A student would then be selected and the professor would then be redirected to a page detailing that student's history.

- The professor could then view what path the student had taken to evaluate if the student had easily maneuvered through in an optimal manner, or if they ended up taking detours because of a lack of understanding.

Example Scenario:
- A professor decides to check a student's score and assessment adventure.
- The professor would navigate to home from any location once logged in.
- The professor would select to view student progress.
- A student would then be selected.
- The professor could then view the student's scores.

Example Scenario:
- A student loses his password and needs the professor to reset it.
- The professor would reach to the home page from any location once logged in.
- The professor would select to view student progress.
- The student would be selected.
- The professor would select to modify the student's preferences.
- The professor would change their password.
- The professor would save the changes.

Performance Requirements
- For 80% of administrators who had previously been able to use this feature will be able to look up a student in under 20 seconds, and look up additional students in a row at no more than 10 seconds each.

General Activity Features

All activities will support both backtracking, and modification of the students score for that game. By backtracking, the student would be able to return to the previous node in order to make a different decision. Each activity could also affect the score of the player in turn reflecting how well the player handled the situations they were given. This would allow a student who ended up visiting jail several times, to have a lower score than someone who answered correctly the entire adventure.

Functional Specifications:
- Users must be able to backtrack in the game.
- A score will be recorded on a per-game basis.
- Users must have a history recorded of the game play.

Example Scenario:
- The user makes a mistake and wants to return to their previous location to change their answer.
- The user selects the backtracking function.
- The user is now at their previous state.

Example Scenario:
- A professor wants to build an assessment adventure.
- The professor creates the adventure as a series of activities all linking to the next activity.
- The professor assigns which answers are correct in order to form a path of correctness for the assessment.

Performance Requirements:

- Buffering of activities involving video media takes under two seconds.
- Once buffered, any video under a minute should not need to pause for additional buffering.

Multiple-Choice Activities

A multiple-choice activity consists of a single video or picture accompanied by a number of multiple-choice answers, each possibly linking to a different node. This will allow the user to interact with the game in a more realistic way then traditionally offered while still giving them some structure to rely on.

Functional Specification:

- Users must be able to navigate through administrator-designed templates.
- Administrators must have the ability to access stored media in order to add it to templates.
- Administrators must be able to instantiate default templates.
- Administrators must be able to add dynamic content to instantiated templates.
- Administrators must be able to edit templates.

Example Scenario:

- A student is playing through an adventure.
- A student is shown a video, along with possible responses.
- A student selects a response.
- The student is presented with the next activity by following a branch according to the chosen response.

Usability Requirements:

- An administrator could design a video multiple-choice activity in under a minute for 80% of users who already understand the process.

Sequencing Activities

A sequencing activity involves reordering using drag and drop to sort items. By passing a sequencing activity, the player moves onto a new section. Sequencing activities will also come with a help animation to instruct the user on how to use the activity.

Functional Specification:

- Users must be able to navigate through administrator-designed templates.
- Administrators must have the ability to access stored media in order to add it to templates.
- Administrators must be able to instantiate default templates.
- Administrators must be able to add dynamic content to instantiated templates.
- Administrators must be able to edit templates.

Example Scenario:

- A student is playing through an adventure.
- A student is confronted with a challenge of ordering items in order to continue.
- The student reorders the items into the correct order.

- The student moves on to the next activity.

Usability Requirements:
- An administrator could design a sequencing activity in under three minutes for 80% of users who already understand the process.

Access to database information

The administrator should be able to export all demographic data associated with their students. This might include their preferences, the adventures they have played or their total score.

Functional Specification:
- Administrators must have the ability to export selected values for their students.

Example Scenario:
- The administrator decides to export information for a study.
- The administrator would be able to navigate to the home page from any point.
- The administrator would then select to view student history.
- The administrator would then select to export a report.
- They would then select what information they would like to export.
- They would then select a location on their local computer to save the exported information.
- The administrator would then be returned to the student history page.

Usability Requirements:
- For 80% of administrators they should be able to export data in under 30 seconds, on their second use.

Performance Requirements:
- Accessing the database should not measurably lengthen the load time of the website.

Constraints and Non-functional Requirements

The product must implement the previously listed features under the following set of constraints.

- Accessible to disability users
    - Consideration must be taken for disability users to ensure equal opportunity.
- Web based
    - Easily accessible to students, including those in distance learning
    - Centrally maintained
- Not requiring a plug-in
    - No hassle to users
    - Universally accessible
- Self contained, installable, and royalty free
    - Product must be able to be redistributed
    - This allows for easy use in academia.
- Data driven design
    - Language independence
    - Flexible

Reliability Requirements:

- The system must be operational 99% of the time the hardware is operational.
- The system must respond normally when being used by at least 40 users simultaneously.
- The system must function properly no matter the amount content uploaded, given the hardware can support that content.

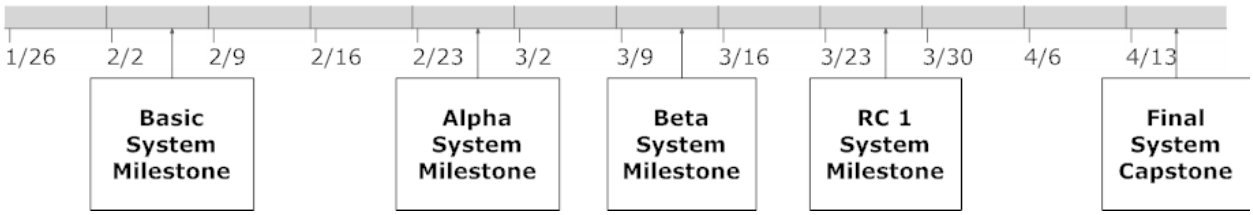
Technical Risks

We foresee the following list of features as being high-risk features, weighted by their difficulty to implement and significance in the final product.

- Pre-caching of videos over a web interface may be problematic because media would need to be sent to the client before the media is used, and maintained when the browser transitions into the next page.
- Streaming recorded audio from the browser to the server may lead to difficulty because an independent technology must be used to deliver media in the opposite the typical direction, without special purpose software on the client.
- Graphically representing the module data structure in a clear manner may be difficult to accomplish given the limited time span of the project. Also representing the module data structure in the database may be non-trivial, and translating the representation back and forth may prove difficult.
- Streaming video content is a possible risk due to lack of prior team experience in that field.
- Drag and drop interfaces may be difficult to communicate to the end user in a web environment.

Project Execution Plan

We have determined to split the development phase into five parts. Each of these five parts will end in a milestone. The below time line depicts our key milestones and is followed by a description for each.



- Basic Website Milestone:
  - February 6th
  - There will be a website, illustrating the general structure of the website.
  - Placeholder pages will contain a description of the eventual functionality of the page.

    - The page will also include one or more drag and drop demos, illustrating the ability to use drag and drop, and to discern order from draggable items.
    - The goal of the basic build will be to have a starting prototype.
- Alpha Website Milestone:
    - February 27th
    - Page navigation will be fully functional.
    - Playable templates will be working.
    - Basic games will be designable using the adventure designer.
    - Preferences will be changeable, and video will stream.
    - The overall goal of the alpha build will be to have an ugly but functional product satisfying most features.
- Beta Website Milestone:
    - March 13th
    - The basic website will contain a fair amount of polish.
    - The ability to upload audio will be available.
    - The adventure designer will be able to display a module graphically.
    - All templates are available and fully implemented.
    - The goal of the beta website is to have an almost completed final product to test on users.
- RC 1 Website Milestone:
    - March 27th
    - The website will contain a high degree of polish.
    - The website will support pre-caching of videos.
    - A dialogue playback of an adventure will be available to users.
    - All templates and their controls will be setup.
- Final Website, Capstone:
    - April 16th
    - There is some included fallback time for schedule slippage.
    - There is some included time for acceptance testing and recoding.

Appendix A: Glossary

Activity: An activity is the most basic element of an adventure that acts as a situation within a module. This may include a single prompt response scenario or an evaluation technique the student must pass.

Adventure: An adventure is an entire trip or assessment from start to finish. An adventure points to a module and once that module is completed the adventure ends.

Assessment Adventure: An assessment adventure is an adventure for which the goal is to act as a quiz.

Buffer: To allow a set amount of media data to collect before play, to avoid sudden starting and stopping.

Cache: Cache is a locally storage space for quick access to information.

GUI: GUI is an acronym for Graphical User Interface.

Game: A single game is a run through of an adventure, possibly in multiple sittings.

Hypercard: HyperCard was an application program for Apple Computer, Inc. that was among the first successful hypermedia systems before the World Wide Web. It combined database capabilities with a graphical, flexible, user-modifiable interface.

Instantiate: To instantiate is to bring into existence.

Media: Media is a picture or a video.

Module: A module is a linear graph structure with a beginning and an end, containing nodes representing activities and module links.

Module Link: A module link is a node of a module that would bring the user into the start of a different module. At the completion of this new module, the player would resume their previous path.

Node: A node is a single point in the graph that links anchor to in order to connect objects.

Parse: Is a term used to describe the way in which software interprets input.

Template: Is a predefined layout from which activities are created.

VISTA: E-learning system being used by Northern Arizona University.

Web 2.0: Web 2.0 is a term used to describe the changing trends in the use of the World Wide Web technology and web design that aim to enhance creativity, communications, secure information sharing, collaboration, and functionality of the web.