[Knights Who Say Ni!]

[CPython]

Version 2.0

Nov 1, 2005

[Team 7]

Powered By

www.construx.com

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Final Draft 1.0	Dustin Roberts, Andrew Nagel, Sam Gawthrop, John Burton	This is the initial version of the document that is to be turned in on Sep 12, 2005	09/11/05
Final Draft 2.0	Dustin Roberts, Andrew Nagel, Sam Gawthrop, John Burton	This is the final version of the document, scheduled for delivery on November 2, 2005	11/1/2005

.

Review & Approval

Project Plan Approval History

Approving Party	Version Approved	Signature	Date
Dustin Roberts	2.0		11/1/2005
Andrew Nagel	2.0		11/1/2005
John Burton	2.0		11/1/2005
Sam Gawthrop	2.0		11/1/2005
Mark Guzdial	2.0		11/1/2005

Project Plan Review History

Reviewer	Version Reviewed	Signature	Date
Dustin Roberts	1.0		10/28/05
Andrew Nagel	1.0		10/28/05
John Burton	1.0		10/28/05
Sam Gawthrop	1.0		10/28/05

Contents

1 Introduction

1.1 Overview

JES is a development environment that is used to give students a nice interface in which to program using Python. Many of the projects that students are assigned involve manipulation of images and audio files. This functionality is not supported natively in Python, instead Java libraries are used to perform the actual manipulation of the image and audio files. This solution works well but it relatively slow. The Objective of this project is to modify the source files that are provided with JES to utilize new library code that is written in C rather than Java. This should dramatically increase the speed with which manipulation of the image and sound files can be performed

1.2 Deliverables

- Media.py – Media.py is the source code for the project. It is a new Python library that will enable Georgia Tech CS1315 students, and other users from the public and academia, to edit and modify both JPEG image files and WAV sound files. Media.py was originally written in Jython using Java libraries, therefore only allowing it to run in the JES environment. The new Media.py will run within CPython, which will have the advantage of faster runtime. Utilizing Cpythong over Jython will also allow users to use any IDE they wish in order to write and run their code.
- Installer – There will be 2 versions of the installer. One version is for the Windows operating system. This installer will install Media.py, wxPython, and PyGames onto the users computer. The installer will also be written in Python, so it is expected that the user has already downloaded and installed the CPython runtime environment, available at *www.python.org*. The other installer will be for the Mac OS X operating system. This will also install Media.py, wxPython, and PyGames on the user's computer. This installer will also be written in Python, therefore the user is expected to have already downloaded and installed the CPython runtime environment for the Mac.
- Installation Documentation – There will be additional installation documentation written for any other user who wishes to install Media.py. These instructions will specify the necessary components that must be installed, and their location in order to use Media.py. These instructions will describe exactly what the installer is doing on the other platforms, but is written for users who either wish to install the components on their own, or are installing it on another operating system such as Linux.
- Documentation – There will be a small set of documentation that will accompany the source code. This documentation will include details on each function written for Media.py, and what that function can do. Mark Guzdial has written a book that gives examples on how to use all of Media.py, and all of its contents

will apply to the new Media.py. The documentation will be in the form of Py-Doc's, which is a standard documentation procedure for all Python source code.

1.3 Assumptions and Constraints

1.3.1 Cross-Platform

Our newly developed code must not be platform dependent. We will assume that users are using either Windows or Mac based platforms. Linux/Unix is not a requirement of the project, but we will test the code for Linux and Unix, in order to find any major bugs. For delivery we will assume that the presence of no major bugs on the Linux and Unix platform is satisfactory for our end user.

1.3.2 C-Code

The libraries used to implement both the window for the image display GUI and the modification of images and sounds are C libraries that already exist, specifically PyGame and wxPython. These should be standard libraries that should fit in well with an easy to use installation system.

1.3.3 Resources

Media.py has no specific restraint on resources other than it must run on any platform that meets the minimum requirements for Python. This only requires that the computer must be able to run the operating system for the version of Python the user is running. Our code is developed for Python version 2.3, which is available on both Mac OS X and Windows XP, the two required platforms.

1.3.4 Time

Due to this project being limited to 1 semester in length, we will have until December 9th of 2005 to complete all parts of the project. This will require each member of the team to put in a minimum of 10 hours per week on the project for a total of 40 hours per week from the entire team.

1.3.5 Speed

There is an assumption that there will be a speed increase by implementing the Media.py functions in CPython, rather than the currently existing Jython implementation. From preliminary tests there is a large increase in speed with normal python functions. This includes a reduction from 10 seconds to about .6 seconds when running a built in function with CPython over Jython.

1.3.6 Libraries

The only constraint on our libraries is that any library used must be available for both Windows and Mac platform's version of CPython. These libraries can be written in any language, as long as they are available for both platforms.

1.4 Reference Materials

Introduction to Media Computing by Mark Guzdial, is the reference material we are using to design the new Media.py. It has examples, explanations and self tests that will allow us to evaluate our Media.py.

1.5 Definitions and Acronyms

CPython – Implementation of python done in the C programming language.

CVS – Concurrent Versioning System

Eclipse – Development suite

GUI – Graphical Users Interface

IDE – Integrated Development Environment

JES – Jython Environment for Students

Jython – Implementation of the python done in the Java programming language.

Media.py – File that will house all functions

PyDev – Python Development plugin for Eclipse

PyDoc – Documenting standard for python

PyGames – Library used to implement sound and image editing in CPython

QA – Quality Assurance

SDMLL - Simple DirrectMedia Layer Library

UML – Unified Modeling Language

2 Management Structure

2.1 Project Lifecycle

There will be three phases to the project. First, there is the learning and training stage. This stage starts on August 30, 2005 and will complete by September 13, 2005. The purpose of this stage is for everyone on the team to familiarize themselves with Python, Eclipse, and the Media.py script that is currently available form Mark Guzdial.

After the learning and training stage is over, we will immediately move into the development stage of the process. During the development stage we will follow an evolutionary development model. As parts of the program are understood, we will implement them using the PyGames.py module. The actual development will take on a waterfall approach, where after the requirements are defined, we will design the software, then implement it and test it. Since we will be learning as we are going along, we will inevitably have to refine our requirements and design. Development should begin on September 28 and be completed by October 19.

The final stage of our development will be the testing and deployment. Testing will begin on Wednesday October 19 and be completed by Wednesday November 2. Although testing will be required throughout the development process, more thorough testing will occur at the end of the life cycle. Beta Deployment will begin on Thursday November 3 and beta testing will be ongoing until Wednesday 23. Development and testing will all be complete by Wednesday November 30.

2.2 Project Organization

Since our team is only 4 people, there are no smaller groups within the organization. There are people who are the head of certain parts, but everyone is to play a supporting role for those parts. Dusty Roberts is the project manager and is in charge of testing and documentation. Andrew Nagel is head of implementation and will be heading up the images part of the program. Sam Gawthrop will be in charge of GUIs and the design lead. John Burton is in charge of sound and will be the planning and tracking lead.

2.2.1 External Interfaces

Mark Guzdial is the only external source that is vital to our project's completion. He is both the advisor and the customer for this project. All four members of the team will have direct access to Mark Guzdial via e-mail or in-person meetings. For any type of major questions, that anyone on the team needs to be aware of, all questions should be directed to Mark Guzdial, through Dustin Roberts.

2.2.2 Internal Structure

2.2.2.1 Roles and Responsibilities

Role	Responsibility
Project Manager	In charge of making sure all deliverables are turned in on time, handing out responsibilities, and finding resources needed.
Planning and Tracking Lead	In charge of setting the schedule, tracking all development and reporting to the project manager and deviations from the plan.
Requirements Lead	In charge of getting requirements for the project. Also in charge of deciding who will get which requirements.
Quality Assurance Lead	Checks code for proper comments, tests code for any problems.
Design Lead	In charge of UML and designing the system.
Implementation Lead	In charge of deciding how parts of the system will be implemented, via PyGame.
Development Engineer	Programmer and minor design decisions.
QA Engineer	In charge of writing both test code and test cases.

2.2.2.2 Staffing

Role	Staff Member	Start Date	End Date
Project Manager	Dusty Roberts	08/22/05	12/2/05
Planning and Tracking Lead	John Burton	08/22/05	12/2/05
Requirements Lead	Sam Gawthrop	08/22/05	12/2/05
Quality Assurance Lead	Dusty Roberts	08/22/05	12/2/05
Design Lead	Sam Gawthrop	08/22/05	12/2/05
Implementation Lead	Andrew Nagel	08/22/05	12/2/05
Development Engineer 1	John Burton	08/22/05	12/2/05
Development Engineer 2	Dusty Roberts	08/22/05	12/2/05
Development Engineer 3	Andrew Nagel	08/22/05	12/2/05
Development Engineer 4	Sam Gawthrop	08/22/05	12/2/05
QA Engineer 1	Dusty Roberts	08/22/05	12/2/05
QA Engineer 2	John Burton	08/22/05	12/2/05

2.3 Communication

There will need to be multiple lines of communication open. First, telephone numbers will be exchanged for all persons involved in the project. Also, AOL Instant Messenger screen names will also be exchanged. The group will meet with Mark Guzdial every Tuesday at 3:00 in his office. Also there will be team meetings every Monday and Wednesday at 4:00 in the College of Computing Commons area. Email will be used to schedule extra meetings or to answer questions that were not asked in meetings.

2.4 Risk and Asset Management

1. The top threat to our project will be the risk of PyGames not implementing all the functionality we need. If this is the case, we will be required to write our own C modules to use in the CPython language, and we will have to drastically change the scope of the project. We will complete our investigation into PyGames by September 21. Mark Guzdial has already done most of the research; he has recommended this library to us because he has determined that it has all necessary pieces.

2. Students dropping the class are another large risk. If this was to occur, we will most likely be forced to reduce the amount of functionality that is available in the delivered product.

3. Teammates who do not do their part is a risk that may possibly occur. If this happens we will be forced to go to the professor in hopes that he can get them back on track. If this does not work, we will be forced to reduce the amount of functionality, or people will have to put in more time on the project.

4. A low risk is the team not being able to learn Python good enough to complete the project. This is a very low risk and can be solved by purchasing books and doing the examples for python. If a teammate is still struggling with Python we will go to pair programming for the first half of the project. This will pair up a good python programmer with the one that is struggling. If all teammates are struggling with Python, we will sit in on cs1315 lectures and get guidance from Mark Guzdial.

5. Another risk is that CPython will not be noticeably faster when media.py is implemented. Although increased speed is the outcome we are hoping for, it is not a requirement. If our implementation is not faster, we will document it very well for all users.

6. The final risk is that our implementation cannot be cross-platform without small alterations to make it work on respective platforms. If that were to occur, we will have to deliver 2 different products, one for the Mac and one for Windows.

2.5 Startup

In order to get the project started, everyone will need to install Python 2.4 on his or her computer. They will also need to install the latest Java runtime environment (5.0) and install Eclipse 3.1. Finally everyone will need PyDev for eclipse in order to develop in Eclipse.

2.6 Closeout

The project will end when all of Media.py has been implemented in a new Media.py that uses CPython, rather than Jython. We are also required to run a satisfying set of example code from Introduction to Media Computing by Mark Guzdial. The satisfying set will be determined by Mark Guzdial.

3 Planning and Control

3.1 Estimate

Estimation is more of a challenge with this project because no one has a scope as to how much coding will be required, because we've never written in Python before. That being said, most estimates that have been found in books, say that Python development is usually quicker than Java or C. We can assume that doing this project would take 3 months time, with 40 hour weeks from the group. The 3-month time limit is imposed by the length of a semester. The 40 hour weeks are determined by the fact that a 3 hour class is suppose to take 10 hours a week outside of class to complete all projects and homework. 10 hours is actually minimum that each person should put into the project every week. That is the most amount of time we will have, of course we will be learning Python, and doing our testing in that time period also. Initially we estimate that the time to complete this project will be about 800 man-hours. This includes coding, testing, beta release, and final release.

3.1.1 Estimation Process

Due to the ambiguity of what we know about our project so far, we will have to refine our estimations quite often to reflect what is actually getting done, and what can be done in the future. It is reasonable to expect our estimations to change weekly up until October 1, at which time we should be fully vested into the project, and have a good handle on estimated time to completion.

We will change our estimation based on the time it takes for us to complete coding tests at the end of the "Learning Python" chapters. We will estimate how long it took to complete 3 of the tasks, and compare its size to the functions in the existing Media.py

3.2 Resource Identification

3.2.1 Staff

At all times during the project there will be 4 people available to do the work. Each person will be expected to do 10 hours of work per week on the project. Each person in the group will be expected to shift from one role to another as the project progresses.

3.2.2 Time

Calendar time available for this project is August 22 – December 2. Each member should be available for 10 hours of work for every week of the project.

3.2.3 Cost

Although Python has a small learning curve, we still have to spend time learning the correct way to do things in the language. This will require us to refactor code more often than if we

were already experts. Therefore we expect everyone to put in about 200 hours of work this semester for a total of 800 hours.

3.2.4 Materials

There is no special equipment required for the project.

3.3 Resource Allocation

3.3.1 Milestones

Milestone 1: 9/12/2005 – Turn in Project Plan

Milestone 2: 9/14/2005 – Turn in Requirements

Milestone 3: 9/21/2005 – All Materials learned

Milestone 4: 9/23/2005 – Turn in Design Document

Milestone 5: 9/28/2005 – Skeleton code of Media.py

Milestone 6: 10/5/2005 – 3.3.2.5.1.1, 3.3.2.5.2.1, and 3.3.2.5.3.1 from WBS

Milestone 7: 10/12/2005 - 3.3.2.5.1.2, 3.3.2.5.2.2, and 3.3.2.5.3.2 from WBS

Milestone 8: 10/19/2005 – 3.3.2.5.1.3, 3.3.2.5.2.3 from WBS

Milestone 9: 10/19/2005 – Turn in test plan

Milestone 10: 10/26/2005 – Code Freeze

Milestone 11: 10/26/2005 – Deliver beta system

Milestone 12: 11/2/2005 – Project Plan regrade

Milestone 13: 11/9/2005 – Requirements Document regrade

Milestone 14: 11/16/2005 – Design regrade

Milestone 15: 11/16/2005 – Final Code Freeze

Milestone 16: 11/23/2005 – Test plan regrade

Milestone 17: 11/30/2005 – Deliver finished product

3.3.2 Work Breakdown Structure (WBS)

3.3.2.1 Research

3.3.2.1.1 Learn Python – All teammates

All of our group members are new to working with Python and so an obvious first step is getting to know the way in which the language works. The syntax, key functions, and coding conventions are all basics topics, which have to be learned in this phase of the project. Since all team members are serving as Software Engineers in addition to their other duties it is necessary for all group members to work in this learning phase

3.3.2.1.1.1 Each group member must purchase "Learning Python" from O'Rielly Publishing.

3.3.2.1.1.2 Each group member must complete reading and all tasks in "Learning Python" by September 14, 2005.

3.3.2.1.2 Review the Current Working System – all teammates

This project requires the team to take an existing system that works under Jython, and make it work under CPython. In order to complete this task, we must learn both how to use the current system and how it works. This will require us to use and study the existing code.

3.3.2.1.2.1 All teammates must download and install JES.

3.3.2.1.2.2 All teammates must study the current Media.py that already exists in JES.

3.3.2.1.2.3 All teammates must be able to use every working function of the Media.py within JES by September 21, 2005.

3.3.2.1.3 Review API for C Media Library – all teammates

We will be altering Media.py to use an existing CPython library called PyGame. This will require us to review PyGame and understand its abilities and limitations.

3.3.2.1.3.1 All teammates must download and install PyGame.

3.3.2.1.3.2 All teammates must be able to use PyGame within Python by September 21, 2005.

3.3.2.2 Planning – Dustin Roberts lead, all others support

Use the knowledge gained through the research we will have completed to draw a basic structural roadmap that can be used in the coding phase.

3.3.2.2.1 Role Assignments

Role assignments were decided on by volunteering. The team discussed each person's strengths and weaknesses and through that we determined who should do which part. Dustin Roberts is the project manager and is in charge of testing and documentation. Andrew Nagel is the implementation lead and is in charge of images and is also a tester. Sam Gawthrop is in charge of the design and will be doing all GUI parts, including a file and folder chooser. John Burton is the planning and tracking lead that will also be in charge of all sound parts.

3.3.2.2.2 Planning Document

The planning document is what you are reading now. The initial draft is due on September 12. In order to meet this deadline, each member has been assigned 1 part and all members will do part of section 3. When each person has done their part (by September 9) we will meet, integrate, and edit.

The re-grade of this document is due by November 2, 2005 and will be completed by Dustin Roberts no later than October 31, 2005. All edits will be completed by November 1, 2005 by the rest of the team.

3.3.2.2.3 Estimation

An accurate estimation of how long the project will take cannot be completed by the initial turn in. Preliminarily we have determined to have all parts coded by October 19, and all major bugs fixed by October 26. We will estimate time based on the amount of time it takes us

to complete projects from "Learning Python" and their compared size to functions in the existing Media.py.

3.3.2.3 Requirements – Andrew Nagel lead, others assist

The JES programmers from 2003 have already laid out the requirements. Mark Guzdial has provided us with those requirements. Additionally we have the restrictions of having to use libraries that run in CPython.

3.3.2.3.1 Requirements Document

The requirements document is due by September 19. In order to meet this deadline the document will be broken into 4 parts and each member will do that part. Members are expected to have their part done by September 17 for integration and editing.

The re-grade of the requirements document is due by November 9, 2005. Dustin Roberts will again be in charge of editing it. He will have completed that by November 7, and the final draft will be written by November 8, after all teammates have proofread and edited it.

3.3.2.4 Design – Sam Gawthrop lead, others assist

Sam Gawthrop will mainly do the design part of the project. Mark Guzdial has provided the previous project's design to us. We will be using it as a template to our design.

3.3.2.4.1 Design Document

Each member will be assigned a written part of the document to complete. They will be required to have this completed by September 18. This will allow Sam to edit those parts and complete the design using UML. Our advisor has told us that only a class diagram is necessary for a project of such a small scope.

Sam will complete the document by September 22, 2005. The re-grade will also be completed by Sam and will be finished by November 14, 2005. All teammates will then proofread and edit the document for Sam.

3.3.2.5 Coding

This is the actual implementation phase where it will be necessary to actually write the bindings that are a necessary part of this project. This phase can be broken up into pieces as the things that need to be coded have distinct boundaries between them.

3.3.2.5.1 Audio – John Burton

One of the functionalities required by JES is the ability to parse multiple audio formats into samples that can be read and modified. The actual work of parsing, reading, and modifying will be handles by PyGame, but the calls of these underlying functions and the formatting of the results will be handled in this coding section.

3.3.2.5.1.1 Parts due by 10/5/2005

play(), getLength(), blockingPlay(), getSound()

3.3.2.5.1.2 Parts due by 10/12/2005

getSamples(), playAtRate(), playAtRateDur(), getSamplingRate(), getSample(), getSampleValueAt(), getSampleObjectAt().

3.3.2.5.1.3 Parts due by 10/19/2005

makeSound(), writeSoundTo()

3.3.2.5.2 Images – Andrew Nagel

Another functionality that must be provided to users is the ability to load a variety of image files and parse them into a collection of pixels each of which can be read and modified. Again, PyGame will do the work but we must provide the functions to the users.

3.3.2.5.2.1 Parts due by 10/5/2005

getWidth(), getHeight(), makeColor(), pickAColor(), makeDarker(), makeLighter()

3.3.2.5.2.2 Parts due by 10/12/2005

getPixels(), getPixel(), getRed(), getBlue(), getGreen(), setRed(), setBlue(), setGreen(), getX(), getY()

3.3.2.5.2.3 Parts due by 10/19/2005

distance(), writePictureTo()

3.3.2.5.3 GUI – Sam Gawthrop

The final phase of coding implementation is the ability to display a modified image to a user using a GUI. This will involve using an underlying windowing system to display a set of pixels that have been modified by the Images portion of the media library for the user to see the result.

3.3.2.5.3.1 Parts due by 10/5/2005

pickAFile(), PickAFolder()

3.3.2.5.3.2 Parts due by 10/12/2005

display a picture

3.3.2.5.4 Installer – Sam Gawthrop

There is an installer in place that installs Python on a user machine. We are going to create our own that installs Mdeia.py, wxPython, and PyGame. This installer will be written in python and is due by October 26, 2005.

3.3.2.6 Testing – Dustin Roberts

3.3.2.6.1 Code Functionality

Testing for this project involves taking the examples that work now under the current JES implementation and ensure that they work correctly using the new media.py bindings. As a result the test cases for this project are largely written already and it is a matter of creating a test process that utilizes the test cases we already have provided for us.

3.3.2.6.2 Portability

This project must be capable of running with a fairly simple installation process on Windows and Mac OSX systems. The changes should be minor if anything at all, but testing will be necessary to ensure portability.

3.3.2.6.3 Test Cases

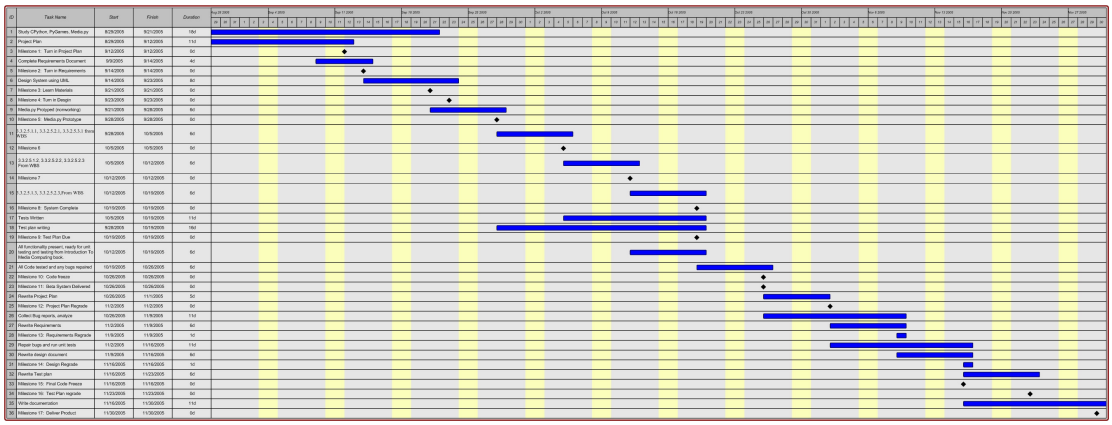
Test cases will come from the book provided by Mark Guzdial. The sample code in the book must be fully operational with the new Media.py.

All test cases must be completed by October 19, 2005.

3.3.2.6.4 Test Document

Dustin Roberts will be in charge of the entire test document. He must have the initial one completed by October 19, and the re-grade completed by December 2, 2005.

3.3.3 Schedule



You can also view this schedule at: http://swiki.cc.gatech.edu:8080/cs4911b-fl05/132

3.4 Tracking and Control

Each team member will be required to track the number of hours they work on the project. This number should be reported to Dusty Roberts by 12:00 midnight every Sunday. Dusty Roberts will post the status report to the class Swiki no later than 4:00 on Monday. Cost is not an issue, therefore they do not have to be tracked. The schedule shall be updated weekly to show how much progress has been achieved, and how much more needs to be done. Functionality will be judged according to how the already implemented Media.py is working. Teammates should also notify all team members when any task is completed, or any milestone reached. Quality will be judged by speed of completion of a test case and the ability to recover from erroneous input. Project Manager will be in charge of assigning tasks to individual members in the group. Among these tasks will be modifications to the project planning and tracking documents and so he will be responsible for tracking and integrating changes to these documents.

4 Technical Process

4.1 Engineering

4.1.1 Environment

Most work will be individually assigned and performed wherever the teammate feels comfortable. Regular meetings will be held to keep the team in touch and on track with the project.

4.1.2 Methods, Tools and Techniques

For documentation, Microsoft Word will be used, and if any diagrams are needed, they will be produced in Visio using UML and distributed in jpeg image format. The primary form of communication will be email, with other forms used when they are more appropriate.

4.2 Technology

4.2.1 Environment

This project can be developed on any machine running the latest version of Microsoft Windows XP or MAC OSX. Portability is theoretically already done for us, but will be tested in the testing phase of the project. All programming will be done in Python. All members of the group will need to have the JES source tree and Java. Additionally, every member will need CPython and the SMDLL library, installed and compiled. Each employee will be allowed to use his preferred text editor, as long as it works for Python.

4.2.2 Methods, Tools, and Techniques

Each team member can use whichever development environment they are accustomed to. Members are required to have all of their code working under CPython 2.3 available at *www.python.org*. Members are also required to use PyDoc to comment their code.

We will use CVS as our management tool. Although this is a small project as far as coding goes, it will be very useful for our documentation.

4.3 Infrastructure

N/A There is only one file being modified, and everything is listed in the "Environment" section above.

4.4 Project Artifacts

N/A

5 Supporting Plans

5.1 Configuration Management

CVS will be used to track version changes and to keep a repository of all code and documents. Sam Gawthrop will set up the CVS repository. All team members will submit their code only if it is running under CPython 2.3. Team members are not allowed to submit non-working code. Documentation will be kept in the repository also. The documentation will be written in Microsoft Word format, but the final copy will be a PDF.

5.2 Quality Assurance

The use of continual testing through the projects will be integral to our quality goals. Bug tracking and reporting will be handled with CVS comments and the versioning system. The project manager will enforce the plan.

All code must conform to the standards prescribed by *www.python.org*. This includes developers using the PyDoc standard. When any code is submitted to the repository it must be able to run using Python 2.3. Code that will not compile will be deleted from the repository after the developer has been notified.

5.3 Testing

Our project, by its very nature, gives us unit testing. The projects from the 1315 class will be implemented as the unit testing, as they contain calls to each of the objects and procedures in media.py. Furthermore, there are PyUnit tests already available from a previous semester's project. We will use each test that is appropriate for our project.

5.4 Deployment

Mark Guzdial is both our advisor and customer, so deployment of the project will simply mean turning over the project to him. We are going to test the program by running the projects that the 1315 class will use, so the test phase will be an almost complete simulation of the actual deployment. As part of the testing, a standardized installation procedure will be assembled, and this final package will be delivered.

5.5 Integration

Integration is rather moot for our project. We should be able to replace media.py with calls to the sdl libraries and the rest of the system will not be affected, but testing will show whether or not there is a need for more integration.

5.6 Procurement

N/A

5.7 Operations

An installation guide will be part of the delivered package.

5.8 Maintenance

N/A

5.9 Staff Development

Ideally, the change from Jython to Cpython should be seamless to the teaching of the 1315 class, except for the change from JES to a different (production) IDE. No training should be necessary, but some documents may be necessary to acclimate the new users or ease the transition from JES.

5.10 Product Acceptance

Mark Guzdial will accept the project, once we show him that it can do all the examples in the book and passes all unit tests that are applicable to our project.