Phase II: Design

What This Is
This guideline lists the process steps for developing a work breakdown structure that identifies all the tasks in the project’s work, the first step in creating a detailed project plan and schedule.

Why It’s Useful
A key to successful project scheduling is to break down the project goals into tasks BEFORE you consider delivery dates, resource constraints, specific named resources, or task dependencies. This helps you to objectively identify all of the work necessary without subconsciously leaving out real work in order to fit date constraints.

The Work Breakdown Structure (WBS) step helps accomplish the following key objectives:
- Develops an objective, rational view of the amount of work required (VITAL!)
- Helps team grasp the skills required and amount of resources required for the project
- Provides a clear framework for assigning to individuals a clear task definition and delegate the responsibility for completion
- Lays a foundation for analyzing the task dependencies and for isolating and managing risks
- Lays a foundation for developing a bottom-up estimate for the project schedule
- Allows tradeoffs to be made consciously and with the proper consensus involved.

How to Use It (Project Managers)
1. Review the process steps for creating a work breakdown structure starting on the next page.
2. Identify how team members will be involved in creating the WBS and educate them on their role. The primary objective is to get all of the team participants to contribute to the definition of the work.
3. Determine an appropriate organization for your WBS and begin identifying major work efforts. See the detailed guidelines which begin on the following page for how your WBS can be organized.
4. Break the top level of your WBS further into a hierarchical set of tasks. Use the guidelines starting on the next page for deciding how far your WBS must be broken down to ensure that you’ve defined enough detail for scheduling your project.
5. This WBS information will feed the next step in the planning/scheduling process: Assigning resources to tasks.
Overview of WBS Creation

The WBS is the first step in developing a detailed DESIGN work plan for the project. Task Identification and WBS Creation bridges from the early Scope definition (DEFINE) to creation of a detailed project schedule.

1) **Start with: The Scope**
Definitions at the highest level what has to be done—what must be created and delivered to the project’s customers. What you WILL cover, what you WON’T cover, and WHY?

2) **Create: The Work Breakdown Structure (WBS)**

*Create a Functionality Chart (as seen in Sarah’s handouts)*

3) **Based on the WBS, develop: The Project Schedule**

a) Create a current Structure Chart demonstrating the Website’s Organization or Site Map what they have now and . . .

b) Create another Site Map of what your team believes the site should look like
The Work Breakdown Process: From the Top Down

Use the following top down approach to iteratively create your WBS:

1. First identify the major components of work to be accomplished.
   Identify 5-10 major work groups which primarily set up how the work is expected to be organized, to form the highest level or first level of the WBS. Choose this top level of your WBS to match your organizational and internal work methodology—the best way to organize the work for this project given the project complexity; how the work is spread across your organization; and how the work will be tracked and managed. Here are seven approaches. (ref: Effective Project Management, Wysoki et al, pg. 120)

2. Identify the next level of work (Level 2) under each major component and list them under their top-level groups. This can be done with indented lists or graphically in an organization chart.

3. Continue to break down the work under each Level 2 items
   Break down to the level of task detail that ensures the top-level components are broken down far enough for identification of all the work that needs to get done. Details under some may break into three or four levels. Others may require no more detail, or only one additional level.

   Write preliminary plans if necessary to help scope the later cross-functional efforts: documentation, manufacturing, testing, etc. Remember what Sarah stated in her lecture, define your tasks, and then break them down to smaller tasks and then even smaller.
Examples and Definitions of Typical WBS Levels

The following is one possible work breakdown approach starting with project lifecycle **phases** at Level 1, major **deliverables** of each phase as Level 2, and the activities, then tasks, to create each deliverable as levels 3 and 4.

<table>
<thead>
<tr>
<th>Phases (you have left II, III, IV, and V)</th>
<th>Identify major phases of work (e.g. specify, design, build, test…)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Major Project Deliverables</strong> and related milestones (for this Phase II Only)</td>
<td>Identify the major component deliverables of work required (e.g., subsystems that must be designed, built, tested, during each phase.)</td>
</tr>
<tr>
<td><strong>Activities (this is overall that the team will accomplish)</strong></td>
<td>Identify the activities needed to create those deliverables. (Some interim, smaller deliverables such as documents may be involved.)</td>
</tr>
<tr>
<td><strong>Tasks (who will be responsible for each?)</strong></td>
<td>Break the activities down to an appropriate level of task detail.</td>
</tr>
</tbody>
</table>
How to Engage the Team in Creating the WBS

This is a team process and should not be done by the Project Manager alone. Plan to engage the project team to analyze and break down the work into a sequence of deliverables, activities, and tasks until the detail defines a manageable project. Here is a suggested process:

1) The Project Manager can propose what the top level of the WBS should be.

2) Call a core team meeting (on SKYPE and then have another one with Debra on SKYPE) to review the suggested top level of the WBS and define the next level of detail. In the meeting, agree on the top level of the WBS and brainstorm a list of items to include at Level 2.

3) Work collaboratively to identify subsequent levels of WBS detail, by drafting WBS activities and tasks onto a whiteboard or flipchart paper on the walls, or by writing deliverables, activities, and tasks on sticky notes and sticking them to the whiteboard or wall underneath the appropriate top level section of the WBS.

4) Publish the draft WBS to team members and others to sanity check the contents, and identify additional work. Ultimately the WBS should take into account information from:
   - other team members
   - other project managers who have done similar work
   - previous project reviews
   - other appropriate groups
   - expert opinion (MAYBE Linda our Librarian?)

5) Get the feedback and incorporate it into the WBS—this will also take place in our next meeting with the client.

6) Review the updated WBS with the team and determine whether you’re ready to proceed to the next step by assigning resources to the tasks in the WBS (i.e., which pages will individual members check-out and work on?).

How Far to Go: How Much Detail is Enough?

The ultimate goal in breaking the work tasks down is to ensure that all of the work that is needed to meet the project’s objectives is recognized and planned for accurately from the beginning.

The level to which you break down elements of your WBS may result in some tasks having less detail and longer duration, if the work in that area is clearly understood and represents well-known work in which the team is experienced and successful.

- **One owner per task:** The tasks must be defined such that they can be assigned to one person who will be doing that work.

- **Clear measurable deliverable with measurement specified:** The tasks must be defined such that the task owner can be given completion criteria that are clear and measurable.

- **Small enough task duration for tracking:** Task duration at lowest level should be less than 5% of total project time, to ensure visibility into task progress, at a small enough resolution to recognize quickly if the project is off track (e.g. 2 weeks if 1 year; 2 days if 2 months).

Greater levels of detail are generally required for projects which are:

- larger
- more risky
- dissimilar to past projects
- difficult to define (susceptible to change)
- performed by internal work groups
- planned for the near future
The WBS will continue to be updated during the Plan and Schedule development process; generally as the process goes forward, additional tasks come to light and must be incorporated into the WBS.

The checklist below will help the team know that a WBS has been created that forms a sound basis for the project’s schedule going forward.

<table>
<thead>
<tr>
<th>WBS Completion Checklist (ANSWER ALL BELOW)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Appropriate level of detail</strong>: Continue to break the work down until a task list is developed which meets the following criteria:</td>
</tr>
<tr>
<td>- one (and only one) owner can be assigned to each of the lowest level tasks</td>
</tr>
<tr>
<td>- clearly defined outputs are evident for each task</td>
</tr>
<tr>
<td>- quality can be monitored through performance criteria associated with each output</td>
</tr>
<tr>
<td>- the tasks communicate the work to be accomplished to the person who is accountable</td>
</tr>
<tr>
<td>- the likelihood that a task is omitted or work flow forgotten is minimized</td>
</tr>
<tr>
<td>- each task is well enough defined and small enough so that estimates of duration are credible</td>
</tr>
<tr>
<td>- the project is broken down to the level at which you want to track</td>
</tr>
</tbody>
</table>
Below is what you will be delivering to the client during your next in-person meeting.

Website Development WBS

1. Website Project

1.1. Design Tasks
   1.1.1. Web User Interface
      1.1.1.1. Functional Specifications
      1.1.1.2. Technical Specifications
   1.1.2. Google App Engine
      1.1.2.1. Engine setup Instructions
      1.1.2.2. Engine classes and components
      1.1.2.3. Engine project file structure
   1.1.3. Interfaces
      1.1.3.1. Tech specification interfaces
   1.1.4. Reports
      1.1.4.1. Functional Specifications
      1.1.4.1.1. Collect User Requirements
      1.1.4.1.1.1. Define Data elements
      1.1.4.1.1.2. Define Frequency
      1.1.4.1.1.3. Define Audience

1.2. Development
   1.2.1. Web Front End
      1.2.1.1. Web Page design/functionality
   1.2.2. Interfaces
      1.2.2.1. Incorporating interfaces
   1.2.3. Reports
      1.2.3.1. Code Reports

1.3. APPENDICES
   1.3.1. Appendix A: Functionality Table
   1.3.2. Appendix B: Current Site Map
   1.3.3. Appendix C: Proposed Site Map
   1.3.4. Appendix D: Team Spreadsheet (Team Members' Name, Phase Manager Position, and Contact Info)
1.1 Design Task

Task given to team for web site design and layout. This provides an overview of the website's design and how it process.

1.1.1 Web User Interface

1.1.1.1 Functional Specifications

The web user interface is the application front end from which a user can submit files to be processed. The main function of the user interface is to allow a file to be uploaded to the website and to be processed. Once uploaded, the server will handle necessary processing and will display the results to the user in the same location as the upload form. This is the main specification for the web user interface.

See Web interface for uploading below

1.1.1.2 Technical Specifications

The web user interface will be hosted on the wasc website on Oucampus. The back end of the program will be hosted on the google app engine. The upload form will be a portal through iframe directly to the google app engine. This upload form will be embedded in a page on the wasc
website. Data handling and processing will not be done on the wasc website.

Example of possible upload form:

```javascript
function upld()
{
    document.getElementById('file_upload_form').onsubmit=function()
    {
        document.getElementById('file_upload_form').target = 'iFrame_name';
    }
}
window.onload=upld;
```

1.1.2 Google App Engine

1.1.2.1 Engine Setup Instructions

1.1.2.1.1 Project Prerequisites:

a. Download and install Google App Engine SDK for Java from https://developers.google.com/appengine/downloads#Google_App_Engine_SDK_for_Java
b. Download and install Eclipse IDE
c. Download and install git

1.1.2.1.2 Clone WASC-Engine Project:

Git clone https://github.com/tstullich/WASC-Engine.git, or navigate to https://github.com/tstullich/WASC-Engine and click on ZIP to download without git

1.1.2.1.3 Test Project:

a. Import WASC-Engine into Eclipse: file → import → existing projects into workspace
b. Make sure you are using the package: com.sjsu.wasc engine and not the default package
c. Run on gae: run → run as: Web Application
d. Wait for the project to compile and display information in the console
e. Navigate web browser to localhost:8888 and ensure the page is visible

1.1.2.2 Engine Classes and Components:

1.1.2.2.1 WASC_EngineServlet.java:
a. The main class for the project. It starts a server that listens on port 8888 for an HTTP Request and listens for an attempt to upload a PDF file. It will then use the PDFExtract class to convert the given PDF file to an ArrayList containing the list of words given in the PDF input. The ArrayList is then fed into the KeywordAnalyzer class for analysis and a final score will be calculated and displayed on the webpage.
b. Author: Tim Stullich
c. Methods:
   ```java
   public void doGet(HttpServletRequest req, HttpServletResponse resp)
   throws IOException:
   The main method: handles listening for a connection request and returns data in the form of a JSON stream.
   ```
d. How to write to web page:
   Use the HTTPServletResponse resp to write back to the browser. Call resp.getWriter().println("Data here");

1.1.2.2.2 PDFExtract.java:

a. Responsible for parsing a PDF file and returning an ArrayList containing the list of words in the file
b. Author: Akshat Kukreti
c. Methods:
   ```java
   public static ArrayList<String> convertToText(String filename)
   throws DocumentException, IOException
   1. Input: filename of PDF file
   2. Output: ArrayList containing list of words
   ```

1.1.2.2.3 KeywordAnalyzer.java:

a. A KeywordAnalyzer that is capable of performing keyword statistical analysis on text. Calling parseText multiple times without a reset will continuously build word counts. Also, multiple keyword files can be read into the tree.
b. Author: Michael Riha
c. Public Methods:
   ```java
   public KeywordAnalyzer()
   public void reset()
   public void purgeKeywords()
   public void readKeywordFile(String filename) throws FileNotFoundException, IOException
   public void parseText(ArrayList<String> words)
   public double[] calculateScores()
   public static double calculateScore(int weightOneCount, int weightTwoCount, int totalWordCount)
   public int getKeywordOccurrences(String keyword)
   public SortedSet<String>[] getKeywordsUsed()
   public double[] getScores()
   public int[] getWordCounts()
   public int getTotalWords()
   ```
1.1.2.2.4 PrefixTree.java:

a. A PrefixTree to store keywords. Keywords have their corresponding weights, rubric, and how many times they have been found stored in the leaf nodes
b. Author: Michael Riha
c. Public Methods:
   public PrefixTree()
   public boolean add(String word, int weight, int rubric)
   public Node find(String word)
   public Node findNoIncrement(String word)
   public void reset()
d. Inner classes:
   public class Node

1.1.2.3 Engine Project File structure

1.1.2.3.1 WASC-Engine: project root
   src: all the sources for the classes go here
   package: com.sjsu.wascengine
   App Engine SDK
   JRE system library
   lib
   gson
   itextpdf
   war
   WEB-INF: this is the root directory for the website
   appengine-generated
   lib
   testfiles: contains test PDF inputs
   index.html: this is the main webpage

1.1.3 Interfaces

1.1.3.1 Tech Specification Interfaces

As of the current status of the program, we expect to use a JSON interface to send result data back to the wasc front end user interface. Information for using JSON can be found at http://www.json.org/javadoc/org/json/JSONObject.html

1.1.4.Reports
Reports would include following listed: These reports would be available in their respective pages of the section of the WASC Essay No.2 project website as seen below:

- 5 rubrics/ proficiency
- Searched keywords
- Their count and points based on the 5 proficiency
- Phase I documentations
- Gantt Chart
- Phase II documentations
- WBS
- Phase III-V development plan
- Java/Tool documentations

5 rubrics/ proficiency would be downloadable in pdf form in Documentation page of website under WASC Essay no.2 project (See appendices)
Searched keywords

Keywords given by client, see Keyword Rubrics
Searched keywords example, after a scanned pdf programming plan see below

Their count and points based on the 5 proficiency, search keywords is shown, counts, as well as the document/pdf score

| Rubric 1 | Weight 1 keywords used: 8  
| apply 1 issues 7  
| Weight 2 keywords used: 9  
| analysis 3 evaluated 2 evaluation 2 evaluations 2 reflect 1  
| synthesized 1  
| Rubric 2 | Weight 1 keywords used: 7  
| context 2 organized 1 summary 4  
| Weight 2 keywords used: 14  
| information 2 research 12  
| Rubric 3 | Weight 1 keywords used: 2  
| paper 1 thesis 1  
| Weight 2 keywords used: 4  
| writing 4  
| Rubric 4 | Weight 1 keywords used: 0  
| Weight 2 keywords used: 1  
| present 1  
| Rubric 5 | Weight 1 keywords used: 0  
| Weight 2 keywords used: 7  
| data 7  

Scores: 4.27 4.81 1.35 0.17 1.18 2.36

Phase I documentations (Project Analysis), Gannt Chart, Phase II documentations (Design Worksheet), WBS, Phase III-V development plan, and Java/Tool documentations would be downloadable as seen below
*note: Java/Tool documentation can also be seen inside java source file with their respective authors

1.1.4.1 Functional Specifications

a. Manual functional and readable document, PDF (only)
b. Program Plan Scanner page (inside an iframe) upload a file using submit button results shown at same page
c. Limitations no graphs and no use of google analytics
Program Plan Scanner

Instructions
Please submit departmental program plan(s) using the form below and click submit.

File: [Choose File] No file chosen
Submit

Thank you for using our Program Plan Scanner.

Click here to use again
1.1.4.1.1 Collect User Requirements

The data elements are the key words extracted from the 5 rubrics which are used to evaluate the documents. The keywords are categorized into 3 levels with 3 different weights. Let's take oral communication rubric for example. This is oral communication rubric which we obtained from our client Dr. Todd.

1.1.4.1.1.2 Define Frequency

Frequency is the amount of times a keyword appeared in the document. If there are 3 “deliver”, 5 “present” and 2 “support” in the document, the frequencies of “deliver”, “present” and “support” are 3, 5 and 2, respectively.

1.1.4.1.1.3 Define Audience

a. WASC administrators
b. Faculty members that hold programming plans from their respective departments.

c. Teams for the next part of this project which requires the documentation of the java program as well as instructions of what has been done in this previous project.

d. All public school institutions
1.2 Development

1.2.1 Web Front End

1.2.1.1 Web Page Design/Functionality

The front end will display a brief prompt asking the user to select a file in pdf format which will then be sent to our Google Application in order to analyze the text.

Website -----> Upload File --- > Google Server (App Engine) --- > Java App ---- > Website ----> Result page

After the application is finished with calculating the necessary data the front end will display this data in a neat column format and provide the user with an option to save the data in a csv (comma separated values) file.

ex. (snippet)
document score : 4.27, 4.81, 1.35 , 0.17, 1.18, 2.36

1.2.2 Interfaces

1.2.2.1 Incorporating Interfaces

An interface might be used to allow for more types of documents to be read into our file (only PDF at the moment) reader in order to process them. These interfaces will be defined within the PrefixTree class and can be implemented depending on how the developer sees fit. This way different files will be able to be read in and processed.

Prefix Tree (snippet)
public boolean add(String word, int weight, int rubric)  
  { ... }  
public Node find(String word)  
  { ... }  
public Node findNoIncrement(String word)  
  { ... }

1.2.3 Reports

1.2.3.1 Code Reports

Code Reports will consist of in-line comments that can be found within the
source code for our application as well as further documentation in
the form of a proper Java Doc file. The links to various documentations is
on WASC site.

WASC
  > Project Documentations
    > Java Docs
    > Phase Docs
1.3 APPENDICES

1.3.1 Appendix A: Functionality Table

<table>
<thead>
<tr>
<th>San Jose State University WASC Projects</th>
<th>About WASC Proj</th>
<th>WASC projects</th>
<th>5 proficiencies</th>
<th>Project Documentations</th>
<th>Program Plan Scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Critical Thinking</td>
<td></td>
<td>Java Application</td>
<td></td>
<td>upload program file and download the results</td>
</tr>
<tr>
<td>goals</td>
<td>Critical Thinking Rubric</td>
<td>Information Literacy</td>
<td>Information Literacy Rubric</td>
<td>Oral Communication</td>
<td>Phases documentation Fall 2012 (phase1, WBS, phase2, Phase3-5)</td>
</tr>
<tr>
<td>Essay No.2</td>
<td>Oral Communication Rubric</td>
<td>Quantitative Literacy</td>
<td>Quantitative Literacy Rubric</td>
<td>Written Communication</td>
<td></td>
</tr>
<tr>
<td>outcome</td>
<td>Written Communication rubric</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lit Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.3.2 Appendix B: Current Site Map

San Jose State University WASC Projects (BEFORE)
Home WASC Projects ([http://dev.sjsu.edu/wascproj](http://dev.sjsu.edu/wascproj))
1.3.3 Appendix C: Proposed Site Map

San Jose State University WASC Projects (AFTER)
Home http://dev.sjsu.edu/wascproj/
SITE MAP

About WASC Proj
Goals
Outcomes

WASC PROJECTS
WASC ESSAY No. 2
LIT REVIEW (see Lit Review Team Documentations)

5 Proficiencies
Critical Thinking
Information Literacy
Oral Communication
Quantitative Literacy
Written Communication

Project Documentation
WASC Essay No.2
Java Scanner
Project Phase Documents
Phase 1 Action Plan
WBS
Phase 2 Design Worksheet
Phase 3 – 5 Development Plan
Lit Review (see Lit Review Team Documentations)

Programing Plan Scanner
1.3.4 Appendix D: Team Spreadsheet (Team Members’ Name, Phase Manager Position, and Contact Info)

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akshat Kukreti</td>
<td>Phase 1 scrum master</td>
<td><a href="mailto:akshatkukreti@gmail.com">akshatkukreti@gmail.com</a></td>
</tr>
<tr>
<td>Charles Long</td>
<td>Phase 5 scrum master</td>
<td><a href="mailto:chars.long@gmail.com">chars.long@gmail.com</a></td>
</tr>
<tr>
<td>Chunbo Tan</td>
<td>Phase 4 scrum master</td>
<td><a href="mailto:chunbo.tan@students.sjsu.edu">chunbo.tan@students.sjsu.edu</a></td>
</tr>
<tr>
<td>Edward Ciotic</td>
<td>Phase 3 scrum master</td>
<td><a href="mailto:ciotic@gmail.com">ciotic@gmail.com</a></td>
</tr>
<tr>
<td>Ezekiel Calubaquib</td>
<td>Phase 5 scrum master</td>
<td><a href="mailto:ezekit@yahoo.com">ezekit@yahoo.com</a></td>
</tr>
<tr>
<td>Michael Keats</td>
<td>Phase 3 scrum master</td>
<td><a href="mailto:michaelkeats@gmail.com">michaelkeats@gmail.com</a></td>
</tr>
<tr>
<td>Michael Riha</td>
<td>Phase 4 scrum master</td>
<td><a href="mailto:rihamichael@gmail.com">rihamichael@gmail.com</a></td>
</tr>
<tr>
<td>Tim Stullich</td>
<td>Phase 3 scrum master</td>
<td><a href="mailto:timstullich@gmail.com">timstullich@gmail.com</a></td>
</tr>
<tr>
<td>Wanzhen Wu</td>
<td>Phase 2 scrum master</td>
<td><a href="mailto:wanzhen719@gmail.com">wanzhen719@gmail.com</a></td>
</tr>
</tbody>
</table>