Foreword

Disclaimer

This document contains sample documentation for Stage A of the IB Comp Sci IA Program Dossier. It was prepared by an IB teacher as a sample for his students. This sample has no "official" status, and was not sanctioned by the IBO.

My Intentions

I will show this sample to my students so they have a clearer idea of what is expected. I hope this example meets the expectations of the examiners, as put forth in the Guide and Teacher Support Materials. My intention was to produce an example that would receive full marks in Stage A (4+4+4), but I cannot guarantee this.

Prototype

Criterion A3 states that "The prototype need not be functional … The intent is to show the user how the system is expected to operate…" I found it useful to create a **functional** prototype of a key part of the system. This very clearly demonstrated to the user some of the key advantages of a computer solution. It made the discussions with the users much easier – I didn't need to work so hard to explain what *would* happen.

Feasibility

The functional prototype has the second advantage that it demonstrates the **feasibility** of the intended solution, specifically:
- that HTML documents can be displayed correctly
- that the user can easily mark a word by double-clicking

I encourage my students to confront some of the tricky programming issues early-on, so they don't spend lots of time planning a solution that they cannot actually implement later.

UML Use Case vs XP User Stories

Extreme Programming (XP) has **User Stories –** these are similar to **Use Cases** in **UML.** I imagine some students will use UML for their designs. I find UML rather daunting for high school students, so I won't be using it. The **XP user stories** seem to fulfill the "systematic method" required in Criterion A1 #4. They also connect easily to user interviews, which helps in some other criteria. My version of **user stories** is not "standard", but I have not found a clear statement about a "standard" way to present these.

Formatting

The formatting is purposely compact, in landscape layout, to make it easier to read on a computer monitor. I don't recommend that students use this layout – it is rather tricky and they needn't waste time trying to duplicate this format. In my experience, IB Comp Sci dossiers waste a lot of paper due to white space, but this is perfectly acceptable. I **do** encourage students to print their program **listings** in landscape mode to accommodate long lines.

Length

I think there are too many pages here for section A, but I could not see a good way to reduce the volume. A few pages are devoted to very large full-screen captures – student projects probably won't contain such large screen captures. And the appendix pages don't count. My intended solution for the problem is perhaps a bit more comprehensive than a typical student's work - a more limited solution would reduce the pages. But I was trying to illustrate what a highly competent student would do to achieve maximum marks. I imagine some students will indeed produce this much (or more).

Permission

ESL Reading Assistant with Vocabulary Support

Investigation and Analysis

The Problem (Overview)

Our international school has many students receiving **ESL** (English as a Second Language) support and instruction. The students receive direct instruction to improve their English, but at the same time they are enrolled in normal classes like Science, Math, and History. They must read standard English language textbooks and worksheets – this is difficult as the students are not yet proficient English speakers.

Existing System

The ESL teachers help the students with their reading. Sometimes they sit with an ESL student and answer questions to help them understand unfamiliar words and complex sentences.

The teachers compile vocabulary lists for specific reading assignments. For example, if all the students in grade 8 read a worksheet about the heart, an ESL teacher may take the worksheet and write a supplementary vocabulary list for that worksheet. This is especially useful if the same worksheet can be re-used each year.

The students also use standard dictionaries (paper) and electronic translating-dictionaries for their own language. The ESL programme currently has many Korean students, so they use an English-Korean translating dictionary.

Of course the students also receive help from their classroom teachers. One of the humanities teachers cooperates closely with Mr S. to build up the stockpile of custom vocabulary for re-used reading assignments. Some teachers have regular (weekly) vocabulary lists and quizzes.

Collection of materials – both reading assignments and vocabulary supplements – is managed by individual teachers. There is no central collection of materials – e.g. no database or central repository of materials.

One of the ESL teachers – Mr S. – has an extensive collection of vocabulary sheets. These were word-processed and collected electronically. When a student needs to read a worksheet, the teacher **prints** the matching vocabulary notes and gives them to the student.

Some students use web-sites for translation and reference. These are attractive because they are free and reasonably quick - but they are inconvenient as the students don't always have a computer available when they are doing their reading. Although our school has many computers available (and most students have computers and Internet connections at home), students are often working in classrooms with no access to computers.

Despite availability of computers, most of the reading materials are distributed on paper. There is relatively little use of web-sites and other electronic distribution systems.

Sample Data

Here are some samples (shortened) of reading assignments and vocabulary lists:

Simultaneous Inequalities *(grade 9 mathematics)*

Businessman face the problem of making choices under restrictions. In a real business, there are lots and lots of variables, and lots and lots of restrictions. We will look at a very simple version, with only 2 variables and 2 restrictions.

Problem
A delivery truck delivers two types of cargo: TV sets and computers..

Each TV set weighs 25 kg, and a computer weighs 10 kg.
The maximum weight of cargo is 500 kg.

Each TV set costs 500 EU, and each computer costs 800 EU.
The maximum value of cargo must be under 25000 EU, for insurance purposes.

A TV can be sold for 150 EU profit, and a computer earns 200 EU profit.

The question is: what is the best number of TVs and computers to deliver, in order to earn the maximum profit?
…..

Vocabulary:

simultaneous
when two things happen at the same time –
in math, when two equations need answers at the "same time"

variable
a letter that represents an unknown number
…..

What Did Columbus and his Men Eat? *(grade 8 Humanities)*

Columbus sailed from Palos de la Frontera on 3 August, 1492. Let us look at the first voyage and the victuals embarked on the three vessels, the Nina, Pinta and Santa Maria. The first problem was to obtain supplies of food, wine and water. At the Canary islands they picked up fresh water, wood and the famous Gomera goat cheese.

Columbus' first voyage had the best victuals (and enough to last a year), not the case in his other voyages.

The menu for Spanish seamen consisted of water, vinegar, wine, olive oil, molasses, cheese, honey, raisins, rice, garlic, almonds, sea biscuits (hardtack), dry legumes such as chickpeas, lentils, beans, salted and barreled sardines, anchovies, dry salt cod and pickled or salted meats (beef and pork), salted flour. The olive oil and perhaps olives were stored in earthenware jugs. All other provisions were stored in wooden casks which, according to some reports, were of cheap and faulty construction permitting the preserving brine to leak out of the meat casks and moisture to invade the casks of dry provisions. All were stored in the hold, the driest section of which was normally reserved for those casks carrying dry provisions. A cooper (barrel maker) was responsible for keeping the casks tight, an almost impossible challenge.

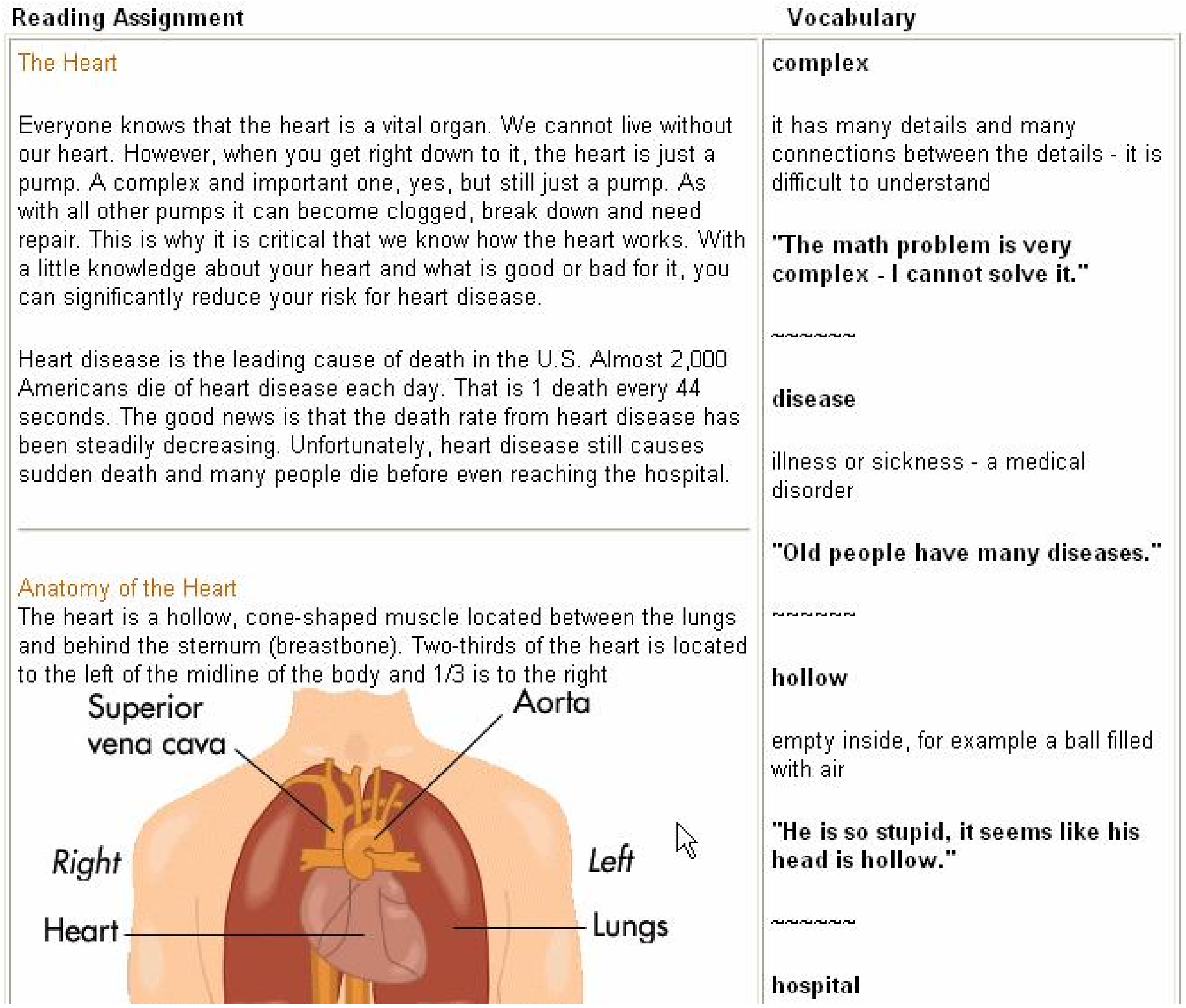
Food, mostly boiled, was served in a large communal wooden bowl. It consisted of poorly cooked meat with bones in it, the sailors attacking it with fervor, picking it with their fingers as they had no forks or spoons.
….

Vocabulary:

victuals	– food
molasses	– a sweep syrup made from sugar cane
legumes	- beans
pickled	– preserved by soaking in liquid (e.g. vinegar and salt)

…..

Another sample reading assignment and vocabulary list
(grade 7 science) – copied from **http://www.howstuffworks.com**



Reading Assignment

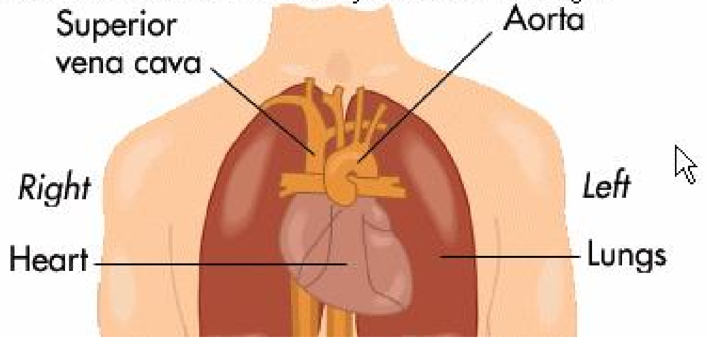
The Heart

Everyone knows that the heart is a vital organ. We cannot live without our heart. However, when you get right down to it, the heart is just a pump. A complex and important one, yes, but still just a pump. As with all other pumps it can become clogged, break down and need repair. This is why it is critical that we know how the heart works. With a little knowledge about your heart and what is good or bad for it, you can significantly reduce your risk for heart disease.

Heart disease is the leading cause of death in the U.S. Almost 2,000 Americans die of heart disease each day. That is 1 death every 44 seconds. The good news is that the death rate from heart disease has been steadily decreasing. Unfortunately, heart disease still causes sudden death and many people die before even reaching the hospital.

Anatomy of the Heart

The heart is a hollow, cone-shaped muscle located between the lungs and behind the sternum (breastbone). Two-thirds of the heart is located to the left of the midline of the body and 1/3 is to the right



Vocabulary

complex

it has many details and many connections between the details - it is difficult to understand

"The math problem is very complex - I cannot solve it."

~~~~~~~

disease

illness or sickness - a medical disorder

"Old people have many diseases."

~~~~~~~

hollow

empty inside, for example a ball filled with air

"He is so stupid, it seems like his head is hollow."

~~~~~~~

hospital
~~~~~~~

Quantity of Data

The system has two kinds of data:
- **reading** assignments
- **vocabulary** lists

The **reading** assignments comprise roughly 1 page per day per subject, or roughly 800 pages per year for the 4 "major" subjects – Math, Science, Humanities, English. This amount exists for each of 4 grades (6-9), so a total of approximately 3000 pages. The majority of the reading assignments are in textbooks, so conversion to electronic format would be difficult – a scanner might make this possible. Something like 20% of the reading is worksheets that already exist in electronic form.

Not every reading assignment requires a custom **vocabulary** sheet. Some sheets already exist – the ESL teachers would like to continue creating more of these. Something like 5% of the reading material might have vocabulary sheets.

The total storage requirement would be for a total of several thousand pages of reading assignments, plus several hundred pages of vocabulary supplements.

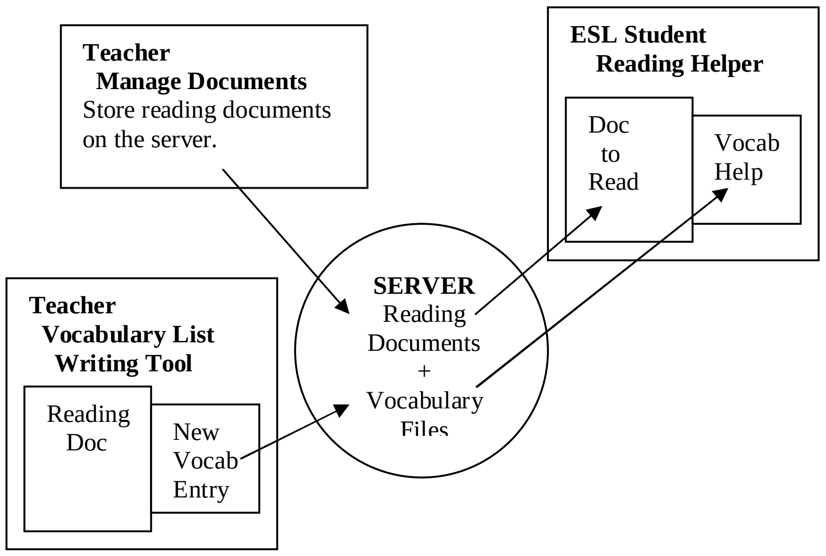
Initial Ideas from the Intended User

Various informal conversations with Mr S led to the idea for this project. Mr S has various wishes:

- easily and quickly increase the vocabulary support materials for the ESL students
- make it easier for ESL students to do their reading and use the vocabulary materials
- store documents centrally (and electronically) for easy retrieval and management
- share documents easily and effectively with colleagues
- automate some tasks like typing vocabulary lists

Initial System Design

This initial design incorporates the ideas from Mr S.



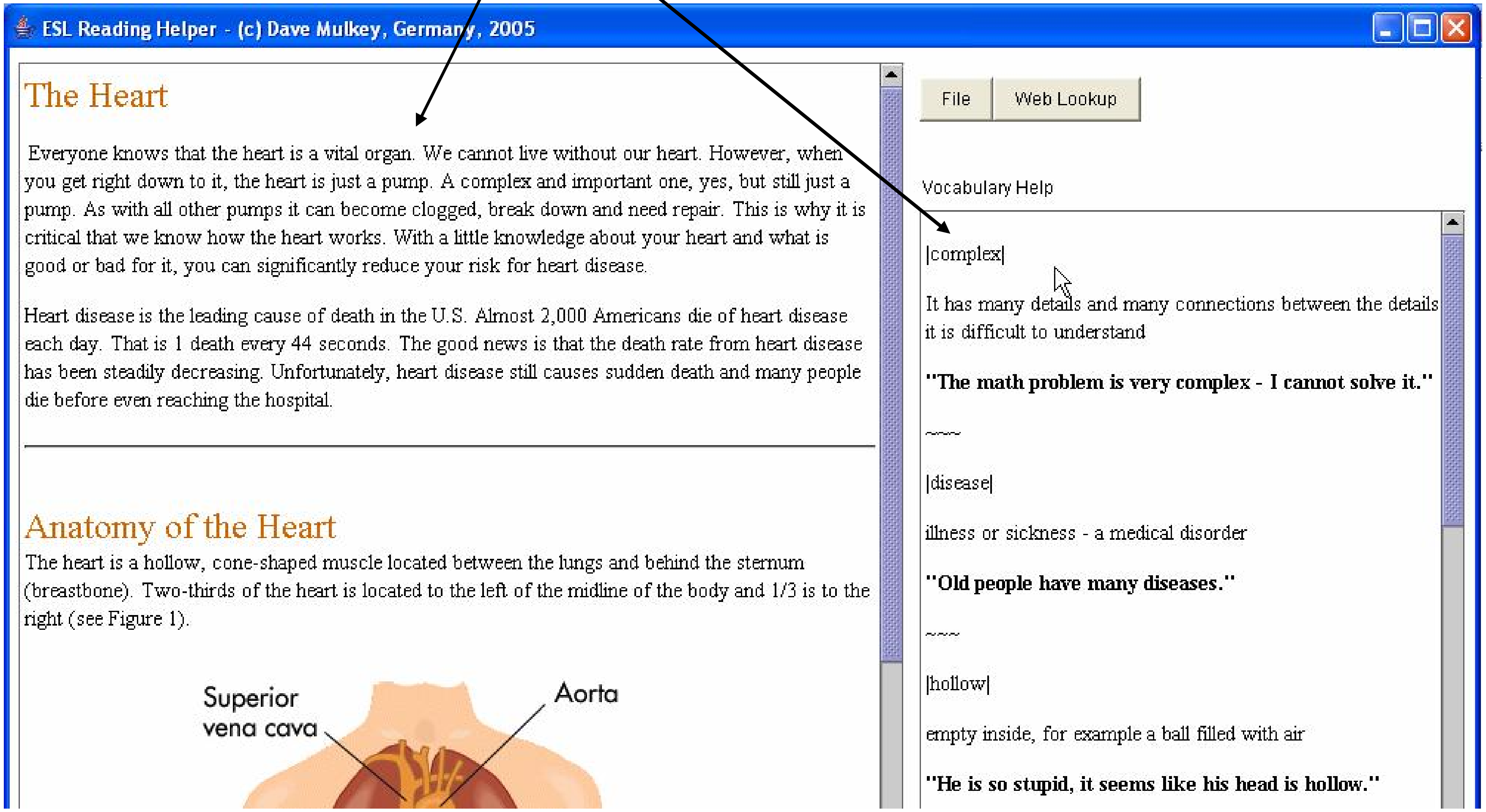
Feasibility Prototype

After preliminary discussions with Mr S, I decided it would be best to show him a **prototype** before engaging in more detailed discussions. He liked the idea of the students retrieving documents from a server, reading them on the screen, and getting automatic vocabulary help by clicking on an unfamiliar word, so these ideas formed the basis for the prototype program.

I was anxious to create a **functional** prototype to prove the **feasibility** of creating the **Reading Helper,** so that it responds with help when the student clicks on a word. If this were not actually possible, the whole concept wouldn't make much sense.

Sample output from the prototype is shown below. This is similar to the presentation made to Mr S. The prototype was written in Java – a listing of the program is included in the appendix.

The user chooses a file (using [File]), and it displays on the left.
The entire vocabulary list appears automatically on the right.



The Heart

Everyone knows that the heart is a vital organ. We cannot live without our heart. However, when you get right down to it, the heart is just a pump. A complex and important one, yes, but still just a pump. As with all other pumps it can become clogged, break down and need repair. This is why it is critical that we know how the heart works. With a little knowledge about your heart and what is good or bad for it, you can significantly reduce your risk for heart disease.

Heart disease is the leading cause of death in the U.S. Almost 2,000 Americans die of heart disease each day. That is 1 death every 44 seconds. The good news is that the death rate from heart disease has been steadily decreasing. Unfortunately, heart disease still causes sudden death and many people die before even reaching the hospital.

Anatomy of the Heart

The heart is a hollow, cone-shaped muscle located between the lungs and behind the sternum (breastbone). Two-thirds of the heart is located to the left of the midline of the body and 1/3 is to the right (see Figure 1).

|complex|

It has many details and many connections between the details
it is difficult to understand

"The math problem is very complex - I cannot solve it."

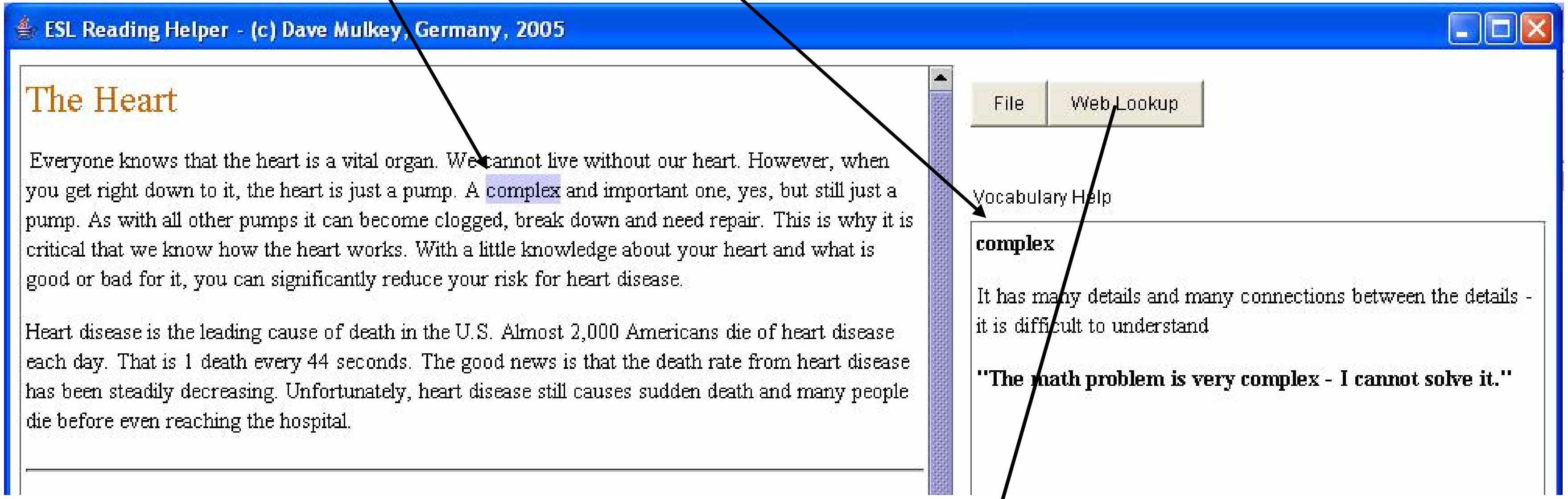
~~~

|disease|

illness or sickness - a medical disorder

"Old people have many diseases."

~~~

|hollow|

empty inside, for example a ball filled with air

"He is so stupid, it seems like his head is hollow."

The user double-clicks on "complex" and the vocabulary entry appears at the right.



The Heart

Everyone knows that the heart is a vital organ. We cannot live without our heart. However, when you get right down to it, the heart is just a pump. A complex and important one, yes, but still just a pump. As with all other pumps it can become clogged, break down and need repair. This is why it is critical that we know how the heart works. With a little knowledge about your heart and what is good or bad for it, you can significantly reduce your risk for heart disease.

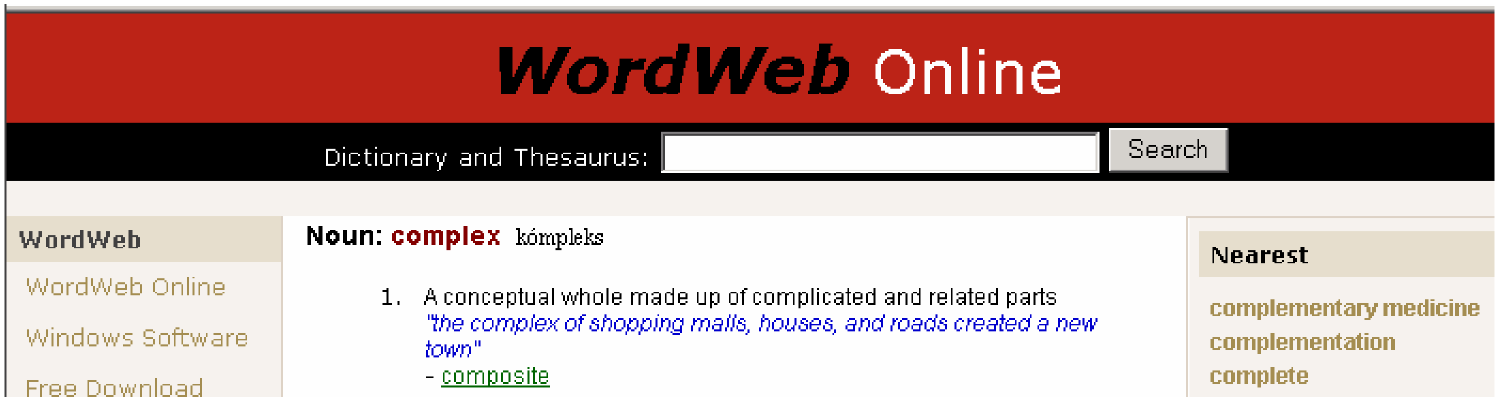
Heart disease is the leading cause of death in the U.S. Almost 2,000 Americans die of heart disease each day. That is 1 death every 44 seconds. The good news is that the death rate from heart disease has been steadily decreasing. Unfortunately, heart disease still causes sudden death and many people die before even reaching the hospital.

complex

It has many details and many connections between the details - it is difficult to understand

"The math problem is very complex - I cannot solve it."

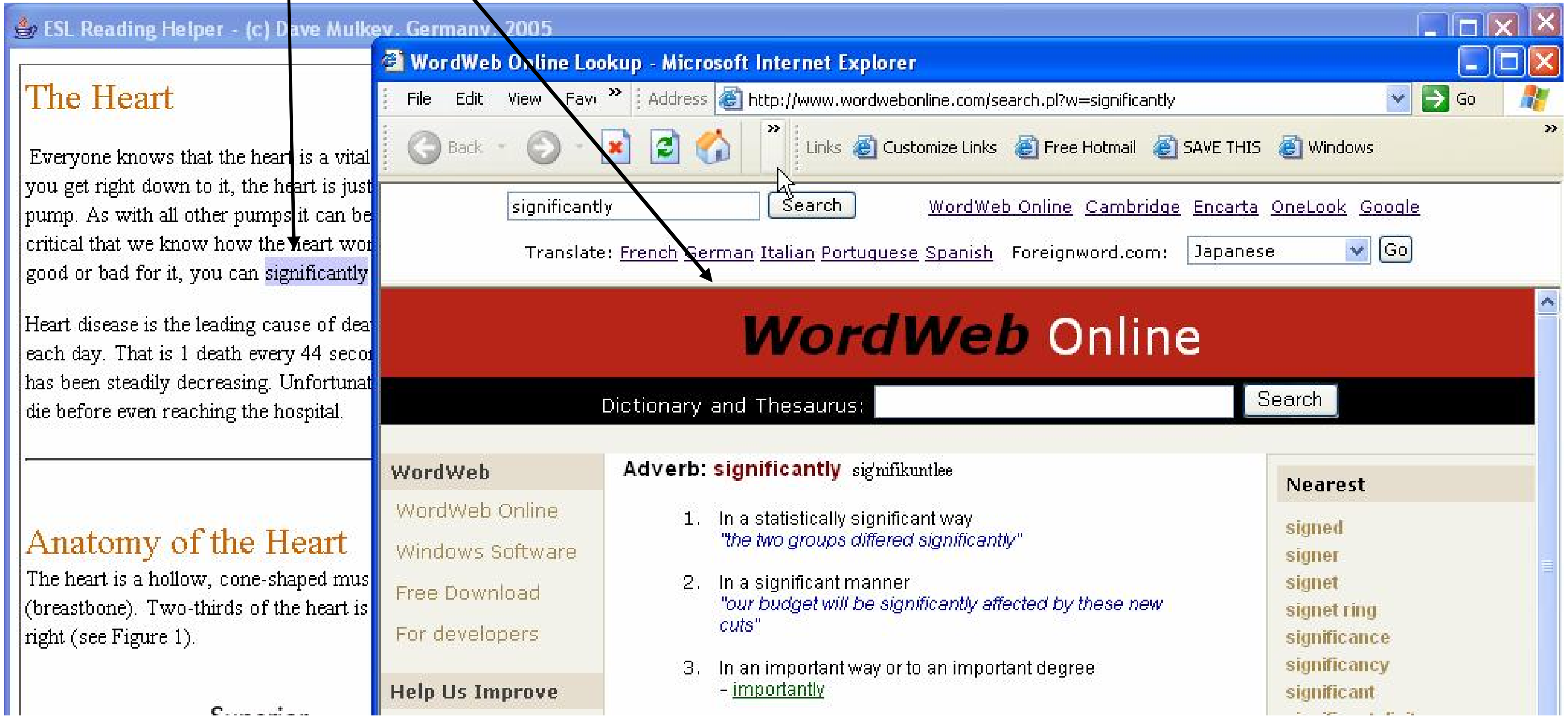
The user can click on [Web Lookup] to get a *standard* definition.



Noun: complex kómpleks

1. A conceptual whole made up of complicated and related parts
   "the complex of shopping malls, houses, and roads created a new town"
   - composite

If the user clicks on a word that is **not** in the vocabulary help-file, the program automatically shows the web-site definition.


ESL Reading Helper - (c) Dave Mulkey, Germany, 2005
WordWeb Online Lookup - Microsoft Internet Explorer
File Edit View Favı
Address http://www.wordwebonline.com/search.pl?w=significantly
Go
Back
Links Customize Links Free Hotmail SAVE THIS Windows
significantly
Search
WordWeb Online Cambridge Encarta OneLook Google
Translate: French German Italian Portuguese Spanish Foreignword.com: Japanese Go
WordWeb Online
Dictionary and Thesaurus: Search
WordWeb
WordWeb Online
Windows Software
Free Download
For developers
Help Us Improve
Nearest
signed
signer
signet
signet ring
significance
significancy
significant

The Heart

Everyone knows that the heart is a vital
you get right down to it, the heart is just
pump. As with all other pumps it can be
critical that we know how the heart wor
good or bad for it, you can significantly

Heart disease is the leading cause of dea
each day. That is 1 death every 44 secon
has been steadily decreasing. Unfortunat
die before even reaching the hospital.

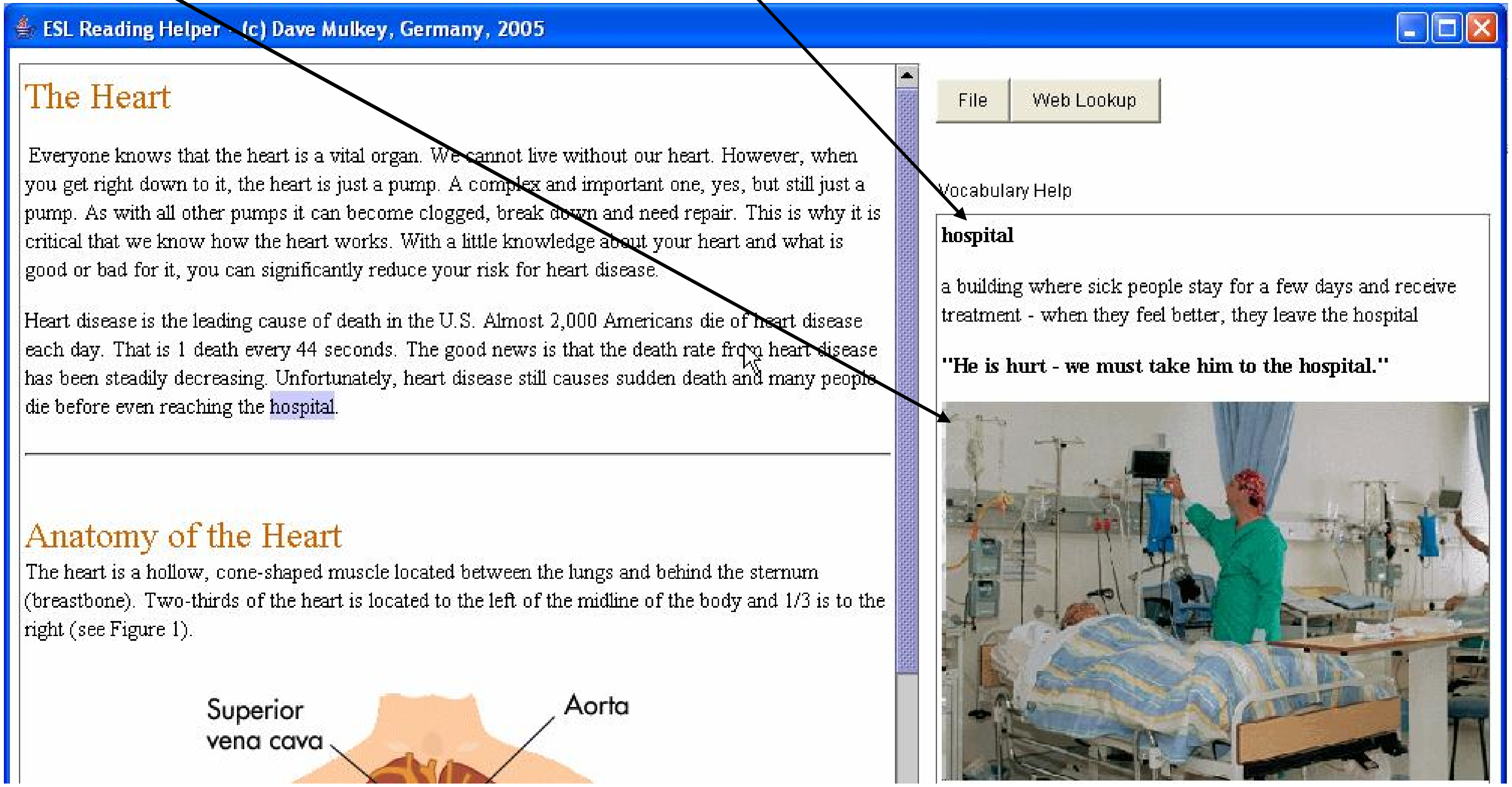
Anatomy of the Heart

The heart is a hollow, cone-shaped mus
(breastbone). Two-thirds of the heart is
right (see Figure 1).

Adverb: significantly sig'nifikuntlee

1. In a statistically significant way
   "the two groups differed significantly"
2. In a significant manner
   "our budget will be significantly affected by these new cuts"
3. In an important way or to an important degree
   - importantly

Both the reading window (left) and the vocabulary window (right) display HTML correctly, so the vocabulary list can display formatting and graphics. (Both display boxes are **JEditorPane** controls.)



ESL Reading Helper - (c) Dave Mulkey, Germany, 2005

The Heart

Everyone knows that the heart is a vital organ. We cannot live without our heart. However, when you get right down to it, the heart is just a pump. A complex and important one, yes, but still just a pump. As with all other pumps it can become clogged, break down and need repair. This is why it is critical that we know how the heart works. With a little knowledge about your heart and what is good or bad for it, you can significantly reduce your risk for heart disease.

Heart disease is the leading cause of death in the U.S. Almost 2,000 Americans die of heart disease each day. That is 1 death every 44 seconds. The good news is that the death rate from heart disease has been steadily decreasing. Unfortunately, heart disease still causes sudden death and many people die before even reaching the hospital.

Anatomy of the Heart

The heart is a hollow, cone-shaped muscle located between the lungs and behind the sternum (breastbone). Two-thirds of the heart is located to the left of the midline of the body and 1/3 is to the right (see Figure 1).

Vocabulary Help

hospital

a building where sick people stay for a few days and receive treatment - when they feel better, they leave the hospital

"He is hurt - we must take him to the hospital."

Discussion with the Intended User

Mr S was very pleased with the prototype for the student Reading-Helper. He said it seemed quite easy to use. He liked the web-site link for looking up words not in a vocabulary list from the teacher. He asked whether we could find out which words students clicked on, so we would know which words needed more/better explanations. He also asked whether he could "track" completion of assignments.

Reader Interface

We agreed that the basic design of the reading interface was good, but a few more items needed to be added.

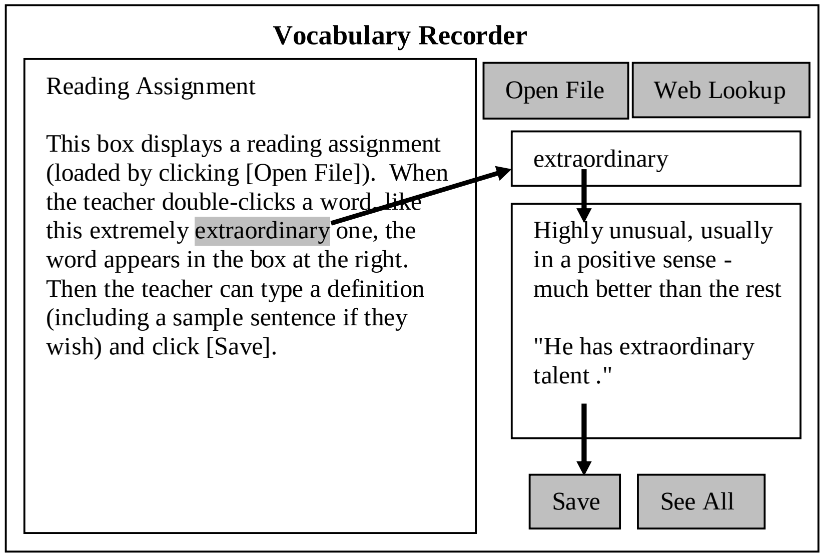
- a student ID **login** would be required, so that the program could record the files (and dates) that students read, and teachers could check that students do assigned reading
- it would be useful to **record** the vocabulary words that students **click** on, so teachers could see which words were clicked often and thus need explanations
- the [File] button should be replaced by a **list** of current reading **assignments** (in a drop-down box)
- a **menu**(s) should be added with things like "Open File", "Open student record", "Log-off", etc.
- some other menus might also be useful, but those should be designed to support the goals (later)

Further Interfaces – Mock-up Prototype

Interfaces are also needed for the teacher modules - writing vocabulary lists and reviewing student records. I sketched these briefly in consultation with Mr S. Then I drew them more clearly with MS Word. The results are shown below.

Vocabulary Recorder Interface

This is similar to the reader interface, with the vocabulary-help box replaced by vocabulary-input boxes. When the teacher clicks on a word, the word appears in the box on the right, the teacher types a definition, and presses [Save] to add this to the vocabulary file for this document. (Arrows illustrate the order of clicks and typing).


Vocabulary Recorder

Reading Assignment

This box displays a reading assignment (loaded by clicking [Open File]). When the teacher double-clicks a word, like this extremely extraordinary one, the word appears in the box at the right. Then the teacher can type a definition (including a sample sentence if they wish) and click [Save].

extraordinary

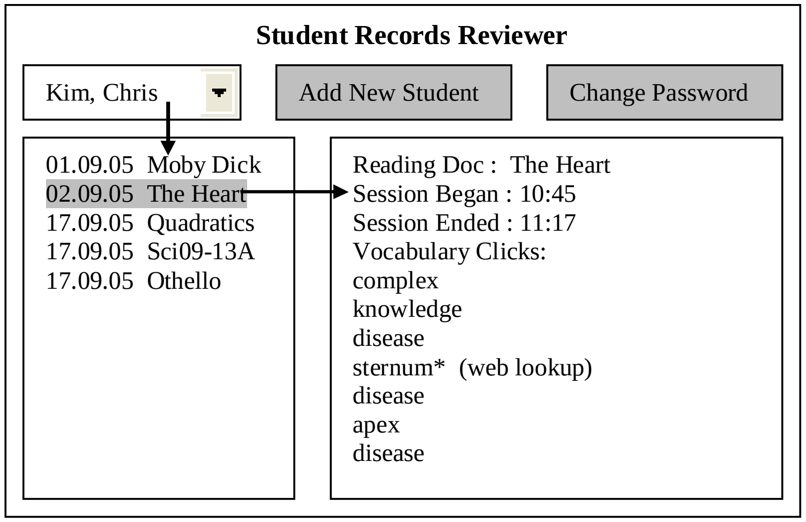
Highly unusual, usually in a positive sense - much better than the rest

"He has extraordinary talent ."

The word and definition are saved in a formatted text-file. The teacher can read this file (in a pop-up window) by clicking on [See All]. Later, this file is converted to an HTML file which can be edited with a standard HTML editor (described below).

Student Records Reviewer

This interface allows the teacher to review student records - to see what they have been reading. Each reading session will have a record of the file-name and all the vocabulary clicks. This interface can also change the students password (to prevent students from using each-others' accounts). The student's name is picked from a drop-down list. Then the teacher can pick a session from the Sessions list and read the information about that session. Arrows show the flow of control (order of clicks).


Student Records Reviewer

Kim, Chris | Add New Student | Change Password

01.09.05 Moby Dick
02.09.05 The Heart
17.09.05 Quadratics
17.09.05 Sci09-13A
17.09.05 Othello

Reading Doc : The Heart
Session Began : 10:45
Session Ended : 11:17
Vocabulary Clicks:
complex
knowledge
disease
sternum* (web lookup)
disease
apex
disease

File Management

After some discussion, we agreed that it wouldn't be necessary to program a document-management interface. Teachers can easily create folders and store documents on the server without a special program.

Further Discussion with the User

Mr S had a variety of questions related to the prototypes. Some are summarized here - there were lots of little questions about colors and other small details. My initial answers are also shown (marked with @).

Some of the questions were suggested by me – things like preventing duplicate entries. Mr S agreed that these were important issues.

Student Reader

- *If the student clicks twice on the same word, will that be recorded?*
  @- Yes, if you want that.

- *How should we record the list of reading assignments for each student?*
  @- Ah, we should add that to the Student Records Reviewer interface. I need to think about that and redesign the interface.

- *Can we include an interactive vocabulary quiz?*
  @- No, I don't think I'll have time to do that, but maybe in a later version.

- *Will there be a [back] button like a normal browser?*
  @- No, I don't know how to do a [back] button, but links will work - connected documents could link to each other

- *What about plurals and other endings? We don't want to write extra entries for these.*
  @- That sounds complicated – I don't know if I can do it.

Vocabulary Recorder

- *Can we use bold and italics in the definitions? How? My colleagues won't be able to type HTML tags.*
  @- You can type HTML tags like <b>bold</b>. The other teachers will need to get along without bold and italics, but maybe I can think of a simple solution.

- *Can we write extra versions of the word in the word-box? Like "geometry" and "geometric"? Or do we need to make 2 separate entries? Can we at least copy the definition?*
  @- It's simplest to make 2 entries, but I'll think about it. Maybe a cleverer search algorithm can solve that problem. I already took care of plurals in the prototype, but it doesn't work perfectly yet.

- *You had a picture of a hospital in the prototype. How can we do that ?*
  @- Write an HTML tag <img ...> It isn't too difficult. Of course you'll need to get a picture, e.g. from Google.

- *Do we need to write a separate list of vocabulary for every document, or can we re-use it for other documents?*
  @- We can add an extra utility for merging vocabulary files. Maybe we can add that to the vocabulary recorder, as a separate function. This is an important issue and will require further thought.

Student Records Reviewer

- *My colleagues and I share some students. What happens if a student name gets added more than once?*
  @- I'll make the program reject duplicate names. If there really are duplicate names, you can add numbers (Kim2).

HTML and File Formats

Mr S was concerned about using HTML as the basic file format. His colleagues only know how to use MS Word. I checked the file formats supported by **JEditorPane**. It displays RTF and HTML and TXT, but not DOC or PDF. He said that RTF would probably work best for the other teachers, but he was happy with HTML.

Using HTML is pretty central to the whole concept - both for document display and for searching for a vocabulary entry. The vocabulary files must be structured, including **delimiters** (markers) marking words with |bars| and the ends of definitions with tildas ~~~. In the prototype it looks like this:

```
|complex|
   it has many details, and many connections
   between the details – it is difficult to understand
~~~
|hollow|
   ……..
```

This is embedded in an HTML file for easy editing. It can still be edited by the user **and** decoded by the program, as demonstrated in the feasibility prototype. It could be done differently, but this method maintains flexibility and reusability for the future. Someone said "XML is the future". I don't know anything about XML, so I decided to stick to HTML.

Standard Tools

Standard editors – especially MS Word for .rtf files and a WYSIWYG editor for HTML – can be used to edit reading files and vocabulary files.

User Stories

The boxes below summarize **user stories** based on the investigation above. User stores are an **Extreme Programming** concept similar to **UML Use Cases**.

<Angle brackets> indicate significant **user actions**.
Asterisks indicate **automated computer processes**.
(Round parentheses) indicate **data-storage** (files).

Student Stories

Task: Read an Assignment **User : Student**

Interface : Reading-Helper
Input : readable document and vocabulary list from (server)
Output : document is *displayed* in a scrollable box
Actions : user <scrolls> up and down, <clicks> on vocab words
Automation : interface *displays* HTML or RTF doc properly,
records user actions in a (log-file)

Task: Look-up Vocabulary Word **User : Student**

Interface : Reading-Helper
Input : readable document and vocabulary list from (server)
Actions : <click> on vocabulary words
Automation : *highlight* clicked word, then *search* for the word in the vocabulary list or *connect* to web-site dictionary
Output : *display* vocabulary word with definition,
record click action in user (log-file)

Task: Log-in **User : Student**
Interface : Pop-up dialog
Actions : <Type> name and password
Automation : *Validate* user name +password – reject if incorrect

Teacher Stories

Task : Write Vocabulary List **User : Teacher**

Interface : Vocabulary-Recorder
Input : reading document
Output : *display* chosen document
Actions : <click> on a word in the document,
<type> a definition for the word
Automation : *display* clicked word in vocabulary entry area
accept definition, *store* new entry in (vocab file)

Task : Review Student Reading Sessions **User : Teacher**

Interface : Student-Records-Reviewer
Input : Student names from (log-in file)
Student session records from (sessions file)
Output : *display* list of student names
display reading session records = times + vocab clicks
Actions : <click> on a student name to *display* list of sessions
<click> on a session to *display* details of that session
Automation : *search* for a student name
retrieve sessions for that student

Task : Add new student **User : Teacher**
Interface : Student-Records-Reviewer
Input : <type> name + password for new student
Automation : <add> new name + password into the (log-in file)
<reject> duplicate student names

Task : Change student password **User : Teacher**
Interface : Student-Records-Reviewer
Input : Student names from (log-in file)
Actions : <choose> a student name , <type> new password
Automation : <find> student name in (log-in file)
<change> password in (log-in file)

More User Stories

Task : Collect Documents to Read **User : Teacher**

Input : Documents can be collected in several ways:
<copy> an existing document
<convert> from MS word .DOC format to .RTF
<type> the text into a word-processor or HTML editor
<scan> a paper document with a scanner
Output : *save* file into (server folders)
Automation : None - use **standard tools** (MS Word, Web editor)
This does not require programming, but needs user instructions and coordination between teachers.

Task : Make individual student reading lists **User : Teacher**

Interface : Should add this to the Student-Records-Reviewer
Input : Current reading list from (assignment file)
List of all document names from (server)
Output : New or changed reading list into (assignments file)
Actions : <click> on lists to *add* or *delete* items
Automation : *collect* list of all document names from (server)
save reading list into (assignments file)

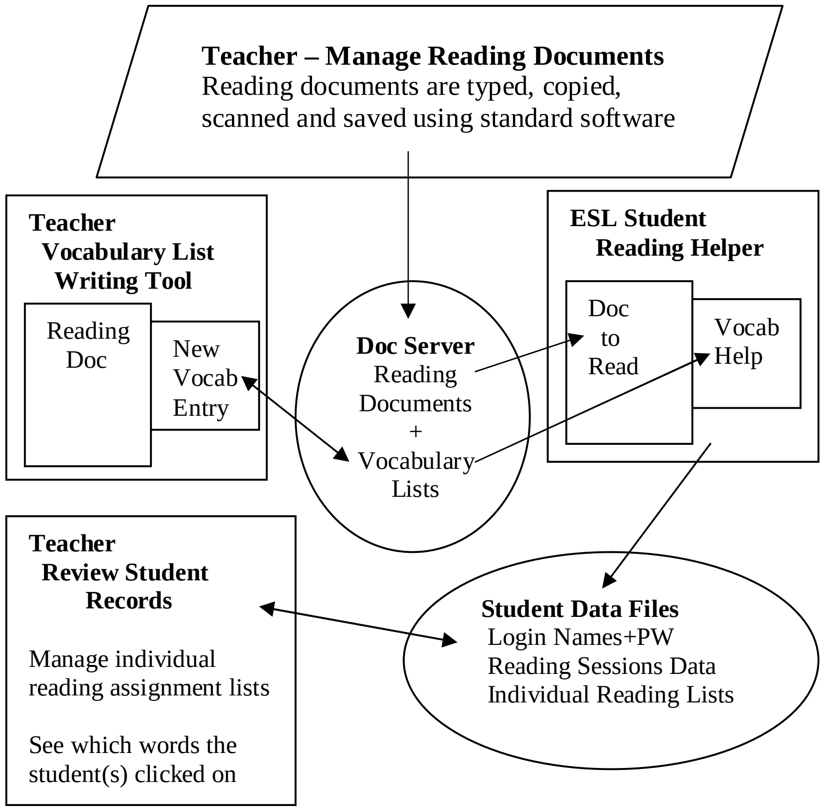
Task : Combine Vocab Lists to Big List **User : Teacher**

Interface: Vocabulary-Recorder
Input : Vocabulary lists from (server)
Output : Big vocabulary list to (server)
Actions : <choose> vocab lists
Automation: *merge lists* , *eleminate* duplicates,
save to (server)

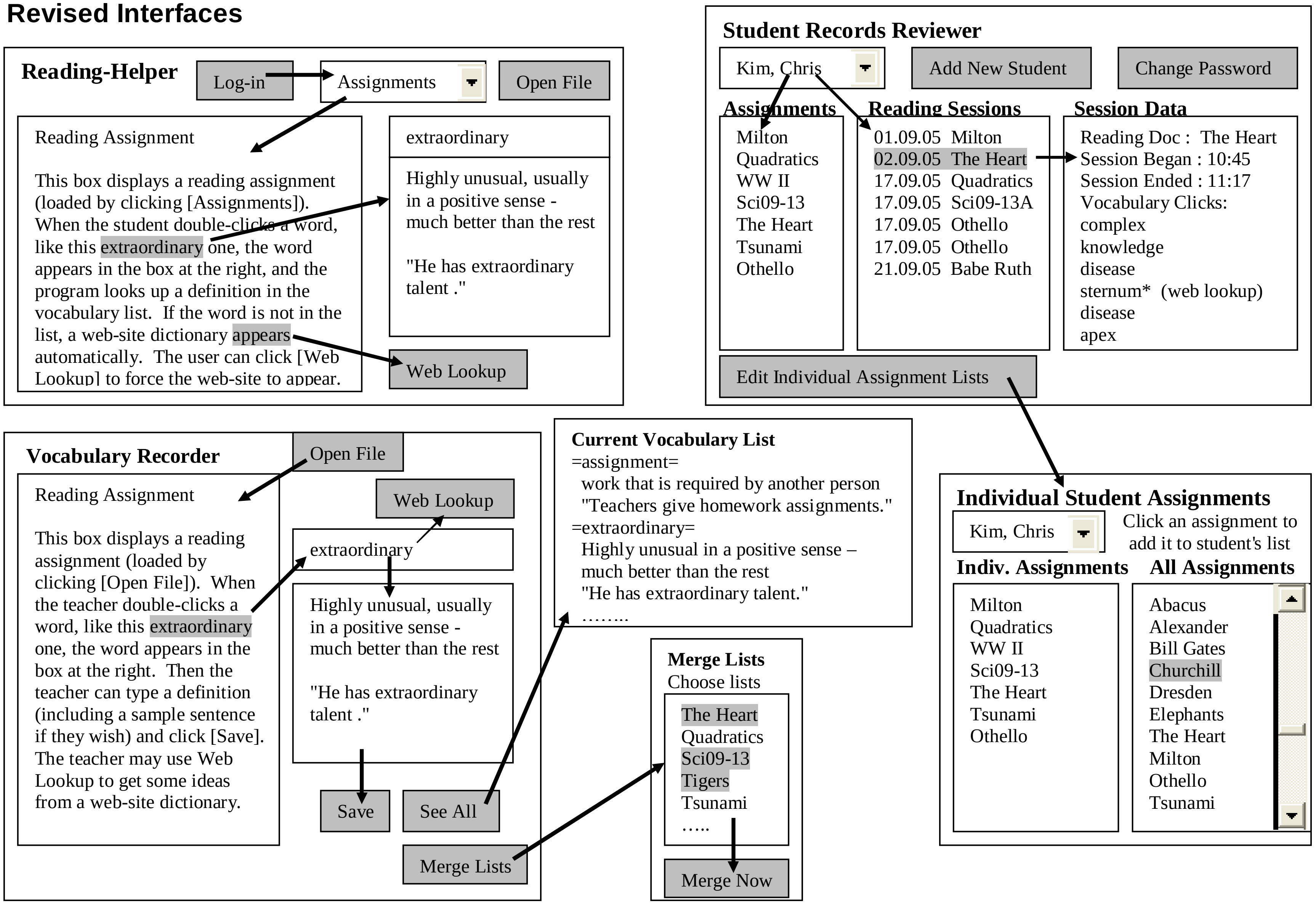
Revisions

After **discussions** with the user and summarizing the **user stories**, a few changes in the initial **system design** and the **user interfaces** were needed. Most important was the addition of **student data files** in the system design.
The revisions are shown below.

Revised System Design


Teacher – Manage Reading Documents
Reading documents are typed, copied, scanned and saved using standard software
Teacher Vocabulary List Writing Tool
Reading Doc
New Vocab Entry
Doc Server
Reading Documents + Vocabulary Lists
ESL Student Reading Helper
Doc to Read
Vocab Help
Teacher Review Student Records
Manage individual reading assignment lists
See which words the student(s) clicked on
Student Data Files
Login Names+PW
Reading Sessions Data
Individual Reading Lists

Revised Interfaces



Reading-Helper

Log-in → Assignments · Open File

Reading Assignment

This box displays a reading assignment (loaded by clicking [Assignments]). When the student double-clicks a word, like this extraordinary one, the word appears in the box at the right, and the program looks up a definition in the vocabulary list. If the word is not in the list, a web-site dictionary appears automatically. The user can click [Web Lookup] to force the web-site to appear.

extraordinary

Highly unusual, usually in a positive sense - much better than the rest

"He has extraordinary talent ."

Web Lookup

Student Records Reviewer

Kim, Chris · Add New Student · Change Password

Assignments	Reading Sessions	Session Data
Milton	01.09.05 Milton	Reading Doc : The Heart
Quadratics	02.09.05 The Heart	Session Began : 10:45
WW II	17.09.05 Quadratics	Session Ended : 11:17
Sci09-13	17.09.05 Sci09-13A	Vocabulary Clicks:
The Heart	17.09.05 Othello	complex
Tsunami	17.09.05 Othello	knowledge
Othello	21.09.05 Babe Ruth	disease
		sternum* (web lookup)
		disease
		apex

Edit Individual Assignment Lists

Vocabulary Recorder

Open File

Reading Assignment

This box displays a reading assignment (loaded by clicking [Open File]). When the teacher double-clicks a word, like this extraordinary one, the word appears in the box at the right. Then the teacher can type a definition (including a sample sentence if they wish) and click [Save]. The teacher may use Web Lookup to get some ideas from a web-site dictionary.

Web Lookup

extraordinary

Highly unusual, usually in a positive sense - much better than the rest

"He has extraordinary talent ."

Save · See All

Merge Lists

Current Vocabulary List
=assignment=
  work that is required by another person
  "Teachers give homework assignments."
=extraordinary=
  Highly unusual in a positive sense – much better than the rest
  "He has extraordinary talent."
  ……..

Merge Lists
Choose lists

The Heart
Quadratics
Sci09-13
Tigers
Tsunami
…..

Merge Now

Individual Student Assignments

Kim, Chris

Click an assignment to add it to student's list

Indiv. Assignments	All Assignments
Milton	Abacus
Quadratics	Alexander
WW II	Bill Gates
Sci09-13	Churchill
The Heart	Dresden
Tsunami	Elephants
Othello	The Heart
	Milton
	Othello
	Tsunami

=== Goals (Objectives) ===

These goals are based on the **discussions** with the user, followed by revised **system design** and **interface designs** to accomplish the tasks listed in the **user stories.**

--- General Goals (wishes) ---

- More effective collection, storage, and distribution of reading assignments and corresponding vocabulary lists
- More effective student access to reading assignments and use of vocabulary lists
- Automation of clerical tasks like compiling and merging vocabulary lists
- Collect and review students' reading habits and vocabulary needs

--- Detailed Goals (requirements) ---

Reading (students can do this)
- Students can read assignments on the computer
- Reading assignments are loaded from central storage
- Students can click on a word to see vocabulary help
- Vocabulary help comes from 3 different sources
  - custom vocabulary list for the current document
  - general vocabulary lists prepared by ESL staff
  - standard web-based dictionary
- Reading lists for each student are stored centrally
- Students' vocabulary clicks are recorded for later review

Writing (teachers can do this)
- Teachers can see reading assignments on the screen
- Teachers can create vocabulary lists easily by clicking on a word, typing a definition, and saving the entry
- Individual vocabulary lists can automatically be merged into larger lists

Reviewing Students' Reading (teachers can do this)
- Teachers can create lists of reading assignments for individual students
- Teachers can see dates when student(s) read assignments
- Teachers can see which words students clicked on

Data Storage (computer must do this)
- Reading assignments (documents) are stored centrally, in a standard format (HTML and/or RTF)
- Reading assignments can be edited with standard editors (MS Word or a WYSIWYG HTML editor)
- Vocabulary lists are stored centrally in structured HTML
- Vocabulary lists can be edited with standard editor(s)
- Individual student reading lists are stored centrally
- Student vocabulary needs are recorded and stored centrally

Automation (computer must do this)
- automatic look-up of vocabulary words
- automatic loading of reading lists and documents
- automated writing and formatting of vocabulary files
- automatic recording and storage of student reading sessions (file-name, date, word-clicks)
- automated restructuring (formatting and merging) of vocabulary lists

Restrictions/Limitations

The following equipment is available at our school:

- PCs in 4 computer labs, as well as 14 PCs in one ESL classroom
- A PC on each teacher's desk
- A local area network, usable by students and teachers
- Servers providing hundreds of GigaBytes of central storage
- All PCs have Internet access
- All PCs run Windows 2000 OS
- All PCs have Java installed (version 1.4 or later)

The solution will be designed to function in the school's network environment as described above.

Access Rights

Students' do not have the same access rights as teachers. Documents need to be stored in an area where teachers have full access, but students have read-only access. Student records (reading session data) must be saved in a different area, where the students have write-access.

Home use

Students have limited access to the school's intranet from home, but this will not be sufficient for this program to run at home. It may be possible to "export" individual reading assignments and vocabulary files so students can do their reading at home, but it probably won't be possible to collect their data when they are working at home, or to provide full access to the entire collection of documents.

Web Access

The system should provide access to web-based dictionaries. When no web-access is available, the rest of the system should still function in our intranet. There is no intention to provide web-access from home to the document collection.

Document Formats

The system will be designed to store and display reading assignments in HTML or RTF formats. The vocabulary lists must be stored as HTML documents.

Equipment Failure

Our intranet experiences occasional outages – on the order of a couple times per week. The ESL reading system will be unavailable during network outages. It would be good if running programs don't crash immediatelhy when the network goes down, but this may be difficult to achieve.

Large Documents

The system is designed with small documents in mind – a few pages. Long documents (like entire books) might function, but there is no intention of providing search facilities or indexing facilities for large documents.

Document Compatibility

The JEditorPane will be used for document display. This imposes some limitations. The program might support hyperlinks, but it will not be a full-fledged browser. Only HTML and RTF will display properly – no PDF or DOC files.

== Appendix ==

-- This is the listing of the **feasibility prototype** presented to the user (Mr S) during the Analysis stage --

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.text.html.*;
import java.io.*;

public class ClickWords extends EasyApp
{
   public static void main(String[] args) throws IOException
   {  new ClickWords();  }

   String homeFolder = System.getProperty("user.dir");

   String[][] dictionary = new String[2][50000];
   int dictionaryMax = -1;

   String[][] vocab = new String[2][10000];
   int vocabMax = -1;

   JEditorPane eHelp ;

   JEditorPane eReading ;
   Button bMagnaC = addButton("MagnaC",1020,50,50,30,this);
   Button bWhatIs = addButton("What Is",1070,50,50,30,this);
   Button bHeart = addButton("Heart",1020,50,50,30,this);
   Button bFile = addButton("File",620,50,50,30,this);
   Button bHelp = addButton("Help",1020,50,50,30,this);
   Button bWeb = addButton("Web Lookup",670,50,100,30,this);
   TextArea test = addTextArea("",620,600,300,100,this);

   Label lVocab = addLabel("Vocabulary Help",620,110,100,30,this);
```

```
public ClickWords() throws IOException
{
   setSize(1000,750);
   setTitle("ESL Reading Helper - (c) Dave Mulkey, Germany, 2005");
   loadDictionary(homeFolder + "\\dictionary\\dictionary.htm");

   eReading = new JEditorPane("file:" + homeFolder + "\\heart\\heart.htm");
   eReading.setContentType("text/html");
   eReading.setEditable(false);

   JScrollPane sWebpage = new JScrollPane(eReading);

   add(sWebpage,null);

   sWebpage.setBounds(10,40,600,700);

   eReading.addHyperlinkListener( new HyperlinkListener()
   //*** This code was downloaded from a Java code snippets web-site ***
   //*** The author claims no credit for this code. ****
      {
         public void hyperlinkUpdate(HyperlinkEvent event)
         {
            HyperlinkEvent.EventType eventType = event.getEventType();
            if (eventType == HyperlinkEvent.EventType.ACTIVATED)
            {
               if (event instanceof HTMLFrameHyperlinkEvent)
               {
                 HTMLFrameHyperlinkEvent linkEvent =
                   (HTMLFrameHyperlinkEvent) event;
                 HTMLDocument document =
                   (HTMLDocument) eReading.getDocument();
                 document.processHTMLFrameHyperlinkEvent(linkEvent);
               }
               else
               {
                  showPage(event.getURL().toString());
               }
            }
         }
      }
   //********  End of downloaded code  ******************************
   );
```

```
      eReading.addMouseListener(
         new MouseAdapter()
         {
               public void mouseClicked(MouseEvent e)
               {
                  try
                  {  String selection = eReading.getSelectedText().trim();
                     showHelp(selection);
                  }
                  catch(Exception exc) {}
               }
               public void mouseEntered(MouseEvent e){}
               public void mouseExited(MouseEvent e){}
               public void mousePressed(MouseEvent e){}
               public void mouseReleased(MouseEvent e)
               {  try
                  {
                     String selection = eReading.getSelectedText().trim();
                     showHelp(selection);
                  }
                  catch(Exception exc) {}
               }
         }
      );

      eHelp = new JEditorPane();
      eHelp.setContentType("text/html");
      eHelp.setEditable(false);

      JScrollPane sHelp = new JScrollPane(eHelp);

      add(sHelp,null);
      eHelp.setPage("file:" + homeFolder + "\\dictionary\\dictionary.htm");

      sHelp.setBounds(620,140,370,450);

   }
```

```
public void actions(Object source,String command)
{
   if (source == bMagnaC)
   {  showPage("http://www.gutenberg.org/dirs/etext06/magna01.txt"); }
   else if (source == bWhatIs)
   {  showPage("http://www.gutenberg.org/files/16728/16728-h/16728-h.htm");  }
   else if (source == bHeart)
   {  showPage("file:" + homeFolder + "\\heart\\heart.htm"); }
   else if (source == bFile)
   {  showPage("file:" + chooseFile());  }
   else if (source == bHelp)
   {  showPage("file:" + homeFolder + "\\helpfile.htm");  }
   else if (source == bWeb)
   {  try{webLookup(eReading.getSelectedText().trim());  }
      catch(Exception exc){ output("First select a word in the reading");}
   }
}


public void showHelp(String word)
{
   word = word.toLowerCase();
   String single = word;
   if (single.charAt(single.length()-1) == 's')
   {  single = single.substring(0,single.length()-1); }
   String help = "";
   for (int x = 0; x <= dictionaryMax; x++)
   {
      if (dictionary[0][x].indexOf("|" + word + "|")>=0
           || (help.length()==0 && dictionary[0][x].indexOf("|" + single + "|")>=0))
      {
         help = "<b>" + word + "</b>" + dictionary[1][x];
      }
   }
   if (help.length()>0)
   {  String helpText = "<html><head><base href=\"" + homeFolder+"\\dictionary\\ \"></head><body>"
                        + help + "</body>";
      eHelp.setText(helpText);
   }
   else
   {  webLookup(word);  }
}
```

```
public void webLookup(String word)
{
   runProgram("explorer \"http://www.wordwebonline.com/search.pl?w=" + word+"\"");
}

private void showPage(String page)
{
   try
   {
      eReading.setPage(page);
   }
   catch (Exception e)
   {
      eReading.setText(page);
   }
}

public void loadDictionary(String fileName)
{
   try
   {
      dictionaryMax = -1;
      BufferedReader info = new BufferedReader(new FileReader(fileName));
      while (info.ready())
      {
         String nextWord;
         boolean found = false;
         do
         {  nextWord = info.readLine();
            if ( nextWord.indexOf("|") >= 0 )
            {  found = true; }
         } while (!found && info.ready());

         if (found)
         {  dictionaryMax = dictionaryMax + 1;

            dictionary[0][dictionaryMax] = nextWord;
```

```
               String ex = "";
               boolean done = false;
               do
               {
                  String more = info.readLine();
                  if ( more.indexOf("~~~") == -1 )
                  { ex = ex + more ; }
                  else
                  { done = true; }
               }    while (!done && info.ready() );
               dictionary[1][dictionaryMax] = ex;
           }
         }
         info.close();
      }
      catch (IOException exc)
      {  output(exc.toString()); }
   }

}
```

Criterion A1: Analysing the problem

The documentation should be completed first and contain a thorough discussion of the problem that is being solved. This should concentrate on the **problem** and the goals that are being set, not on the method of solution. A good analysis includes information such as sample data, information and requests from the **identified end-user**, and possibly some background of how the problem has been solved in the past. A **systematic method** is one that takes into account what input and output will occur and what calculations and processes will be necessary to obtain the desired output.

0 : The student has not reached a standard described by any of the descriptors given below. For example, the student has simply described the programmed solution.
1 : The student only **states** the problem to be solved **or shows** some evidence that relevant information has been collected.
2 : The student **describes** the problem to be solved.
3 : The student describes the problem **and provides evidence** that information relating to the problem has been collected.
4 : The student provides evidence that a **systematic method** has been used in the analysis of the problem.

This section of the program dossier would typically be two to three pages in length. It should include a brief statement of the problem as seen by the end-user. A discussion of the problem from the end-user's point of view should take place, including the user's needs, required input and required output. For example, evidence could be sample data, interviews and so on, and could be placed in an appendix.

Criterion A2: Criteria for success

This section of the program dossier will clearly state the objectives/goals of the solution to the problem. The expected behaviour of the solution should be clearly described and the limits under which it can operate outlined.

0 : The student has not reached a standard described by any of the descriptors given below.
1 : The student **states some** objectives of the solution.
2 : The student **describes most of** the objectives of the solution.
3 : The student **relates all of** the objectives of the solution to the analysis of the problem.
4 : The student relates all of the objectives of the solution to the analysis of the problem, and **outlines** the limits under which the solution will operate.

This section of the program dossier would typically be one to two pages in length. Objectives should include minimum performance and usability. These criteria for success will be referred to in subsequent criteria, for example criterion C2 (Usability), C4 (Success of program); D2 (Evaluating solutions) and D3 (Including user documentation).

The limits under which the solution will operate will vary. Some examples are:
• Time taken to return a research result from a data file
• The response of the program to invalid and extreme data input
• Limitations on the volume of data stored in the program
• Usability of user input screen
• The proper response of the program to user input.

Criterion A3: Prototype solution

The prototype solution **must** be preceded by an initial design for some of the main objectives that were determined to be the criteria for success. A prototype of the solution should be created.
A prototype is: “The construction of a simple version of the solution that is used as part of the design process to demonstrate how the system will work.”

0 : The student has not reached a standard described by any of the descriptors given below.
1 : The student includes only an **initial design**.
2 : The student includes an initial design **and a prototype**, but they do not correspond.
3 : The student includes an initial design and a prototype that **corresponds**.
4 : The student includes an initial design and a complete prototype that corresponds to it **and documents user feedback** in evaluating the prototype.

The prototype need not be functional, it could be constructed using a number of tools such as: Visual Basic, PowerPoint, Mac Paint, Corel Draw for a simple Java program. The intent is to show the user how the system is expected to operate, what inputs are required and what outputs will be produced. A number of screenshots will be required for the user to be able to evaluate the solution properly. The prototype, at its simplest, could be a series of clear, computer-generated drawings, a hierarchical outline of features in text mode, or a series of screenshots.

Documentation of user feedback could be, for example, a report of the user’s comments on the prototype

Tips and Suggestions

How do I get started?

Don't start by writing a computer program. You can do a bit of programming early (functional prototype), but some investigation and design **must be done first**. Choose a **specific problem** and identify an **intended end-user** before doing anything else.

Aiming for Maximum Marks

IA assessment is **criterion** based. This is not a test. There are no "right answers". There are no "points" to add up.

Your documentation must **meet the criteria** to score maximum marks. For example, if you don't have an "intended user", you cannot score 4 marks in section A3, and probably cannot score more than 2 marks in section A1.

** **Read** the criteria and **follow them. ****

Mastery Factors

Check with your teacher early on to ensure the problem and your solution are sufficiently complex to encompass 10 mastery factors. For example, if you don't store any data in data files, you won't fulfill mastery of files (SL) or random-access files (HL). Even in the early stages, you must be looking forward to the programming process. A missed mastery mark carries a 10% penalty!

Okay, how do I proceed?

You need to complete the **(A) Analysis** and **(B) Detailed Design** before you start writing the program. The following process for Stage A is intended to : (1) meet the **criteria** for maximum marks; (2) be **doable** by a student; (3) get **maximum benefit** from a moderate investment of time and effort. Be sure to **keep records** of all your work (even scrap paper) – you will need them later.

Suggested Process for Stage A - Analysis

1. Choose a **problem** (be very **specific**)
2. Choose the **intended-end-user(s)** (at least one real person) – this should not be "everybody" or "me myself"
3. **Describe** the problem, including **current/previous solutions.** What happens now? How is it done? What is unsatisfactory?
4. **Collect** sample **documents** and **data** from the current solution
5. Outline **intended improvements** of a new system
6. Create an **initial system design** (simple) for the new system
7. Create a **prototype** – either **functional** (a program) or **mocked-up** (interfaces only).
8. *optional:* Use a functional prototype to investigate the **feasibility** of programming the new system (solve some tricky problems)
9. Use the prototype(s) to **discuss** the new system with the user
10. Collect the user's **reactions** and **suggestions** (written down)
11. **Document** the most significant **user stories** which identify the major **tasks** to be performed with the new system
12. **Improve** the **system design** and **interfaces** to meet the **user's suggestions** and fulfill the needs in the **user stories**
13. Write a **clear** and **complete** set of **goals** – extracted from the user stories and revised design – specifying what it **WILL do**
14. **Clarify limitations** of the intended system – what it **WON'T do**
15. Review **goals** and **limitations** with the intended user and revise (if necessary) until the user is satisfied