**Software for the Teaching of Organic Chemistry Compounds and their Nomenclatures**

Chris Lansing

Tim Elam

Roland Heintze

John Gibbons

#shaft @irc.freenode.net

(Placeholder Coversheet)

Table of Contents (to be added)

|  |  |  |
| --- | --- | --- |
| Name: | Email: | Role: |
| Dr. Shainaz M. Landge | slandge@georgiasouthern.edu | Customer |
| Roland Heintze | dh02599@georgiasouthern.edu | Project Lead |
| Tim Elam | te00838@georgiasouthern.edu | Lead Developer |
| Chris Lansing | cl02064@georgiasouthern.edu | Developer/Clerk |
| John Gibbons | jg03039@georgiasouthern.edu | Quality Assurance |

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Number:** | **Revision Date:** | **Author:** | **Summary of Changes:** |
| 1 | 2-28-13 | John Gibbons | First revision of draft. Reworded sentences and corrected grammatical errors. |
| 2 | 3-4-13 | Roland Heintze and Chris Lansing | Second revision of draft. Moved document into Word. Added use cases, cover page, formatting and a table of contents. |
| 3 | 4-9-13 | John Gibbons | Third revision. |
| 4 | 4-25-13 | John Gibbons | Fourth revision while updating requirements and platform changes. |
| 5 |  | Roland Heintze | Final revision. |

OVERALL DESCRIPTION

This document specifies the Software Requirements for the application for teaching organic compound nomenclature methodology. It describes the scope, requirements, constraints, and interfaces.

a) Product Perspective

The product will operate as an application for assisting students in learning the proper naming conventions of organic compounds. It will allow users to generate random organic molecules or create their own. The user can then choose to be given the proper nomenclature of the molecule or provide a guess. The user will submit guesses of what the proper name of the compound is. Whether deciding to guess the name or not , the user may elect to watch a brief animation of an extended metaphor that explains the naming process from the perspective of a hypothetical mailman.

The dual purpose of this document is to gather requirements for our groups planned solution to deliver to Dr. Landge and to track deviations from our initial development plans. The audience of this document is Dr. Landge and her peers.

1.1.1 User Interfaces

This should specify the following:

a) The logical characteristics of each interface between the software product and its users.

The GUI must be tailored in such a manner as to be easy to use on the target platform. It must also be easy to use for people with limited technical experience. The GUI must have complete functionality without being overly difficult or involved.

b) All the aspects of optimizing the interface with the person who must use the system.

The User Interface (U.I.) should be easy to learn and use.

The U.I. should be simple and organized in a neat manner.

The U.I. should provide obvious and direct links to other forms/menus.

The U.I. should have error feedback such that the user can know what they did wrong and what they need to do correctly the next time in case of an error.

The U.I. should contain vocabulary in which the user will be familiar with.

The U.I. design and colors should be pleasant to the user.

1.1.2 Software Interfaces

This should specify the user of other required software products and interface with other application systems.

Operating System:

Windows XP, Vista, Windows 7, Windows 8, Linux, Solaris, and Mac OS.

Windows: RAM - 128mb, 64mb for Windows XP (32 bit), Disc Space - 124 MB, Browsers - Internet Explorer 7.0 and above, Firefox 3.6 and above, Chrome

Mac OS X: Intel-based Mac running Mac OS X 10.7.3 (Lion) or later. Administrator privileges for installation, 32/64-bit browser.

Linux - Oracle Linux 5.5+, Oracle Linux 6.x (32-bit), 6.x (64-bit), Red Hat Enterprise Linux 5.5+, 6.x (32-bit), 6.x (64-bit), Ubuntu Linux 10.04 and above, Suse Linux Enterprise Server 10 SP2, 11x, Ram - 64 MB, Disk space - 58 MB, Browsers - All OS that support versions of Firefox 3.6 and above.

1.1.3 Communications Interfaces

This should specify the various interfaces to communications such as local network protocols, etc.

The user needs to have a working internet connection to be able to download the software or a working USB drive. The user may also need to download Py2exe, a Python Interpreter, in the event the of the software’s exported .exe file does not work on their machine.The connection will be provided by Georgia Southern University via the campus network or the software will be provided by the professor via USB or emailed to the student’s My.GeorgiaSouthern emails.

1.1.4 Memory Constraints

This should specify any applicable characteristics and limits on primary and secondary memory.

At least 128mb of RAM and at least 50mb of persistent storage.

1.1.5 Operations

This should specify the normal and special operations required by the user such as

a) The various modes of operations in the user organization (user-initiated operations)

The user clicks a button to generate a random bonded organic molecule. After viewing the animation that shows the molecule’s naming procedure, the user can click a button to re-watch the animation. The user can also click within boxes arranged in a grid to generate a set of molecules in a window to form an organic compound and then watch the animation of its naming process. Before choosing to watch the animation, the user has the option to enter in what they believe to be the name of the organic compound and check to see if their guess is correct. This is completely optional and is not required to begin the animation. The user can enter in and check the validity of their guess as many names as they wish before clicking on the button to begin the animation. At the end of the animation, the correct answer for the name of the organic compound will be displayed. If the user wishes to see the name of the molecule before the animation starts, they have the ability to do so.

b) Periods of interactive operations and periods of unattended operations.

The user can choose to participate in an interactive series of operations by making their own organic compound. This will involve clicking in boxes laid out out in a grid format to customly form their own compound. This step is optional and is not required to be completed should the user so choose to. In the initial release of the software, the users will be limited to single bonds of organic molecules. There are no unattended operations.

c) Data processing support functions.

Generating random organic molecules and determining the proper name for a given bonded organic molecule.

d) Backup and recovery operations.

There is no backup option. The data is randomly generated or created by the user and is not critical to the program or later instances of the program. Each new randomly generated organic compound or user created organic compound is sufficient data for the program to use and for the user to learn from.

1.1.6 Site adaptation requirements

a) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode (e.g., grid values, safety limits, etc)

The program will run on any machine that has Py2exe installed. However, Linux and Unix machines already have Python Interpreters and do not require this software.

1.2 Production Functions

*This subsection of the SRS should provide a summary of the major functions that the software will perform. For example, an SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires.*

*Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product. Note that for the sake of clarity*

a) The functions should be organized in a way that makes the list of functions understandable to the customer or anyone else who reads the document for the first time.

This program will allow any user to generate a random organic compound and view an animation on the procedures of the scientific nomenclatures for carbons up to and including pentadecane. The user can also create their own organic compound through a series of clicking events into a series of boxes laid out in a grid format. Either process delivers an animation for the compounds’ naming process. Before the user clicks the button to view the animation, they have the option of entering in the name of the bonded molecule and checking to see if their guess is correct. The user can repeat the process of entering in a name and checking its validity as many times as they wish before viewing the animation. To begin the animation, the field where the name can be entered does not have to contain anything or contain the correct answer to begin. The design of the animation will be generated to exactly mimic the shape of the randomly generated or customly created organic molecule. A mailman will appear at the far left and upper position of the generated animation which resembles a road. The mailman will then walk the correct path down the roads while delivering mail to show how the naming process functions. After the animation has completed, the correct name of the compound will be displayed.

1.3 User Characteristics

*This subsection of the SRS should describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. It should not be used to state specific requirements, but rather should provide the reasons why certain specific requirements are later specified in Section 3 of the SRS.*

The intended users for this software will be Organic Chemistry professors and students. Each instance of the program will support one user at a time.

Professors:

* + 1. Education Level
       1. They possess a PH.D. in Organic Chemistry and have research experience in this field.
       2. They have intimate knowledge of how the naming process works and how it was created.
    2. Experience
       1. Dr. Landge has experience with the teaching method that generates the organic compound. She has no experience with software which contains animations or the ability to craft organic compounds.
       2. The professors have experience with an operating system capable of running the software.
    3. Technical Expertise.
       1. Ability to navigate common desktop environments.
       2. Understanding of virtual form controls.
       3. Understanding of the mouse clicking interaction metaphor.

Students:

* + 1. Education Level.
       1. Students who are currently studying Organic Chemistry and can range from freshmen to postgraduate levels.
       2. Students possess little to no knowledge of how organic compound naming process works or how it was created.
    2. Experience.
       1. They possess no experience with this software.
       2. Assumed to posses basic to high level experience with common desktop applications.
    3. Technical Expertise.
       1. Ability to navigate common desktop environments.
       2. Understanding of virtual form controls.
       3. Understanding of the mouse clicking interaction metaphor.

1.4 Constraints

*This subsection of the SRS should provide a general description of any other items that will limit the developers options. These include*

a) Hardware Limitations

Monitors should support 4:3 and 16:10 aspect ratios. Minimum resolution of 800x600. The program should run on all common graphical displays.

b) Interface to other applications

The program will not interface with any external applications.

c) Parallel operation

The application will run as a single process running two separate threads. One thread will be used to receive input events from the user and update the display, while a second thread will be responsible for maintaining the internal state of the program and handling any calculations that might slow down the user interface.

d) Audit functions

Does not apply.

e) Higher-order language requirements

We are using Python 3 with the PyQt4 library to generate the randomly bonded or customly designed organic compounds and also to generate the animations for the naming process.The software requires a cross-platform graphical user interface and support for multithreading and synchronization.

f) Reliability requirements

This software will be used for teaching and studying purposes. It follows that the application should perform consistently and correctly use naming conventions of organic molecules.

g) Criticality of the application

This program will teach how organic compounds are named. It is not considered critical.

h) Safety and security considerations

The program does not contain any information that would need to be guarded or encrypted.

1.5 Assumptions and Dependencies

*This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.*

Our team assume that the users will mainly be using this program on the school computers, which have Windows 7 installed. The school computers are known to support 4:3 and 16:10 displays and support a resolution that exceeds 800px:400px.

1.6 Apportioning of Requirements

*This subsection of the SRS should identify requirements that may be delayed until future versions of the system.*

The client requires the development of animation software to illustrate the naming process for organic compounds. Our software will allow for the generation of random compounds as well as user-defined ones. The software will then generate an animation. The largest foreseeable improvements on the software for future versions would be to increase the quality of the animations and other graphical artifacts. The initial release will have simplistic animations and user interface flourishes.

2. Specific Requirements

*This section of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input or in support of an output. As this is often the largest and most important part of the SRS, the following principles apply:*

1) Specific requirements should be stated in conformance with all the characteristics described in 4.3.

2) Specific requirements should be cross-referenced to earlier documents that relate.

3) All requirements should be uniquely identifiable.

4) Careful attention should be given to organizing the requirements to maximize readability. Before examining specific ways of organizing the requirements, it is helpful to understand the various items that comprise requirements as described in 5.3.1 through 5.3.7.

Specific Requirements:

This project entails creating software to generate and animate different organic compounds. The user will also have the option of generating their own compounds. They will then be capable of entering in what they believe to be the name of the displayed/made organic molecule and check if their answer is correct.

1. Creating the backend.

a. Having it create a data structure to represent the organic compound.

b. Adding in the ability for the user to create their own organic compound.

c. Adding in the ability for the user to enter in what they believe to be the name of the organic compound and checking to see if the answer is correct.

2. Creating the animation.

a. Design and implement a new U.I. to handle the animation and all controls that go with it. This includes a button to re-watch the animation, a textbox to enter in the name of the compound as well as a button to check the answer, and buttons to generate another random compound or exit the program.

b. Creating the algorithm which will define the data structure for the compound and its various components.

c. Creating the design of the animation to mimic the exact organic compound that is stored.

2.1 External Interfaces

*This should be a detailed description of all inputs into and outputs from the software system. It should include both content and format as follows:*

1. Creation of Randomly Generated Organic Bonded Molecule:

a. Name of item:

Randomly generated or user created organic compound.

b. Description of purpose:

An organic compound which the user will learn how to name.

c. Source of input or destination of output:

The application will randomly generate this compound or allow the user to construct their own.

d. Valid range, accuracy, and/or tolerance:

The input will be a correct version of an organic compound limited by the current functionality of the program.

e. Units of Measure:

The only units of measurement will be the atoms themselves.

f. Timing:

Does not apply.

g. Relationships to other input/outputs:

The animation will mimic the generated organic compound to show the naming process. The nomenclature displayed at the end of the animation will be calculated by a separate naming algorithm.

h. Screen formats/organization:

The program will have focus and be in the forefront of the window until the user clicks the animation button. A canvas section will then show the animation and upon completion will display the forming of the compound’s nomenclature. After the animation is finished, the animation screen will disappear and the programs’ main form will regain forefront attention.

i. Window formats/organization:

One window for the application with appropriate frames.

j. Data Formats:

The organic compound will have its own class and have the appropriate attributes needed for the different algorithms.

k. Command formats:

Does not apply.

l. End Messages:

None are needed. The user is brought back to a menu screen with options.

2.2 Functions

*Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as shall statements starting with “The system shall”*

The system shall accept and check the input of the generated organic compound and ensure it is a valid structure. It will not accept further action until a valid compound is provided by the user.

The system shall accept and check the name for the currently displayed organic compound. The system will then check to see if the name is correct or incorrect and inform the user of the result. The system will allow the user to repeat this process as many times as they desire.

The system shall generate an animation, in which it will show an analogy of a mailman delivering mail. The mailman will perform the same steps as are taken in the process of naming an organic compound. Roads will be representative of the longest chain while houses will be representative of all substituents.

2.3 Performance Requirements

*This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following:*

a) The program will support one user at a time per client.

2.4 Design Constraints

*This should specify design constraints that can be imposed by other standards, hardware limitations, etc.*

The system should be able to run the exported .exe file. In the event there is an issue, the system shall have Py2exe Python Interpreter installed to run the file. The system shall also have Internet Explorer 7.0 or newer, Firefox 3.6 or newer or Chrome to download the Py2exe software if it is not currently on the machine. The rest of the constraints and limitations will be met due to the specifications required for windows 7/Mac OS/Linux/Unix on the machines.

2.4.1 Standards Compliance

This subsection should specify the requirements derived from existing standards or regulations.

The only standard applicable to this project is the IEEE standard. This project is not subject to FERPA nor HIPPA.

a) Report format:

All reports and paperwork pertaining to this project will follow guidelines set by IEEE and other Software Engineering practices and standards.

b) Data naming

The names of all classes, methods, functions and variables will be modeled after standard Object Oriented Design principles. They will be concise and related to the actions performed or information stored. This will assist later development teams in adding functionality to the software to include other variations and types of organic compounds.

c) Audit tracing

Our Software Development group will be using Google Docs for documentation collaboration and git for code progress. Additionally, we will be utilizing Github, a project management and bug-tracking application. This will provide group members with an online interface for the groups’ repository as well as collaboration tools such as a tickets, a wiki, and a forum. These tools will assist the team in tracking member participation and progress.

2.5 Software System Attributes

*There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. Sub clauses 2.5.1 through 2.5.5 provide a partial list of examples.*

2.5.1 Reliability

This should specify the factors required to establish the required reliability of the software system at time of delivery.

This program should be reliable and able to catch and throw errors with easily understood directions for resolving them. If the user enters in invalid information into the nomenclature guessing box, the program shall tell the user if their input is correct. If the user decides to manually generate an organic compound, the program shall inform the user if their molecule is invalid.

2.5.2 Availability

This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

This program will be accessible on all machines. If some machines have issues with the exported python file to a .exe file, then any machine that contains Py2exe software will be able to successfully run the program. The packaged executable will include all necessary libraries and dependencies. There is no persistent data so no database or network connection is necessary. For the purposes of demonstration, the application will be served as an applet on the internet. As such, a network connection will be necessary.

2.5.3 Security

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:

a) Utilize certain cryptographic techniques

No cryptographic techniques need to be used for this program or subsequent later versions unless the requirements drastically change.

b) Keep specific log or historical data sets

No logs or historical data needs to be collected as the information generated and used by the program is disregarded after use or when new information is generated. No logs need to be created to keep track of users and time used on the program as per the current requirements.

c) Assign certain functions to different modules

Does not apply.

d) Restricted communications between some areas of the program.

No such restrictions exist or are needed in this program.

e) Check data integrity for critical variables.

If the user decides to create their own organic compound, the program will have to validate it. Once the integrity and validity of the compound are ensured, all other functionality can be expected to operate without error.

2.5.4 Maintainability

*This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.*

The code will be broken up into well developed methods and functions as well as be professionally commented. Any and all methods and functions will have comments describing their functionality and usability. This will allow future development teams to make additional features and/or changes to the program with relative ease and understanding of how the current version and previous versions function.

2.5.5 Portability

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

a) Percentage of components with host-dependent code.

The host of the program may require Py2exe software to be installed in order to run the program if the exported .exe file does not execute properly. Thus, there is no component with host-dependent code.

b) Percentage of code that is host dependent.

Zero percent of the code is host dependent.

c) Use of a proven portable language.

Since the application is written in Python 3, all major operating systems that contain a Python Interpreter if the exported .exe file is nonfunctional will run it.

d) Use of a particular compiler or language subset.

Because our team wanted this to program to be easily portable to other machines, we chose Python 3 with PyQt4 library.

e) Use of a particular operating System.

The most used operating system will be Windows 7 as the computers in the school labs have it installed. Most students who will use this program generally have Windows 7 installed. The program will be able to run on any mainstream operating system as long as the exported .exe file is functional or Py2exe Python Interpreter software is installed.

3. Use Cases

**Use Case 1:** User randomly generating an organic molecule up to and including pentadecane and view the animation of the naming process.

**Interest:** User wants to instantly create a bonded organic carbon molecule then view the animation of the naming process..

**Prerequisites:** None

**Scenario:**

* 1. User opens program.
  2. User clicks ‘Generate Molecule’ button
  3. User clicks ‘View Animation’ button.
  4. User closes program.

**Technical Requirement:** Software must be able to generate a random bonded organic molecule up to and including a pentadecane. Software must then create and execute the animation in accordance with the scientific nomenclature designed to the molecule generated.

**Use Case 2:** User wishes to randomly generate an organic molecule, guess the name of the molecule and then view the animation of the nomenclature process.

**Interest:** User wants to instantly create a bonded organic carbon molecule, guess the name of the generated molecule then view the animation explaining how the molecule is named.

**Prerequisites:** User must possess the knowledge for the format pertaining to the names for the molecules.

**Scenario:**

* 1. User opens program.
  2. User clicks ‘Generate Molecule’ button.
  3. User enters in guess for name of generated molecule.
  4. User clicks ‘Check Name’ button.
  5. User clicks ‘View Animation’ button.
  6. User closes program.

**Technical Requirement:** Software must be able to generate a random bonded organic molecule up to and including a pentadecane. Software must then be able to compare the user entered guess to the generate molecule. Finally, the software must then generate the animation and follow the nomenclature relating to the generated molecule.

**Use Case 3:** User wishes to customly create an organic molecule and view the animation of the nomenclature process.

**Interest:** User wishes to create their own version of an organic molecule up to and including a pentadecane. Afterwards, the user wishes to view an animation of the nomenclature process for the created molecule.

**Prerequisites:** User must have knowledge pertaining to the designs for organic molecule up to and including pentadecane molecules.

**Scenario:**

* 1. User opens program.
  2. User goes through a series of mouse clicks in boxes laid out in a grid format to form the shape of the organic molecule.
  3. User clicks ‘Validate Molecule’ button.
  4. User clicks ‘View Animation’ button.
  5. User closes program.

**Technical Requirements:** The software must be able to compare the user created organic molecule with all forms of organic carbon molecule up to and including pentadecane. Then the software must be able to generate an animation for the nomenclature of the user created organic molecule.

**Use Case 4:** User wishes to customly create an organic molecule, guess the name of the created molecule and then view the animation for the nomenclature of the molecule.

**Interest:** User wishes to personally create an organic compound, guess the name of said compound, and view the nomenclature process.

**Prerequisites:** The user must posses knowledge for designs of organic compounds up to and including pentadecane molecules as well as their naming formats.

**Scenario:**

* 1. User opens program.
  2. User goes through a series of mouse clicks in boxes laid out in a grid format to form the shape of the organic molecule.
  3. User clicks ‘Validate Molecule’ button.
  4. User enters in guess for name of molecule.
  5. User clicks ‘Check Name’ button.
  6. User clicks ‘View Animation’ button.
  7. User closes program.

**Technical Requirements:** The software must be able to compare the user created organic molecule with all forms of organic carbon molecules up to and including pentadecane. The software must then compare the user entered guess to the name of the created compound. Finally, the software must generate an animation for the nomenclature of the created compound.

4. Prototypes