UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project
Final Paper 2022

Title: Ensuring the Usability and Maintainability of Scientific Web-Tools: A Case Study on "*Glycarbo*"

Author: Lauren Paton

Project Abbreviation: SugarToo

Supervisor(s): Michelle Kuttel

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	
Theoretical Analysis	*0*	*25*	
Experiment Design and Execution	*0*	*20*	
System Development and Implementation	*0*	*20*	
Results, Findings and Conclusions	*10*	*20*	
Aim Formulation and Background Work	*10*	*15*	
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Ensuring the Usability and Maintainability of Scientific Web-Tools

A Case Study on "*Glycarbo*"

Lauren Paton†
Department of Computer Science
University of Cape Town
Cape Town, South Africa
ptnlau002@myuct.ac.za

ABSTRACT

Research into the structural properties of carbohydrates is vital for the development of vaccines and in drug design. Such research relies on the use of software tools in order to visualise and understand the 2D and 3D structure of molecules. There are many existing tools for this purpose but few that provide both 2D and 3D visualisations through an intuitive and usable interface. UCT's *Glycarbo* is a tool built for this purpose and this project demonstrates the inspection, editing and testing of *Glycarbo* to produce a usable and maintainable tool.

KEYWORDS

Visualisation, Carbohydrate, Residue, Glycan, Glycoscience

1 INTRODUCTION

Carbohydrates are a focus of vaccine development because of their dense distribution on different pathogens [18]. According to Watson, Crick and Franklin's "Central Dogma of life", molecules "express" their function throughout their structure [5] and so the 3-dimensional qualities of a molecule determine their biological function. Glycans (carbohydrates) are complex bio-molecules [13] and structural data of glycan interactions is rare [18] so the development of structural visualisation tools is an important part of glycoscience research.

There are three ways in which glycans can be represented: as strings (one-dimensional), graphically (2-dimensional) and as 3D rendered molecules. Both 2D and 3D visualisations of carbohydrates are very useful in the study of the molecule structure and functionality and can be used to populate figures in research papers.

SHAPE	White (Generic)	Blue	Green	Yellow	Orange	Pink	Purple	Light Blue	Brown	Red
Filled Circle	Hexose	Glc	Man	Gal	Gul	Alt	All	Tal	Ido	
Filled Square	HexNAc	GlcNAc	ManNAc	GalNAc	GulNAc	AltNAc	AllNAc	TalNAc	IdoNAc	
Crossed Square	Hexosamine	GlcN	ManN	GalN	GulN	AltN	AllN	TalN	IdoN	
Divided Diamond	Hexuronate	GlcA	ManA	GalA	GulA	AltA	AllA	TalA	IdoA	
Filled Triangle	Deoxyhexose	Qui	Rha		6dGul	6dAlt		6dTal		Fuc
Divided Triangle	DeoxyhexNAc	QuiNAc	RhaNAc			6dAltNAc		6dTalNAc		FucNAc
Flat Rectangle	Di-deoxyhexose	Oli	Tyv		Abe	Par	Dig	Col		
Filled Star	Pentose		Ara	Lyx	Xyl	Rib				
Filled Diamond	Deoxynonulosonate		Kdn				Neu5Ac	Neu5Gc	Neu	Sia
Flat Diamond	Di-deoxynonulosonate		Pse	Leg		Aci		4eLeg		
Flat Hexagon	Unknown	Bac	LDmanHep	Kdo	Dha	DDmanHep	MurNAc	MurNGc	Mur	
Pentagon	Assigned	Api	Fru	Tag	Sor	Psi				

Figure 1: Symbol Nomenclature for Glycans (SNFG) [13]

Casper notation/format is used to represent a glycan as a string [14]. A simple glycan in Casper format consists of a lowercase a/b (anomeric configuration) followed by an uppercase D/L (absolute configuration) followed by a residue's abbreviation. This can be followed by other bonding (furonose/pyronse) and substituent information. A set of brackets containing bond locations shows two bonded monosaccharides, for example aDGal(1->4)aDGlc. Side chains and repeats are represented using square brackets. CarbBuilder, a tool for rendering 3D glycans developed at UCT, uses Casper Format [14].

Since structural glycobiology relies so heavily on use of tools and databases, certain conventions are required to efficiently communicate within the field [19]. There have been many attempts at developing visual and textual conventions for representing carbohydrates [13] and because of the wide variety of glycans – it has been a complex process [19]. One of the most recently developed and widely used 2D graphical representations for glycans is the SNFG (Symbol Nomenclature for Glycans) [13] convention. SNFG (figure 1) has been developed through an active international effort to create a convention that copes with rapidly growing information in Glycoscience. It is also compatible with many other graphical representations [19].

2D glycans can be built by combining and "bonding" different residues from this convention set. There are many available carbohydrate visualisation tools that are both useful and usable. Throughout the course of this project, tools such as PolysGlycanBuilder [16], GlyCamWeb [6] and DrawGlycan-SNFG [4] were considered as reference tools – as each displayed useful design choices. 3D glycans can usually be generated from a Casper string or PDB file. There are also some existing tools for converting graphically-represented glycans to 3D structures including GlycanBuilder, DrawRings [19] and *Glycarbo* [21].

Glycarbo is a web-tool for 2D and 3D building of carbohydrate molecules developed at UCT. Figure 2 below shows *Glycarbo*'s interface. *Glycarbo* allows users the freedom to use any one of the three representations and for this reason, is one of the most useful carbohydrate visualisation tools to date. It comprises of two parts: A 2D

builder (also known as *Glycano*, a tool also developed at UCT [7]) and a 3D rendered. The last iteration of developments on *Glycarbo* left it incomplete and difficult to maintain. Figure 3 below shows a use case diagram of the interface, the use cases in red represent those facing functional issues - mainly related to incorrect naming conventions. *Glycano*'s codebase structure was Suited to its original choice of convention, the "UCT" convention which was based on the use of both residues and substituents. However, upon being integrated into *Glycarbo* the codebases's primary convention became SNFG, to which it was not suited. The previous iteration of design did not address any structural issues and just built over the existing substituent functionality. *Glycano*'s codebase lacked any documentation - maintainability and readability of scientific software is integral in ensuring its continued use. In *Glycano*'s case, the rapidly developing research into glycans means its lack of flexibility and documentation made it difficult to refactor and expand. The problem of *Glycano*'s maintainability and functionality is the focus of this paper. This paper presents the functional and non-functional changes made to *Glycano* and *Glycarbo*'s codebases to improve their chemical accuracy and usability, the documentation added to improve *Glycano*'s maintainability as well as testing performed.

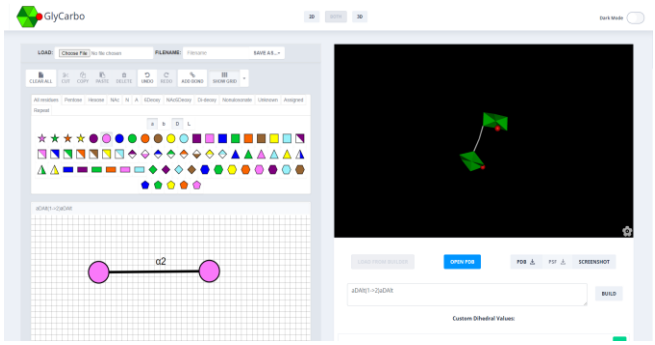


Figure 2: *Glycarbo*'s Interface

2 BACKGROUND AND PREVIOUS WORK

Many of the available tools for carbohydrate visualisation are both useful and usable. *PolysGlycanBuilder* [16] has a "drag and drop" 2D builder as well as a 3D builder which uses the 2D glycan. However, the process of building is very unintuitive and slow. *GlyCamWeb*'s Carbohydrate builder [6] allows users to build glycans by clicking residues from a subset of the SNFG residues and generates a string. There is however, no visual representation of the structure. Finally, DrawGlycan-SNFG [4] allows users to enter a string and generate a 2D structure – this structure is not able to be manipulated by the user and also does not follow SNFG visual conventions. Although each tool fulfills a certain need, most lack evidence of usability practices in their design. *Glycarbo*'s functionality also encapsulates that of most the tools discussed above, while also allowing greater user freedom and control by allowing users to create and manipulate the structures– an important usability practice [11].

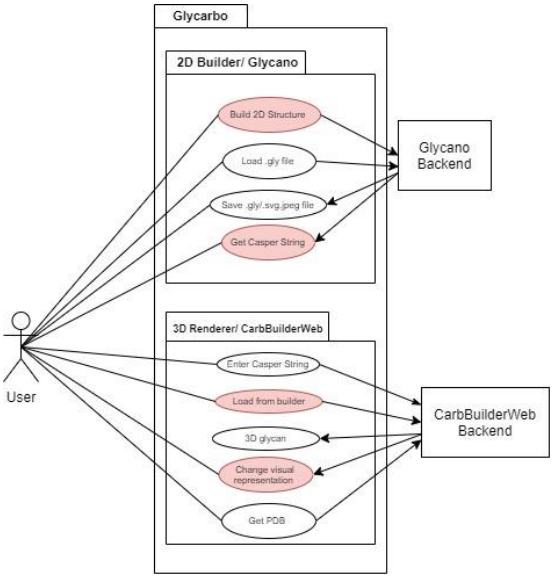


Figure 3: Glycarbo Use Case Diagram

Glycarbo has been built and refactored through a series of iterations since 2015, with changes made to its two parts – *Glycano* (2D visualisation) and its 3D builder.

2.1 *Glycano*

Glycano was originally built in 2015 as a stand-alone project for 2D-visualisation of carbohydrate molecules [7]. It is built using ScalaJS (a functional JavaScript framework) and React. It is designed to accept different conventions of residue representation but was primarily structured for a convention named "UCT".

A round of testing and editing was done on *Glycano* in 2020 so that it could be used in *Glycarbo*, to build 3D molecules. The edits made included changing *Glycano*'s primary convention to SNFG. The new interface can be seen on the left in Figure 2 above. However, the naming conventions remained incorrect and instead of being restructured to suit the SNFG convention, new functionality was merely added to *Glycano*'s codebase. This meant a lot of redundant functionality was built on top of and the final codebase was extremely complex and lacked proper structure. The codebase is extensive and has very little documentation, with a total of two lines of comments in the entire codebase.

2.2 *Glycarbo*

Glycarbo was built in 2020 and generates 3D carbohydrate molecules. *Glycano* is incorporated into the interface and acts as the 2D builder. The 3D molecules are built using a Casper string (either inputted by the user or automatically taken from the 2D builder) and uses CarbBuilder (a UCT-built software) [14] to generate the image. The front-end users Angular to encapsulate *Glycano* and the 3D-builder and the back-end uses CarbBuilderWeb. Also developed in 2020, CarbBuilderWeb is used to connect CarbBuilder to *Glycarbo*. As it stands the 3D builder works suitably well when given a Casper string in the correct format (it does struggle to build large molecules resulting in the browser crashing). It allows the user to change the bond angles as well as the type of image rendered (ball-and-stick etc.). However, the naming conventions of the 2D builder (*Glycano*) remained incorrect and most residues built in 2D could not then be built in 3D.

3 METHODOLOGY

We followed an iterative approach to documenting and extending the codebases for both *Glycano* and *Glycarbo.* As usability evaluations are most effective when done iteratively throughout the design process [11], we performed regular inspections with a domain expert in order to ensure all requirements were being attended to throughout the project and were in keeping with the previous aims of the tools. The first phase consisted of studying and documenting the codebase. The structure of the codebase was also documented, see Appendix A, in order to study the interactions between classes. This phase was followed by an initial interface inspection with a domain expert – which acted as a requirement gathering session. An initial coding block was then completed prior to testing to incorporate the feedback from our domain expert.

3.1 Documentation of Codebase

The main aim of this project was to ensure that *Glycarbo* reaches its full potential as a multi-use carbohydrate visualisation tool. In order to ensure the continued use of *Glycarbo* it is necessary to ensure its maintainability. *Glycano*'s maintainability was the main focus as the main functionality issues were related to *Glycano* and the 2D builder. The readability of its codebase was poor, with no documentation within in an already complex codebase. This phase of the project included improving the build instructions to include solutions to common issues that arose during development, commenting within the codebase and curating instructions for future development of *Glycano* – see Appendix B.

3.2 Requirements Gathering

A primary interface examination with a supervisor was done to record the expectations of a future user of *Glycarbo* and to highlight the design and logic errors of the system. This step was completed in order to record the expectations of a domain expert when using a visualisation tool.

3.3 Initial Code Fixes

Prior to the software feasibility demonstration and testing, the main issues regarding the structure of the codebase and the incorrect conventions were approached. Following this, other issues related to functional requirements were fixed – these included those related to bonding and other chemical functions. The non-functional needs were met following user- and expert-testing.

3.3 Testing

In order to test the changes implemented after the initial inspection, both expert and non-expert testing were performed on *Glycarbo*. An interface inspection and test were performed with three domain experts. Following this, a task-based user test was designed and completed by two UCT postgraduate Computer Science students with experience with usability practices and testing, but with limited carbohydrate knowledge. The choice to hold two focus group-like sessions of user-tests was to allow discussion between users [9]**.** Elaboration between the users when audio-recorded, is a very helpful form of feedback. It also means there is a higher likelihood of catching errors. Both sessions were audio-recorded and the recordings were deleted after the feedback was extracted. All users signed consent forms stipulating their consent to be recorded on the condition the recordings were deleted and the information was anonymised.

3.3.1 Expert Testing

The aim of doing expert testing is to get feedback from actual potential users of the interface. Doing so before a final round of code editing is to ensure all naming and structural conventions are correct and any important changes can be implemented before any other user tests. At this stage, most of the work done on *Glycarbo* was done on the 2D part, so this was the subject of testing.

Three domain experts performed unstructured usability testing. After a brief verbal introduction of the software and its capacity, one expert was asked to use the tool while the others observed. The tester answered any questions that arose but for the most part, observed the users' interactions with the interface in order to note any changes that need to made.

3.3.2 Task-Based User Tests

In order to improve quality of feedback from testing, the design of the user-tests was an important aspect of the testing phase. It is important to incorporate usability practices throughout the design process [10] and so user-tests were conducted prior to a final code edit, in order to incorporate user feedback into the final design. Two non-experts completed a semi-structured task-based test. The test allowed the users freedom to navigate the interface with very general instructions/tasks to complete. Again, the users were observed and their discussion recorded. The main aim of doing task-based user tests with non-

Issue Description	Type of Issue	Priority (1-3 with 1 being highest)	Difficulty (1-3 with 1 being most)
Incorrect Names (COOH, Alt2N etc.)	Functional	1	1
Unsupported residues/unable to build	Functional	1	1
More shorthand names needed	Non-Functional	3	3
Removing "Straighten" feature	Functional	2	3
Bonding needs to be redone	Functional	1	2
More obvious transition to 3D	Non-Functional	3	1
Needs to bond to middle (visually)	Non-Functional	3	3
Angles section of 3D builder a bit crowded	Non-Functional	3	2

Figure 4: Results from Initial Expert Inspection

experts was to highlight any obvious usability and flow issues. Additionally, users with no domain knowledge would have had no expectations of the software and would be able to provide unbiased feedback.

Again, the test mostly focused on the 2D builder. However, in this case, users also interacted with *Glycarbo's* 3D renderer.

3.3.3 Stress Testing

Stress testing was done in the form of generating specified glycans using the 2D builder. A list of ten glycans was provided from the database *GlycanDB* (another Honours project) and used to cover the use cases of *Glycano* and to generate an in-depth list of issues.

4 DOCUMENTATION OF CODEBASE

This phase consisted of adding the necessary documentation to *Glycano*'s code so that it can be maintained and edited as required. At the start of the project the codebase was 0% documented. This was vastly improved upon and each class now contains a description of its use and interactions with other classes. Important methods are preceded with a description of their functionality.

There was also a lack of in-depth build instructions for both *Glycano* and *Glycarbo* and so setting up the development environment was time-consuming. More depth was added to the build instructions, as well as development "shortcuts" learned along the way – in order to improve the efficiency of future work on these projects. A guide for future developers was also curated (see Appendix B). This includes a list of classes and methods to change based on the desired changes. This also contains information about previous versions of the codebase – should there be any need to reimplement functionality that has been removed.

5 REQUIREMENTS GATHERING

The results from the requirements gathering phase are tabulated above (figure 4). The issues that arose are all mainly related to issues of convention, bonding and some non-functional issues. The structural issues within the codebase overarch all of the above issues and were the main priority for the first phase of coding.

5.1 Structural Issues

A core function of *Glycarbo* is the use of the 2D builder to generate a Casper string that is used by the 3D builder. However, after the last iteration on *Glycarbo* most of the residue names were incorrect and thus this function was not operational. This made the task of correcting the residue names the highest priority. The way that *Glycano* was previously designed required a restructuring of the codebase in order to change the residue names. Previously the "UCT" convention was used. This allowed users to combine both residues and substituents to build a glycan. This meant that both the residue and substituent type dictated the shape and colour of residues on the canvas. Although this functionality is useful for some purposes, it is not suited for the purposes of *Glycarbo* – nor for the SNFG convention. The task of renaming the residues was made more complicated by the use of substituents as *Glycano*'s parser was programmed to expect a bond location before the substituent type – which was not compatible with SNFG. A class diagram of the former version of *Glycano* (Appendix A) demonstrates the redundant classes and functionality that were removed during this task- all classes in red were removed. The removal of these classes will open up opportunity for small, incremental changes to *Glycarbo* in the future, a good usability practice [6] as the codebase now relies less on the convention used and will be more flexible to change. Following the code clean- up, the SNFG naming convention was implemented throughout the system. The use of substituents was removed from all classes but will be easy to reimplement if needed. Functionality related to Residue Types (Aldoses, Ketoses, etc.) remains in the codebase for future use.

5.2 Bonding and Chemical Properties

Both usability and functional issues were identified during the initial examination when inspecting the bonding feature of *Glycano*. In previous versions, bonding was done by dragging a residue to the canvas and rotating the residue around until the correct bond location was found. This feature was largely faulty as not all linkage locations could be reached without overlapping residues. Bonds were done by finding the closest distance between residues and this limited user control. This feature was replaced with manual bonding in order to improve user-freedom and control and address the bonding difficulties experienced during the evaluation.

Some functionality related to improving the accuracy of the chemical properties of the molecules was not in place. Residues are joined by a glycosidic linkage from the anomeric carbon of one residue to a carbon on the other [3]. The structure of carbon molecules differs for each residue and thus residues have different linkage functionality – for example all residues except for ketones bond from carbon position one, while ketones bond from position two [2]. This functionality was not included in *Glycano*'s previous iterations and was thus added. Additionally, some resides have link locations to which bonds cannot be formed. This functionality was also added and the corresponding link locations were "blocked" to ensure no "illegal" residues were generated or passed into the 3D builder. Below (figure 5) shows the new bond design, which allows the user to click the link location they desire, while also displaying the locations that are blocked in red (specific to each residue).

Features of *Glycano* (2015) were removed when it was edited in 2020 and a mode known as "straighten" was added, which centered all bonds to the middle of the

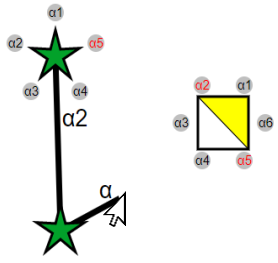


Figure 5: New Bonding Functionality

Issue Description	Type of Issue	Priority (1-3 with 1 being highest)	Difficulty (1-3 with 1 being most)
"Create bond" button not obvious	Non-Functional	2	3
Casper string incorrect when using repeats	Functional	1	1
Should have feature that makes all bonds same length/structures everything properly	Non-Functional	1	1
Want to be able to remove bonds added previously	Non-Functional	2	3
Want to be able to enter Casper String and 2D residue generated	Functional	3	1

Figure 6: Feedback from Expert Testing

residue shape. This functionality was made the default and the option to be in "build" or "straighten" mode removed.

5.3 Non-Functional Requirements

Prior to user testing there were still some minor "flow" issues that needed to be addressed. This included the bond labels being hidden at certain angles and an unclear call to action between 2D and 3D builders. These cosmetic flaws were fixed while user feedback was incorporated into the interface.

Initial Software Feasibility Demonstration

During the demonstration it was finalised that the primary concern of the project (after fixing the major functional issues) would be to improve the maintainability of *Glycarbo*'s codebase. The metrics for ensuring a developer-friendly code-base were discussed, including adequate build instructions, necessary method descriptions and comments throughout the code-base.

6 TESTING

6.1 Expert Testing

The feedback from the from the expert testing is tabulated above (figure 6).

In both user tests, there was no immediate call to action after the residues were placed on the canvas. This meant that the "create bond" button needs to be made more obvious.

The graph-building functionality, when using repeat residues, was not robust. The Casper string generated is correct only if the user bonds from begin repeats and to end repeats. For example, the Casper string for *Klebsiella pneumoniae* K2 shows up as [->3)bDGlc(1->4)[aDGlcA(1->3)]bDMan(1->4)[](1->6)]aDGlc when the user bonds from the end repeat, when it should be [->3)bDGlc(1->4)[aDGlcA(1->3)]bDMan(1->4)aDGlc(1->]. This was fixed so that the user did not need to bond "from right to left" in order to generate the correct Casper string.

A feature to make the 2D build uniform (in terms of bond length and alignment) was discussed as a very useful feature. This was added, and upon the user clicking "straighten", bonds are aligned at 45°,90° and 180°/0° degrees to their "parent" (the residue to which they are bonded) - depending on their original alignment. The bonds are also all set to a uniform length.

The functionality to remove/edit all bonds existed in the original version of *Glycano*. This was reimplemented in order to allow users easy reversal of errors [11].

It was also brought up that being able to enter a Casper string and generate a 2D glycan would be very useful. This is the same as the functionality of *Glycarbo* and is noted as potential for future work. It is possible that some of the graph-building logic used in *Glycarbo* could be used for this purpose.

The new bonding functionality was commented on as being much easier to use.

Task Number	Task	Feedback	Type of Issues
1	**Create 2D residue**	Couldn't see "add bond" button New bonding structure better than before Temporary bond doesn't make it obvious that you have to click link location Commented on previous straighten feature, nice to have option to align bonds	Non-Functional
2	**Export File as SVG**	No issues	-
3	**Change bond type**	Confusion with what changing bond type entails (refers to bond type of residue)	Non-Functional
4	**Use Casper String to build 3D residue**	No issues	-
5	**Create 2D residue and build 3D residue**	No clear call to action from 2D to 3D, did not know what to click	Non-Functional
6	**Change 3D representation**	Couldn't find "apply" button	Non-Functional
7	**Change linkage angles**	No "apply" button so not sure if anything changed	Functional

Figure 7: Feedback from Task-Based User Tests

Issue	Type of Issue	Plan
"Click Bond" should be clicked once and user should be able to make multiple bonds in one go	Non-Functional	Fix this feature if time allows
Export as .png is erratic, sometimes produces dark background	Functional	Fix this feature if time allows
Pyr cannot be generated with just SNFG residues	Functional	Note for future work
Bond disappears if residue we are bonding to has child residues	Functional	Fix this feature if time allows
Linkage type doesn't change if residue already selected	Functional	Fix this feature if time allows

Figure 8: Results from Stress-Testing

6.2 Task-Based User Tests

The feedback from the task-based user testing is tabulated above (figure 7). Similar to the expert tests the bonding functionality was noted as being useful and clear. However, the users found that having the bond line attached to the mouse didn't make it clear that the user needed to click a bond location. This has hence been edited so there is no visible bond until a bond location is clicked.

The non-experts also tested *Glycarbo*. No issues were faced when entering a valid Casper string and generating a 3D render. When asked to change the 3D representation, the "apply" button was not obvious enough.

The "load from builder" button also confused users.

6.3. Stress Testing

The results from the stress testing showed several minor issues in *Glycano*'s functionality. The results are tabulated in figure 8 above. Most issues that arose during stress testing were minor. Issues such as the bond disappearing and the linkage types not changing were easy to recover from by just re-bonding or by changing the linkage type using the residue overview panel that was added back to *Glycano*'s functionality following expert testing. These issues will be fixed if time allows. Adding the option to use furonose/pyronose should also be noted for future work on *Glycano*. Some functionality required to build certain *Klebsiella pneumoniae* bacteria is not available. This will be noted for future work. See Appendix C for the output of the stress testing (the ten *Klebsiella pneumoniae* glycans).

7 DISCUSSION

7.1 Code Improvements

7.1.1 Maintainability

As discussed, a primary concern about *Glycarbo* was its complexity and lack of maintainability. The aim with this project was to ensure *Glycarbo*'s future use and capacity to change. In cleaning and documenting the codebase of *Glycano*, editing *Glycarbo* will become a much smaller task, and it can hopefully change as user requirements grow. The documentation in the form of a class diagram - see Appendix A - will be extremely helpful to developers and researchers at UCT in the future, as well as the editing instructions -see Appendix B. Documentation within the code will help future developers to pin-point where edits need to be made without requiring a complete understanding of the entire codebase. The codebase for *Glycano* can be considered much more maintainable and readable.

7.1.2 Usability

The introduction of manual bonding to *Glycano* means that users now have much more control over the molecular and visual layout of 2D builds. This feature will hopefully make *Glycarbo* the choice tool for rapidly drawing and building residues as this is a rare feature. The option to edit previous bonds also promotes user control and freedom [11].

7.1.3 Functionality

As a result of Glycano's new bonding functionality and naming conventions, all Casper strings generated will be correct and usable by *Glycarbo*'s 3D renderer.

7.2 Testing

The use of focus group - like testing sessions was extremely useful in providing in- depth feedback and this structure should continue to be used when testing specialised software like *Glycarbo*.

Performing tests with two groups (with different levels of expertise) was also very helpful for maximising feedback. Not only but the quality of feedback was excellent, with the different groups finding different types of errors or design flaws.

8 CONCLUSIONS

The main aims for *Glycarbo* of maintainability, readability, functionality and improved usability have been achieved. The combination of adding documentation to all classes within *Glycano*'s codebase and providing editing guidance will streamline any future development on *Glycano*, and contribute to its maintainability and readability. Improving the build instructions by incorporating development tips will also speed up the development process. Removing the redundant classes will also simplify the process of understanding the

codebase. However, should the use of substituents be required in the future – it will be relatively easy to re-implement the functionality removed.

In terms of the functionality of *Glycarbo*, ensuring the naming conventions were correct allowed *Glycarbo* to operate at its full capacity and render any 3D glycans that can be built from residues within the SNFG set. Thus - the functionality of *Glycarbo* has been improved.

Improving the bonding functionality to allow better user control and also allow users to reverse bonding errors at any stage have made *Glycano* a much more usable and intuitive tool. There is room to improve the intuitiveness of the interface and make the flow of control more obvious to the user (such as making the "create bond" and "load from builder" buttons more obvious).

9 FUTURE WORK

As noted during expert testing, *Glycano* could become an even more useful tool if it had the capacity to take in a Casper string and generate its 2D glycan. This requires building a parser to build a graph and reverse-engineer a glycan. This can be proposed as another iteration of *Glycano*- DrawGlycanSNFG [4] has this functionality, but can be vastly improved upon. *Glycano* also has room to expand as more capabilities (eg. Furonose/Pyronose) and residues are added and could possibly become the tool of choice for populating visual glycan databases. *Glycano* and *Glycarbo* still face minor functionality and flow issues that could be improved upon.

REFERENCES

[1] Ajit Varki, Richard D Cummings, Markus Aebi et al. Symbol Nomenclature for Graphical Representations of Glycans, *Glycobiology*, Volume 25, Issue 12, December 2015, Pages 1323–1324, https://doi.org/10.1093/glycob/cwv091

[2] Albert L. Lehninger, David L. Nelson, and Michael M. Cox. (2017). *Chapter 7: Carbohydrates and Glycobiology*. Lehninger Principles of Biochemistry (7th Ed.) W. H. Freeman.

[3] *Classes of Monosaccharides.* (2019). Chapter 16: Carbohyrdates. 410 Health Chemistry. LibreTexts (Chemisty).

[4] DrawGlycan-SNFG. 2020. Render glycans and glycopeptides with fragmentation info. using the Symbolic Nomenclature for Glycans [SNFG]. Retrieved from http://www.virtualglycome.org/DrawGlycan/. Accessed May 2022.

[5] Felipe Albrecht, Peter Ebert and Markus List. (2017). Ten Simple Rules for Developing Usable Software in Computational Biology. PLOS Computational Biology. Volume 13(1), 1-5. https://doi.org/10.1021/acs.jctc.1c00169.

[6] GLYCAM Web. 2020 Complex Carbohydrate Research Center, University of Georgia, Athens, GA. Retrieved from http://glycam.org. Accessed May 2022

[7] *Glycano*. *Glycano*. Available at: http://*Glycano*.cs.uct.ac.za/

[8] Jennifer Preece, Yvonne Rogers, Helen Sharp. (2019). Chapter 13: Introducing Evaluation. *Interaction Design: Beyond Human-Computer Interaction (5th edition).* Wiley.

[9] Jennifer Preece, Yvonne Rogers, Helen. Sharp (2019). Chapter 7: Identifying needs and establishing requirements. *Interaction Design: Beyond Human-Computer Interaction (5th edition).* Wiley.

[10] Jonah Miller, Raniere Silva, Francisco Queiroz. 2017. Track 1 Paper: Good Usability Practices in Scientific Software Development. DOI: 10.6084/m9.figshare.5331814

[11] Laurisha Rampersaad, Sarah Blyth, Ed Elson and Michelle Kuttel, M. 2017. Improving the Usability of Scientific Software with Participatory Design: a new Interface Design for Radio Astronomy Visualisation Software. In Proceedings of SAICSIT'17, September 26- 28, Thaba Nchu, South Africa, 1-9. https://doi.org/10.1145/3129416.3129899

[12] Lavanya Ramakrishnan and Daniel Gunter. Ten principles for Creating Usable Scientific Software. (2017). IEEE 13TH International Conference on eScience. October 24-27, 2017, Auckland, New Zealand, 210-218. doi: 10.1109/eScience.2017.34.

[13] Markus Aebi, Richard D Cummings Ajit Varki et al. (2015). Glyco-Forum. Glycobiology. Volume 25, issue 12. Pages 1323 -1324. doi: 10.1093/glycob/cwv091.

[14] Michelle Kuttel et al. 2016. CarbBuilder: Software for building molecular models of complex oligo-and polysaccharide structures. Journal of computational chemistry. 37, 22 (2016), 2098–2105

[15] Naelah Al-Ageel, Areej Al-Wabil, Noura AlOmar and Ghada Badr. (2015). Human factors in the design and evaluation of bioinformatics tools. Procedia Manufacturing, Volume 3, 2003-2010. https://doi.org/10.1016/j.promfg.2015.07.247.

[16] PolysGlycanBuilder. A user friendly tool for to build 3D structures of complex glycan and polysaccharides. http://glycan-builder.cermav.cnrs.fr/. Accessed May 2022.

[17] Reed Milewicz, Paige Rodeghero et al. 2019. Position Paper: Towards Usability as a First-Class Quality of HPC Scientific Software. In Proceedings of the IEEE/ACM 14th International Workshop on Software Engineering for Science. May 28, 2019, Montreal, QC, Canada. 41-42. doi: 10.1109/SE4Science.2019.00012

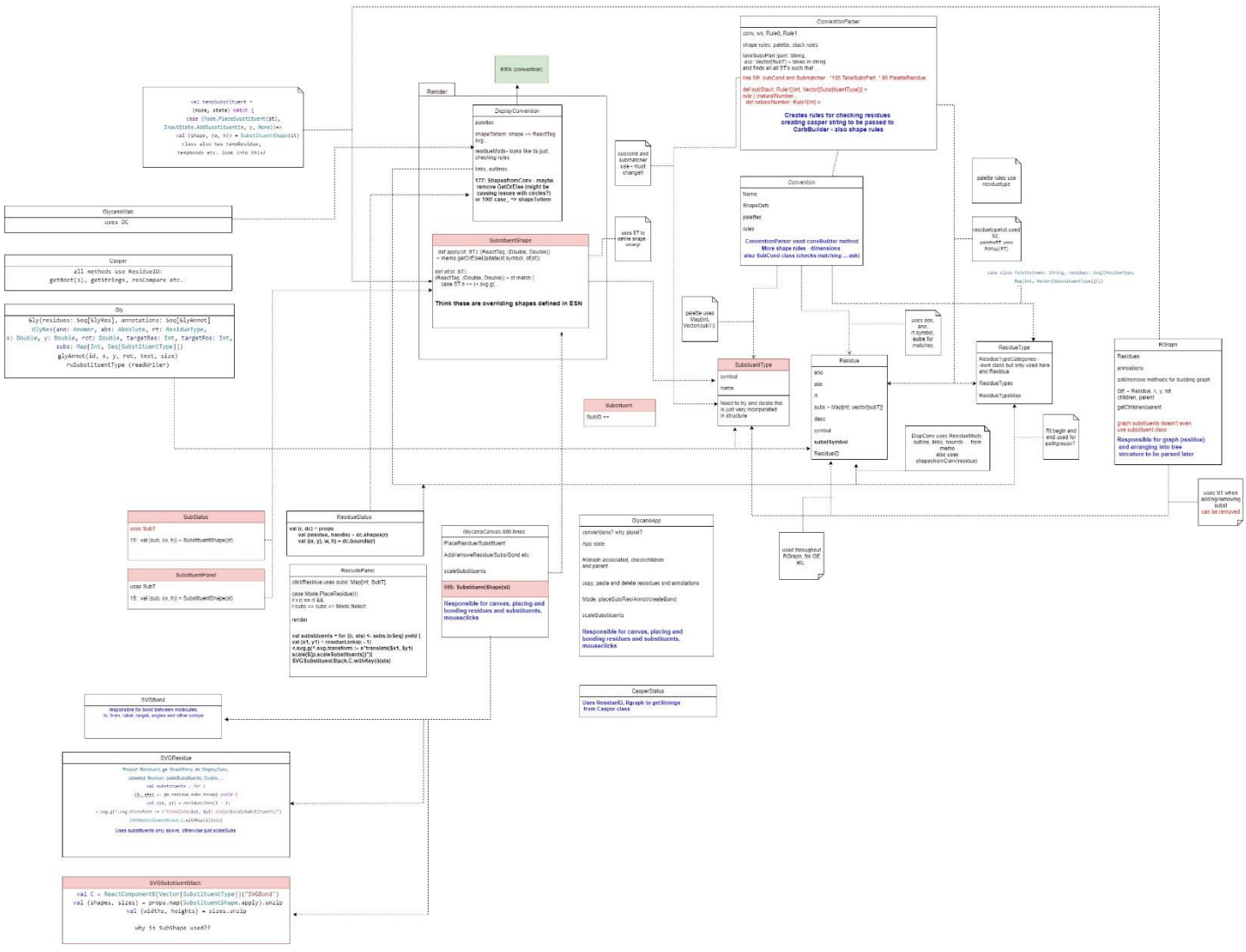
[18] Rena D. Astronomo and Dennis R. Burton. (2010). *Carbohydrate vaccines: developing sweet solutions to sticky situations?* Nature Reviews | Drug Discovery. Volume 9. Pages 308-324. DOI:

[19] Serge Perez and Kiyoko F. Aoki-Kinoshita. (2017). *Chapter 2: Development of Carbohydrate Nomenclature and Representation*. A Practical Guide to Using Glycomics Databases. Pages 7 -25. DOI 10.1007/978-4-431-56454-6_2

[20] Sofya I. Scherbinina and Philip V. Toukach. 2020. Three-Dimensional Structures of Carbohydrates and Where to Find Them. Int J Mol Sci, Volume 21(20), 1-46. Doi: https://dx.doi.org/10.3390%2Fijms21207702.

[21] Tristan Wood and Renee Panday. 2020. *Glycarbo*. *Glycarbo*.cs.uct.ac.za

Appendix A



Appendix B

Edit	**Difficulty** (1-3 with 1 being lowest)	Classes to change	Instructions
Add residue	1	SNFG ResidueType	**SNFG**: Add Residue to correct palette **ResidueType**: add Residue with linkage information to correct ResidueCategory
Edit residue linkage	1	SNFG ResidueType	**SNFG:** make sure link shape matches the linkage required **ResidueType:** change linkage in constructor
Edit shape or colour of residues	1	SNFG	**SNFG:** to change colour, go to bottom to "style" allocations, to change shape, look for the corresponding palette and change the [outline] or [links] to change the shape. [primary] and [secondary] refer to shapes with an internal shape as well.
Add blocked link for residue	1	Residue	**Residue:** add number of link to "blocked" for corresponding category
Add or change residue categories	1	ResidueType ResidueCategory	**ResidueType**: Add/change resiude category in class statement after "extends", create constructor for residues of that category (see lines 13-32), add trait and all residues in that category, add trait to ResidueTypes **ResidueCategory**: create/change residueCategory and add to ResidueCategories
Change bond functionality	2	GlycanoCanvas SVGBond SVGResidue	**GlycanoCanvas:** edit methods closestValidLink, closestValidLinkDsq (eg. To change bondthreshold) **SVGBond:** edit visual bond object **SVGResidue:** edit appearance of links

Appendix C