Essential Design Modeling for Scientific Software Solutions Development

Zeeshan Ahmed[1, 2, 3],

[1]**The Jackson Laboratory for Genomic Medicine, CT, USA**

[2]**University of Massachusetts, School of Medicine, MA, USA**

[3]**University of Wuerzburg, Department of Bioinformatics, USA**

Abstract

Software design and its engineering is essential for bioinformatics software impact. We propose a new approach 'Butterfly', for the betterment of modeling of scientific software solutions by targeting key developmental points: intuitive, graphical user interface design, stable methodical implementation and comprehensive output presentation. The focus of research was to address following three key points: 1) differences and different challenges required to change from traditional to scientific software engineering, 2) scientific software solution development needs feedback and control loops following basic engineering principles for implementation and 3) software design with new approach which helps in developing and implementing a comprehensive scientific software solution. We validated the approach by comparing old and new bioinformatics software solutions. Moreover, we have successfully applied our approach in the design and engineering of different well applied and published Bioinformatics and Neuroinformatics tools including DroLIGHT, LS-MIDA, Isotopo, Ant-App-DB, GenomeVX and Lipid-Pro.

Introduction

Computer Science has revolutionized almost all other fields of life. Common man including engineers, doctors, artists, technicians and scientist etc., somehow, every one's life is now partially depending on the usage of informatics. In the past (1980s), the informatics (IT) issues were related to the development of the large sized but small-scaled applications. Later on (1990s), with the passage of time systems started becoming complex but smaller in size, especially with the evolvement of the concept i.e. Component Based Systems (CBS) [1] and the innovations of advanced programming tools and technologies[2] e.g. Enterprise Java Beans, Microsoft COM and CORBA etc. So far the focus of the last decade (2000s) was to develop smart, intelligent and robotic applications.

Particularly in life science, with the front-runner field bioinformatics, the world has been changed by small, efficient, fast, logical, embedded and intelligent software, databases and management systems. Even this year's (2013) Nobel Prize winners (Arieh Warshel, Martin Karplus, Michael Levitt) [3] in the field of Chemistry relied on powerful computational programs to understand and predict biochemical processes and molecular dynamics, giving testimony to the novelty and innovation of bioinformatics.

Software Engineering Principles

To establish and expedite the processes of scientific software engineering (SSE), many Software Development Life Cycle (SDLC)[4] models have been introduced e.g. Waterfall Model, V-Model, Spiral Model, Iterative and Incremental Model, Rapid Prototype Model, Extreme Programming Model, Evolutionary Model, Agile Development Model, Code and Fix Model etc., and some other Process improvement models[5].

SDLC is a goal-oriented approach toward the software development. Almost all of the proposed SDLC models provide distinct processes for the software implementations. Depending upon the nature of the end product, the right model has to be chosen and applied. Based on the process' artifacts and logical steps for developing a software project (e.g. time, quality, size, development effort etc.), it is not easily possible to compare different SDLCs[6], but doing so reveals differences[7]. Own efforts did focus on quality improvement of software[8, 9, 10, 11, and 12].

Depending upon the observed commonalties, in general, we state that the software engineering is an integrated, cyclic and product line combination of following independent modular approaches: requirements engineering[13, 14, 15], design modelling[16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28], programming, testing and deployment.

These five modular approaches (Figure 1) follow the procedures of some life cycle management approaches, which can help them in performing their individual functionalities as well as regulating tasks in cyclic chain processes.

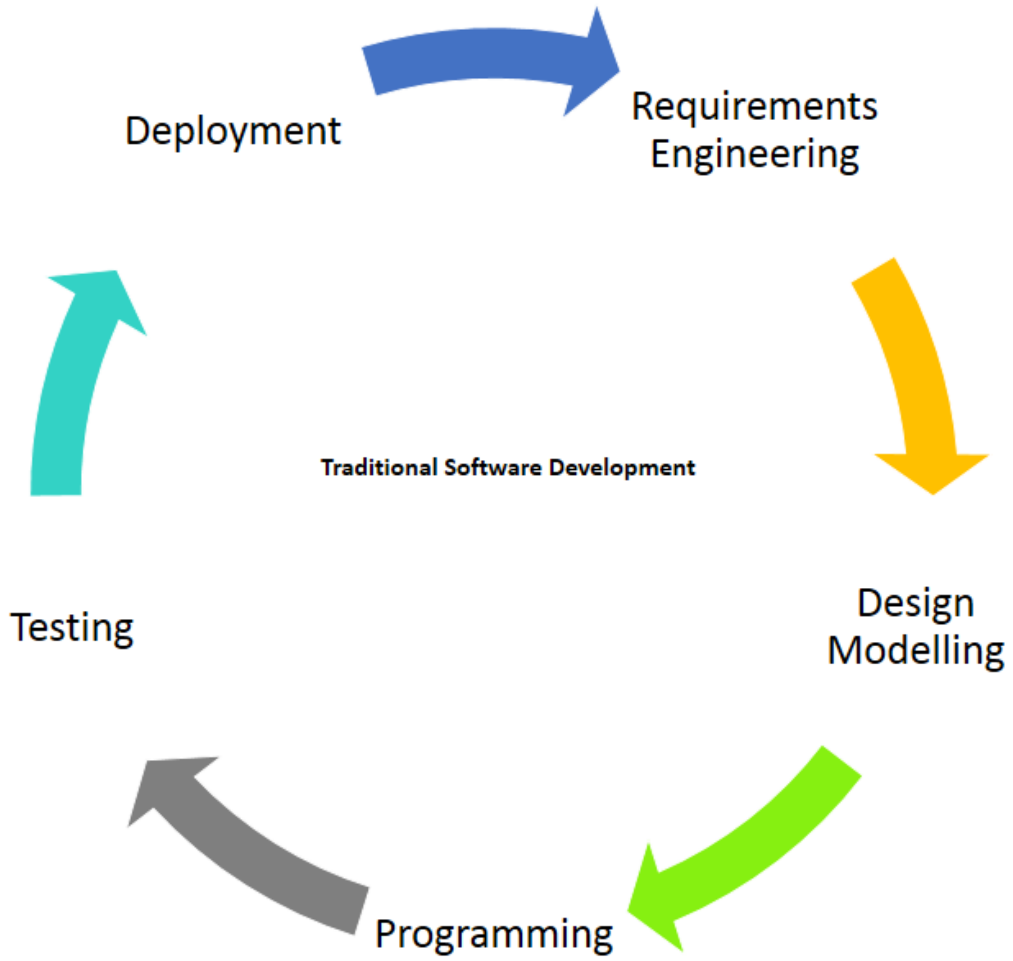


Figure 1. Traditional Software Development; consisting of integrated and cyclic combination of the following independent modular approaches: requirements engineering, design modeling, programming, testing and deployment [65, 66].

Scientific Software Engineering

Testing of integrated and individual modules becomes time consuming (Figure 2), as new test cases have to be rewritten all the times or the application exists with a high expectation of ripple effects [29] (i.e. unidentified logical or syntax errors in the system which arise while fixing the identified logical or syntax errors). The quality of a software application decreases with an increase in the ripples a change in software creates. Moreover measured optimum software maintenance can only be achieved with the accessibility of the concrete information about the ripples effect in the system [30]. Depending upon the nature of the system, many approaches have been proposed to improve the software quality measurement processes (e.g. [31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]etc.), towards the traditional software development but one can also use these in the scientific software solution's quality assurance and for improvements as well.

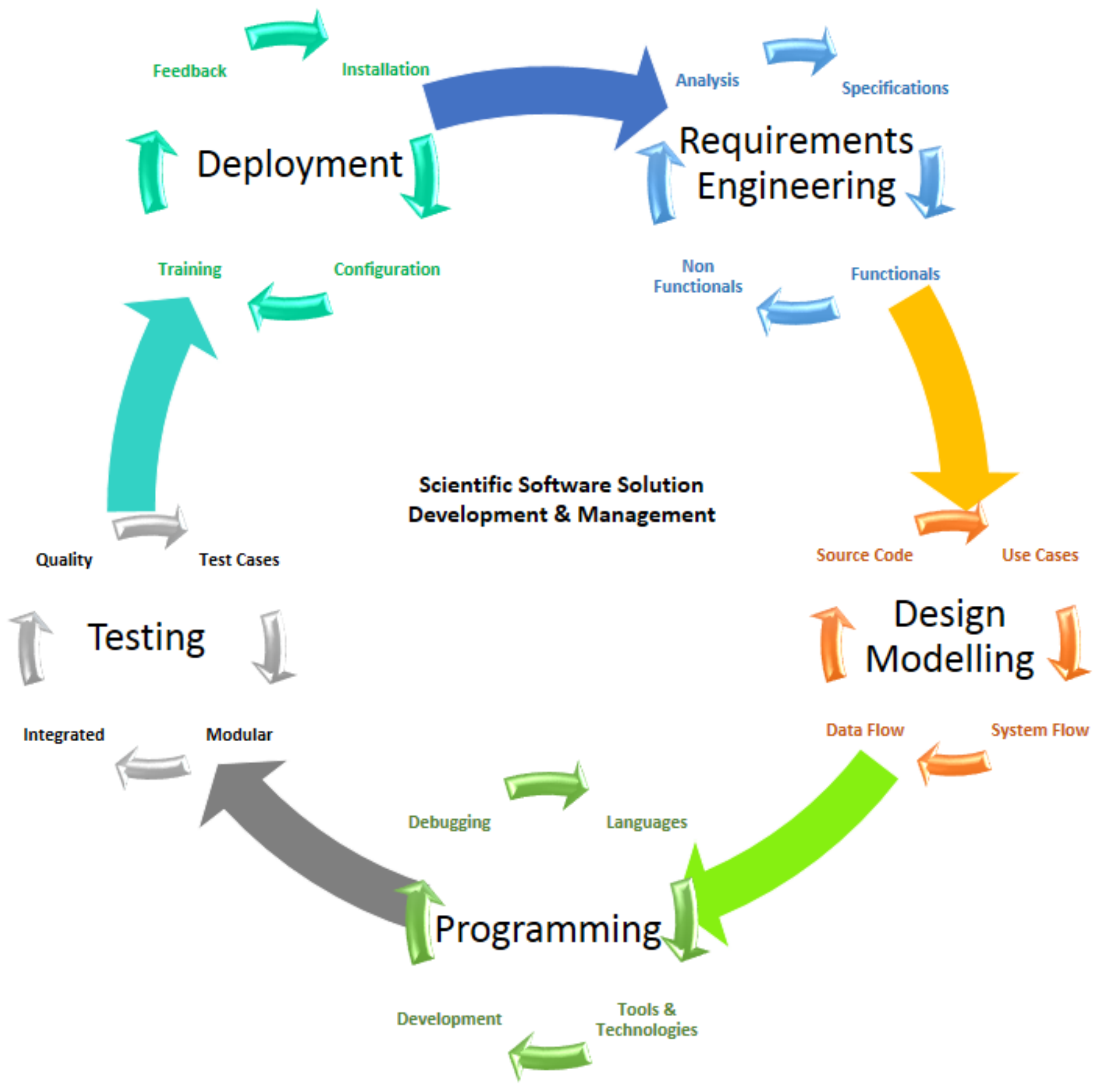


Figure 2. Scientific Software Engineering (SSE)[65, 66]. SSE integrates and combines in a development cycle the following independent main modular approaches: requirements engineering, design modeling, programming, testing and deployment. Each approach consists of its own sub-modular, integrated and cyclic combination of internal phases: requirement engineering consists of specification, functional, non-functional, and analysis; design modeling consists of use cases, system flows, data flow and source code; programming consists of languages, tools and technologies, development, and debugging; testing consists of test cases, modular, integrated and quality; finally, deployment consists of installation, con-figuration, training, feedback. Iterative cycles lead to continuous improvements, achievements translate the goals into good software.

Database manipulation and management system: If the target scientific software solution has this focus, then it requires to properly modeling the database schema (entity relationship model) by reducing the levels of data redundancy and dependency, using data normalization. There are five data normalization forms: 1NF, 2NF, 3NF, 4NF and 5NF, conceptual procedures for comprehensive database designing[44]. These data normalization forms help in shaping the data types (1NF), developing relationships between non-key and key fields (2NF, 3NF) [45, 46], and deals with multi-valued facts corresponding to the many to many relationships (4NF and 5NF) [47, 48].

Human Computer Interaction and Scientific Applications: The Human Computer Interaction (HCI), also known as Human Machine Interaction (HMI) [49, 50, 51], has to correlate with the Scientific Application development. HCI defines the implementation of the mechanisms to establish the efficient communication protocols between human and machines. These protocols are based on the textual, visual, sensory, audio and event based information, provided by both the user and the machine (computer).

Butterfly Workflow Design and Software Examples

To implement the Butterfly model [65, 66], we have designed a three-layered architecture (Figure 3), going from abstract planning (gray) to designers and developers (yellow) to implementation and user (green).

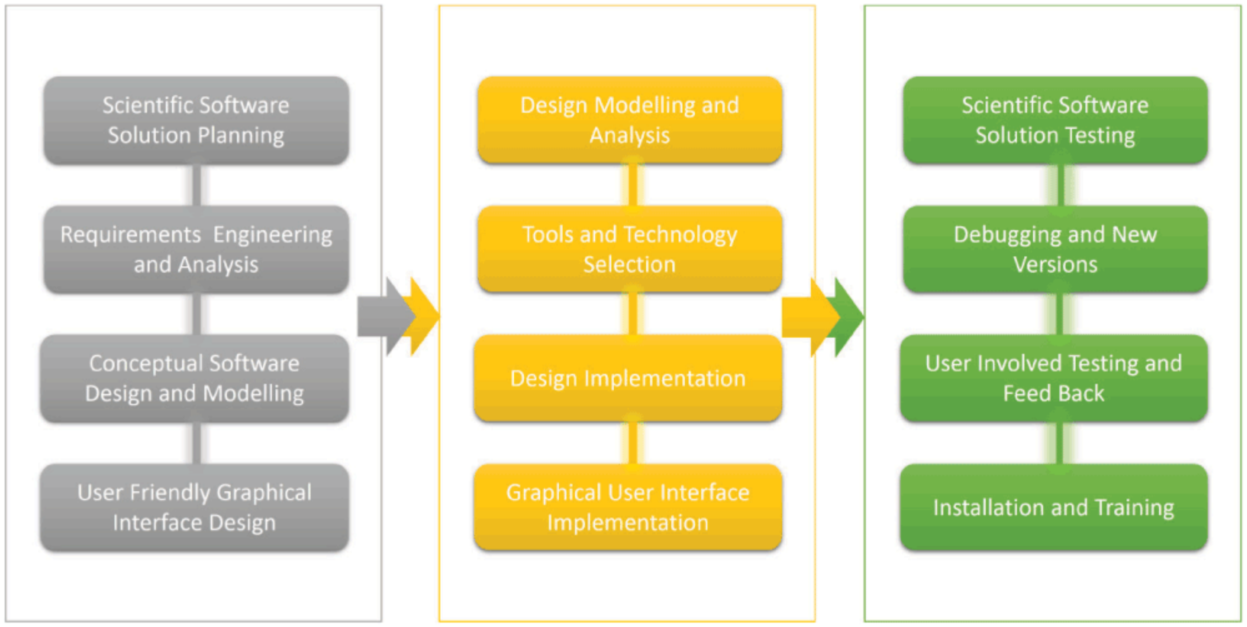

Scientific Software Solution Planning
Requirements Engineering and Analysis
Conceptual Software Design and Modelling
User Friendly Graphical Interface Design
Design Modelling and Analysis
Tools and Technology Selection
Design Implementation
Graphical User Interface Implementation
Scientific Software Solution Testing
Debugging and New Versions
User Involved Testing and Feed Back
Installation and Training

Figure 3. Butterfly three-layer model [65, 66]. Shown in grey is the abstract layer, the basis for design and development (yellow), followed by implementation and testing by the user (green) so that the software is released including installation and training.

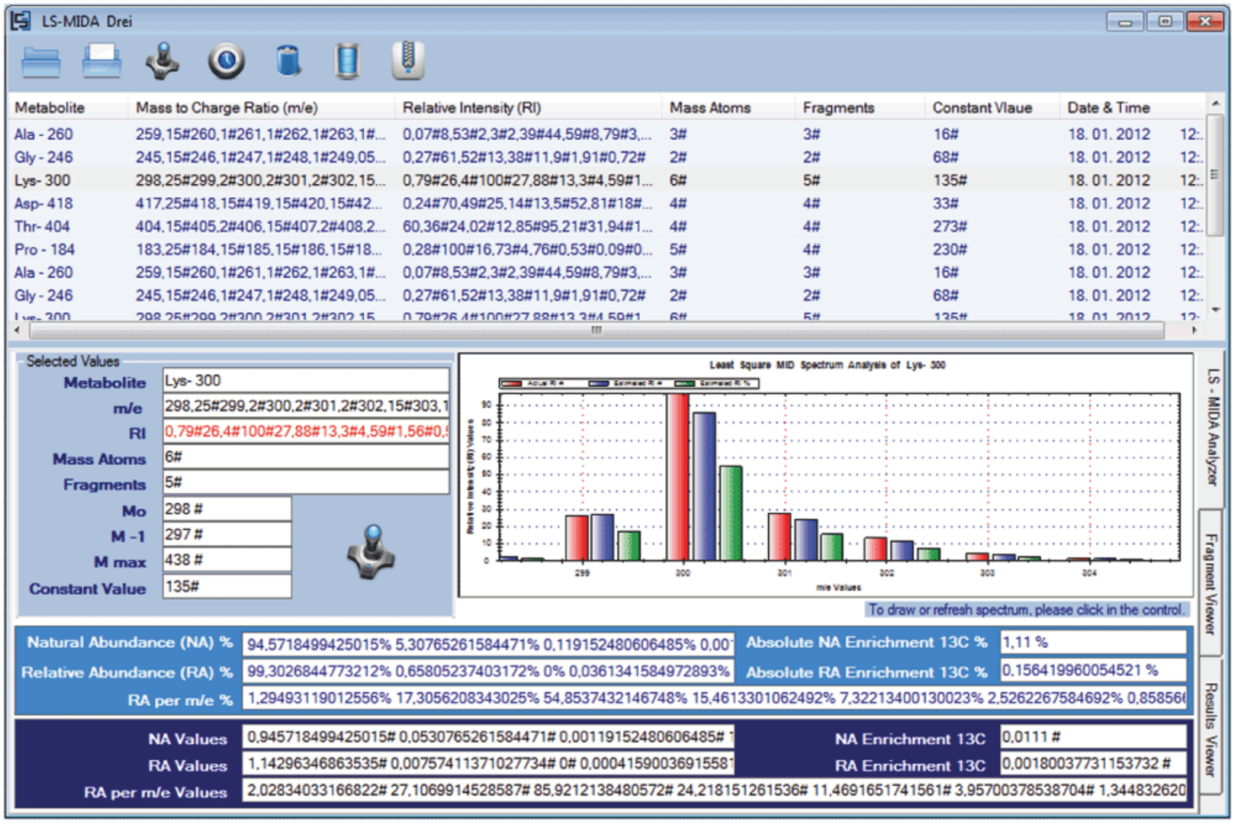
Abstract planning: Scientific software solution planning is the first step towards a new scientific application development. It requires good knowledge of the field (e.g. biochemistry, neurobiology, genetics, metabolomics, proteomics etc.) as well as project related information (e.g. what could be the end product, in-put to the system, expected output from the system, methodology, ideas, user opinions etc.). The next important phase is to perform requirements engineering and analysis. The third phase is the conceptual software design and modeling. Before moving ahead, first go for some abstract designs based on functional requirements and discuss these in your team. The last phase is software solution planning. It concerns the design of a user-friendly graphical interface.

Software design: This layer involves the designers and developers. It consists of four steps: design and modeling and analysis, tools and technology selection, design implementation and graphical user interface implementation.

Implementation: The last layer concerns implementation and programming and involves in house testing and debugging (by the developers and tester). Steps include scientific software solution testing, debugging and creation of new versions, users involved in testing and feedback and finally installation and training.

The Butterfly workflow design accentuates experience from previous software developments including a number of larger efforts (Table 1). Most of these are team efforts that simply have come close to the Butterfly paradigm, but by chance and pressure, not by explicitly following a scientific approach. With rapid development of new software applications, the need to formalize the software solution development principles increases to ensure that all scientific applications follow the standard scientific paradigms.

Adopting the concepts of Butterfly model, some new scientific software applications have already been proposed, designed, implemented, tested and are in use (LS-MIDA[52, 53] (Figure 4), DroLIGHT [54, 55, 56] (Figure 5), Isotopo [57] (Figure 6), Lipid-Pro [66] (Figure 7), App Ant Database[67] (Figure 8) and GenomeVX[68] (Figure 9).



LS-MIDA Drei

Metabolite	Mass to Charge Ratio (m/e)	Relative Intensity (RI)	Mass Atoms	Fragments	Constant Vlaue	Date & Time	
Ala - 260	259,15#260,1#261,1#262,1#263,1#...	0,07#8,53#2,3#2,39#44,59#8,79#3,...	3#	3#	16#	18. 01. 2012	12:.
Gly - 246	245,15#246,1#247,1#248,1#249,05...	0,27#61,52#13,38#11,9#1,91#0,72#	2#	2#	68#	18. 01. 2012	12:.
Lys- 300	298,25#299,2#300,2#301,2#302,15...	0,79#26,4#100#27,88#13,3#4,59#1...	6#	5#	135#	18. 01. 2012	12:.
Asp- 418	417,25#418,15#419,15#420,15#42...	0,24#70,49#25,14#13,5#52,81#18#...	4#	4#	33#	18. 01. 2012	12:.
Thr- 404	404,15#405,2#406,15#407,2#408,2...	60,36#24,02#12,85#95,21#31,94#1...	4#	4#	273#	18. 01. 2012	12:.
Pro - 184	183,25#184,15#185,15#186,15#18...	0,28#100#16,73#4,76#0,53#0,09#0...	5#	4#	230#	18. 01. 2012	12:.
Ala - 260	259,15#260,1#261,1#262,1#263,1#...	0,07#8,53#2,3#2,39#44,59#8,79#3,...	3#	3#	16#	18. 01. 2012	12:.
Gly - 246	245,15#246,1#247,1#248,1#249,05...	0,27#61,52#13,38#11,9#1,91#0,72#	2#	2#	68#	18. 01. 2012	12:.
Lys- 300	298,25#299,2#300,2#301,2#302,15	0,79#26,4#100#27,88#13,3#4,59#1	6#	5#	135#	18. 01. 2012	12:.

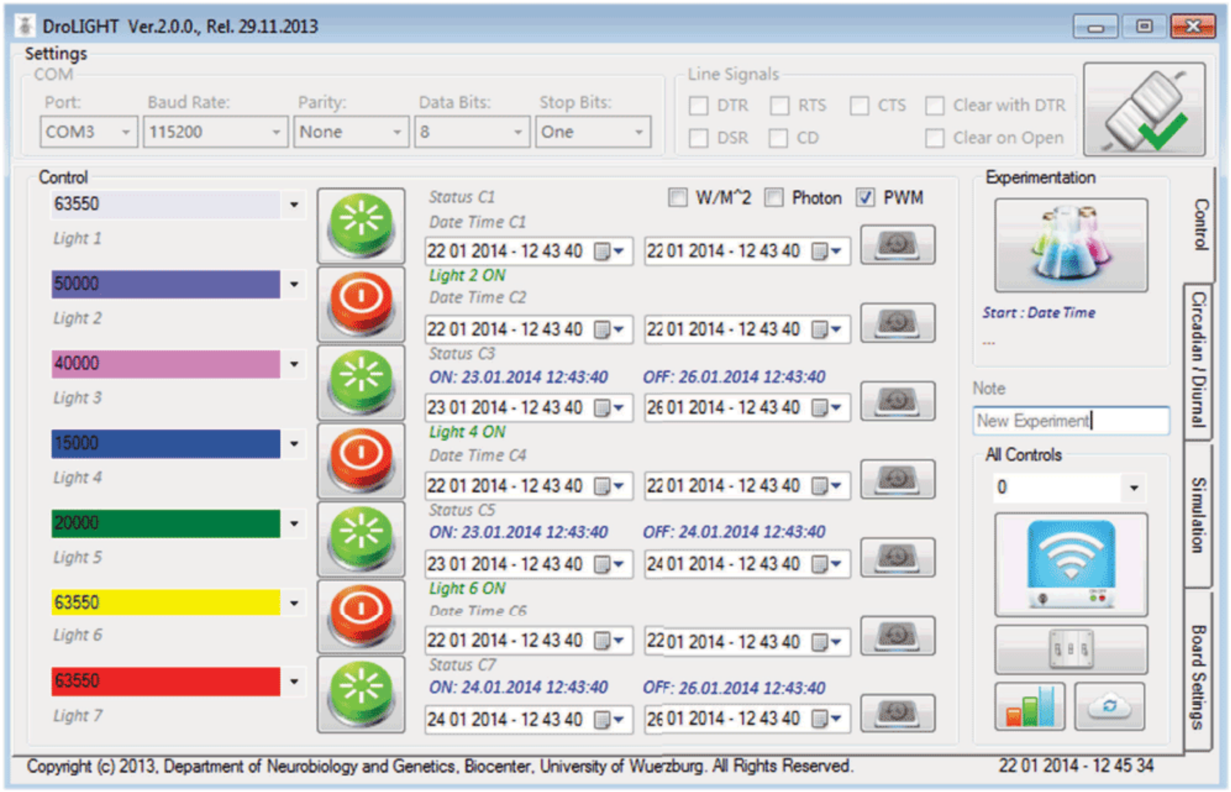
Selected Values

Metabolite	Lys- 300
m/e	298,25#299,2#300,2#301,2#302,15#303,1
RI	0,79#26,4#100#27,88#13,3#4,59#1,56#0,
Mass Atoms	6#
Fragments	5#
Mo	298 #
M -1	297 #
M max	438 #
Constant Value	135#

To draw or refresh spectrum, please click in the control.

Natural Abundance (NA) %	94,5718499425015% 5,30765261584471% 0,119152480606485% 0,00	Absolute NA Enrichment 13C %	1,11 %
Relative Abundance (RA) %	99,3026844773212% 0,65805237403172% 0% 0,0361341584972893%	Absolute RA Enrichment 13C %	0,156419960054521 %
RA per m/e %	1,29493119012556% 17,3056208343025% 54,8537432146748% 15,4613301062492% 7,32213400130023% 2,5262267584692% 0,85856		
NA Values	0,945718499425015# 0,0530765261584471# 0,00119152480606485#	NA Enrichment 13C	0,0111 #
RA Values	1,14296346863535# 0,00757411371027734# 0# 0,0004159003691558	RA Enrichment 13C	0,00180037731153732 #
RA per m/e Values	2,02834033166822# 27,1069914528587# 85,9212138480572# 24,218151261536# 11,4691651741561# 3,95700378538704# 1,344832620		

Figure 4. Graphical user interface of LS-MIDA



DroLIGHT Ver.2.0.0., Rel. 29.11.2013

Settings

COM

Port:	Baud Rate:	Parity:	Data Bits:	Stop Bits:
COM3	115200	None	8	One

Line Signals: DTR, RTS, CTS, Clear with DTR, DSR, CD, Clear on Open

Control

63550	Light 1	Status C1 / Date Time C1	22 01 2014 - 12 43 40	22 01 2014 - 12 43 40
50000	Light 2	Light 2 ON / Date Time C2	22 01 2014 - 12 43 40	22 01 2014 - 12 43 40
40000	Light 3	Status C3 / ON: 23.01.2014 12:43:40 / OFF: 26.01.2014 12:43:40	23 01 2014 - 12 43 40	26 01 2014 - 12 43 40
15000	Light 4	Light 4 ON / Date Time C4	22 01 2014 - 12 43 40	22 01 2014 - 12 43 40
20000	Light 5	Status C5 / ON: 23.01.2014 12:43:40 / OFF: 24.01.2014 12:43:40	23 01 2014 - 12 43 40	24 01 2014 - 12 43 40
63550	Light 6	Light 6 ON / Date Time C6	22 01 2014 - 12 43 40	22 01 2014 - 12 43 40
63550	Light 7	Status C7 / ON: 24.01.2014 12:43:40 / OFF: 26.01.2014 12:43:40	24 01 2014 - 12 43 40	26 01 2014 - 12 43 40

W/M^2 Photon [✓ PWM]

Experimentation

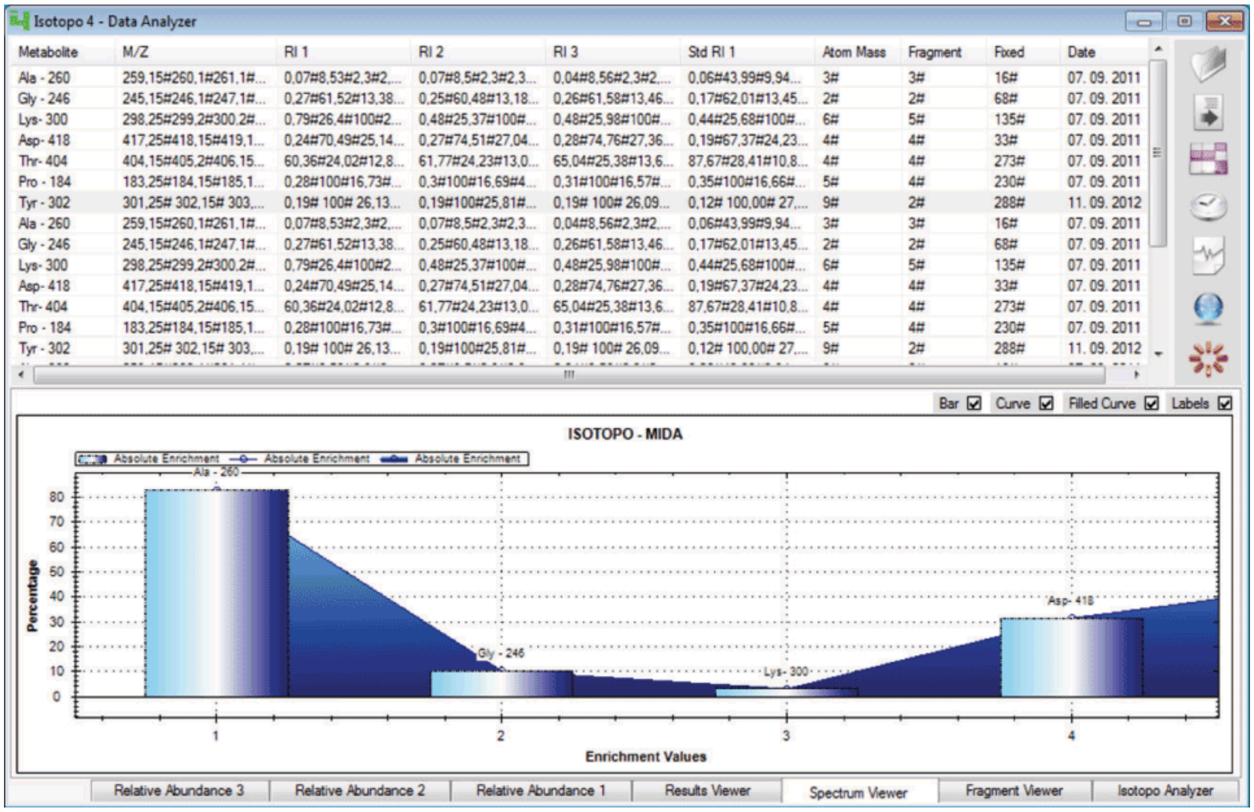
Start : Date Time

Note: New Experiment

All Controls: 0

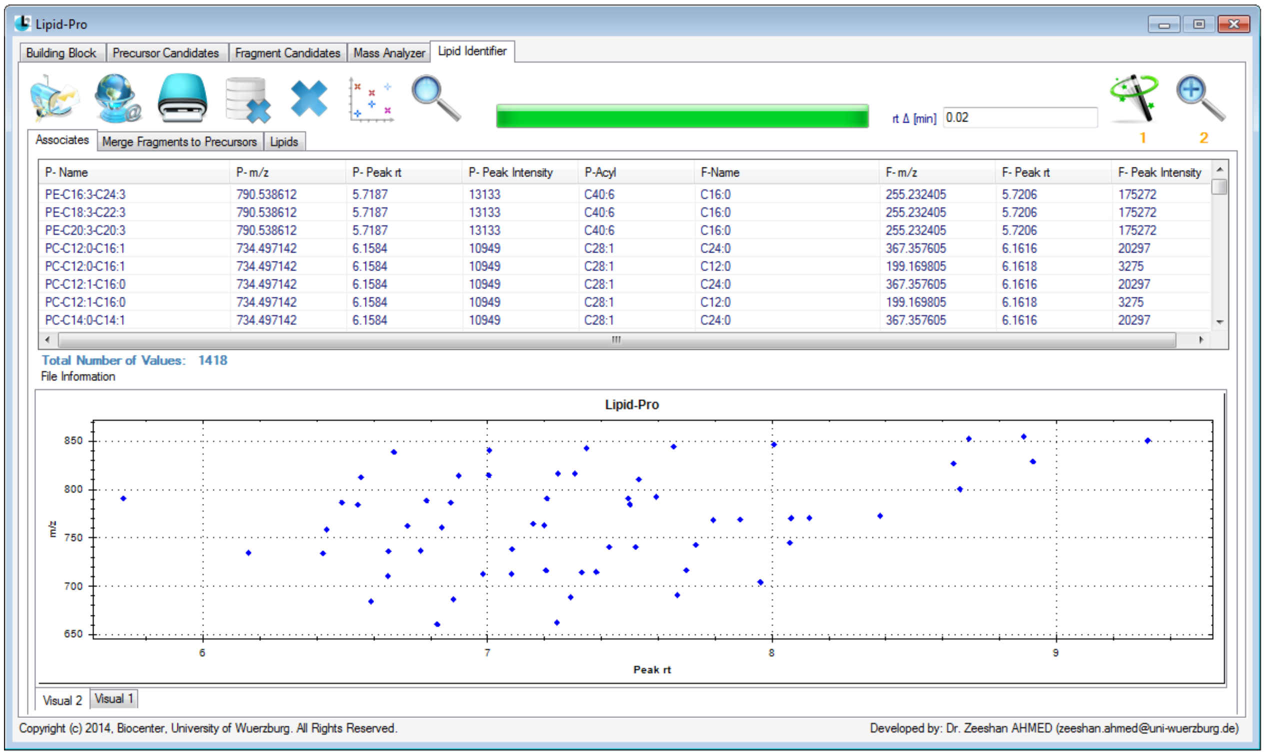
Copyright (c) 2013, Department of Neurobiology and Genetics, Biocenter, University of Wuerzburg. All Rights Reserved. 22 01 2014 - 12 45 34

Figure 5. Graphical user interface of DroLIGHT



Metabolite	M/Z	RI 1	RI 2	RI 3	Std RI 1	Atom Mass	Fragment	Fixed	Date
Ala - 260	259,15#260,1#261,1#...	0,07#8,53#2,3#2,...	0,07#8,5#2,3#2,3...	0,04#8,56#2,3#2,...	0,06#43,99#9,94...	3#	3#	16#	07. 09. 2011
Gly - 246	245,15#246,1#247,1#...	0,27#61,52#13,38...	0,25#60,48#13,18...	0,26#61,58#13,46...	0,17#62,01#13,45...	2#	2#	68#	07. 09. 2011
Lys- 300	298,25#299,2#300,2#...	0,79#26,4#100#2...	0,48#25,37#100#...	0,48#25,98#100#...	0,44#25,68#100#...	6#	5#	135#	07. 09. 2011
Asp- 418	417,25#418,15#419,1...	0,24#70,49#25,14...	0,27#74,51#27,04...	0,28#74,76#27,36...	0,19#67,37#24,23...	4#	4#	33#	07. 09. 2011
Thr- 404	404,15#405,2#406,15...	60,36#24,02#12,8...	61,77#24,23#13,0...	65,04#25,38#13,6...	87,67#28,41#10,8...	4#	4#	273#	07. 09. 2011
Pro - 184	183,25#184,15#185,1...	0,28#100#16,73#...	0,3#100#16,69#4...	0,31#100#16,57#...	0,35#100#16,66#...	5#	4#	230#	07. 09. 2011
Tyr - 302	301,25# 302,15# 303,...	0,19# 100# 26,13...	0,19#100#25,81#...	0,19# 100# 26,09...	0,12# 100,00# 27,...	9#	2#	288#	11. 09. 2012
Ala - 260	259,15#260,1#261,1#...	0,07#8,53#2,3#2,...	0,07#8,5#2,3#2,3...	0,04#8,56#2,3#2,...	0,06#43,99#9,94...	3#	3#	16#	07. 09. 2011
Gly - 246	245,15#246,1#247,1#...	0,27#61,52#13,38...	0,25#60,48#13,18...	0,26#61,58#13,46...	0,17#62,01#13,45...	2#	2#	68#	07. 09. 2011
Lys- 300	298,25#299,2#300,2#...	0,79#26,4#100#2...	0,48#25,37#100#...	0,48#25,98#100#...	0,44#25,68#100#...	6#	5#	135#	07. 09. 2011
Asp- 418	417,25#418,15#419,1...	0,24#70,49#25,14...	0,27#74,51#27,04...	0,28#74,76#27,36...	0,19#67,37#24,23...	4#	4#	33#	07. 09. 2011
Thr- 404	404,15#405,2#406,15...	60,36#24,02#12,8...	61,77#24,23#13,0...	65,04#25,38#13,6...	87,67#28,41#10,8...	4#	4#	273#	07. 09. 2011
Pro - 184	183,25#184,15#185,1...	0,28#100#16,73#...	0,3#100#16,69#4...	0,31#100#16,57#...	0,35#100#16,66#...	5#	4#	230#	07. 09. 2011
Tyr - 302	301,25# 302,15# 303,...	0,19# 100# 26,13...	0,19#100#25,81#...	0,19# 100# 26,09...	0,12# 100,00# 27,...	9#	2#	288#	11. 09. 2012

Figure 6. Graphical user interface of Isotopo



P- Name	P- m/z	P- Peak rt	P- Peak Intensity	P-Acyl	F-Name	F- m/z	F- Peak rt	F- Peak Intensity
PE-C16:3-C24:3	790.538612	5.7187	13133	C40:6	C16:0	255.232405	5.7206	175272
PE-C18:3-C22:3	790.538612	5.7187	13133	C40:6	C16:0	255.232405	5.7206	175272
PE-C20:3-C20:3	790.538612	5.7187	13133	C40:6	C16:0	255.232405	5.7206	175272
PC-C12:0-C16:1	734.497142	6.1584	10949	C28:1	C24:0	367.357605	6.1616	20297
PC-C12:0-C16:1	734.497142	6.1584	10949	C28:1	C12:0	199.169805	6.1618	3275
PC-C12:1-C16:0	734.497142	6.1584	10949	C28:1	C24:0	367.357605	6.1616	20297
PC-C12:1-C16:0	734.497142	6.1584	10949	C28:1	C12:0	199.169805	6.1618	3275
PC-C14:0-C14:1	734.497142	6.1584	10949	C28:1	C24:0	367.357605	6.1616	20297

Total Number of Values: 1418
File Information

Copyright (c) 2014, Biocenter, University of Wuerzburg. All Rights Reserved.
Developed by: Dr. Zeeshan AHMED (zeeshan.ahmed@uni-wuerzburg.de)

Figure 7. Graphical user interface of Lipid-Pro

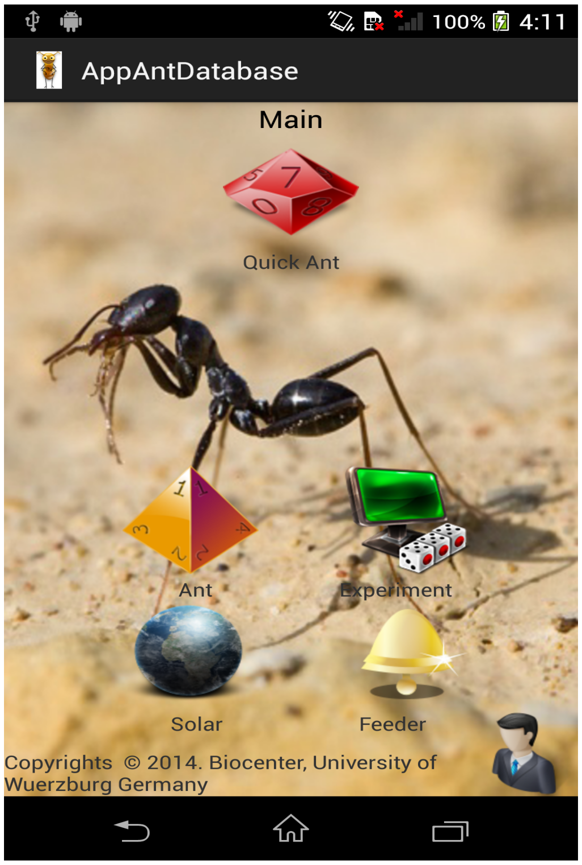
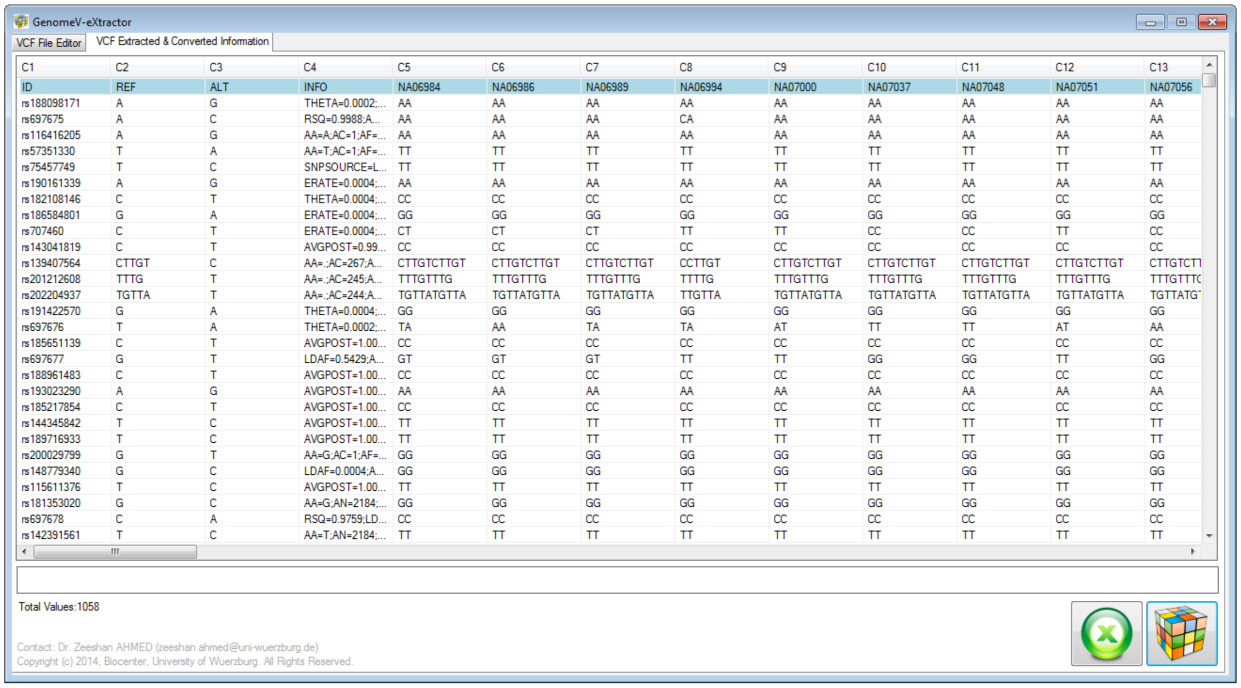

100% 4:11
AppAntDatabase
Main
Quick Ant
Ant
Experiment
Solar
Feeder
Copyrights © 2014. Biocenter, University of Wuerzburg Germany

Figure 8. Graphical User interface of Ant-App-DB



C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
ID	REF	ALT	INFO	NA06984	NA06986	NA06989	NA06994	NA07000	NA07037	NA07048	NA07051	NA07056
rs188098171	A	G	THETA=0.0002;...	AA	AA	AA	AA	AA	AA	AA	AA	AA
rs697675	A	C	RSQ=0.9988;A...	AA	AA	AA	CA	AA	AA	AA	AA	AA
rs116416205	A	G	AA=A;AC=1;AF=...	AA	AA	AA	AA	AA	AA	AA	AA	AA
rs57351330	T	A	AA=T;AC=1;AF=...	TT	TT	TT	TT	TT	TT	TT	TT	TT
rs75457749	T	C	SNPSOURCE=L...	TT	TT	TT	TT	TT	TT	TT	TT	TT
rs190161339	A	G	ERATE=0.0004;...	AA	AA	AA	AA	AA	AA	AA	AA	AA
rs182108146	C	T	THETA=0.0004;...	CC	CC	CC	CC	CC	CC	CC	CC	CC
rs186584801	G	A	ERATE=0.0004;...	GG	GG	GG	GG	GG	GG	GG	GG	GG
rs707460	C	T	ERATE=0.0004;...	CT	CT	CT	TT	TT	CC	CC	TT	CC
rs143041819	C	T	AVGPOST=0.99...	CC	CC	CC	CC	CC	CC	CC	CC	CC
rs139407564	CTTGT	C	AA=.;AC=267;A...	CTTGTCTTGT	CTTGTCTTGT	CTTGTCTTGT	CCTTGT	CTTGTCTTGT	CTTGTCTTGT	CTTGTCTTGT	CTTGTCTTGT	CTTGTCTT
rs201212608	TTTG	T	AA=.;AC=245;A...	TTTGTTTG	TTTGTTTG	TTTGTTTG	TTTTG	TTTGTTTG	TTTGTTTG	TTTGTTTG	TTTGTTTG	TTTGTTT
rs202204937	TGTTA	T	AA=.;AC=244;A...	TGTTATGTTA	TGTTATGTTA	TGTTATGTTA	TTGTTA	TGTTATGTTA	TGTTATGTTA	TGTTATGTTA	TGTTATGTTA	TGTTATG
rs191422570	G	A	THETA=0.0004;...	GG	GG	GG	GG	GG	GG	GG	GG	GG
rs697676	T	A	THETA=0.0002;...	TA	AA	TA	TA	AT	TT	TT	AT	AA
rs185651139	C	T	AVGPOST=1.00...	CC	CC	CC	CC	CC	CC	CC	CC	CC
rs697677	G	T	LDAF=0.5429;A...	GT	GT	GT	TT	TT	GG	GG	TT	GG
rs188961483	C	T	AVGPOST=1.00...	CC	CC	CC	CC	CC	CC	CC	CC	CC
rs193023290	A	G	AVGPOST=1.00...	AA	AA	AA	AA	AA	AA	AA	AA	AA
rs185217854	C	T	AVGPOST=1.00...	CC	CC	CC	CC	CC	CC	CC	CC	CC
rs144345842	T	C	AVGPOST=1.00...	TT	TT	TT	TT	TT	TT	TT	TT	TT
rs189716933	T	C	AVGPOST=1.00...	TT	TT	TT	TT	TT	TT	TT	TT	TT
rs200029799	G	T	AA=G;AC=1;AF=...	GG	GG	GG	GG	GG	GG	GG	GG	GG
rs148779340	G	C	LDAF=0.0004;A...	GG	GG	GG	GG	GG	GG	GG	GG	GG
rs115611376	T	C	AVGPOST=1.00...	TT	TT	TT	TT	TT	TT	TT	TT	TT
rs181353020	G	C	AA=G;AN=2184;...	GG	GG	GG	GG	GG	GG	GG	GG	GG
rs697678	C	A	RSQ=0.9759;LD...	CC	CC	CC	CC	CC	CC	CC	CC	CC
rs142391561	T	C	AA=T;AN=2184;...	TT	TT	TT	TT	TT	TT	TT	TT	TT

Total Values:1058

Contact: Dr. Zeeshan AHMED (zeeshan.ahmed@uni-wuerzburg.de)
Copyright (c) 2014, Biocenter, University of Wuerzburg. All Rights Reserved.

Figure 9. Graphical User interface of GenomeVX

To determine in a more objective way potential gains from scientific software de-sign following our proposed butterfly paradigm we have performed a short comparative analysis of some bioinformatics software applications (C13 [58], Metatool [59], BioOpt [60], FiatFlux [61] ReMatch [62], Biolayout [63], LS-MIDA [52, 53], DroLIGHT [54, 55, 56], Isotopo [57]), describing their type, methodology, implementation, user friendliness, configuration etc., based on the provided, published information (summarized in Table 2).

Table 1. Comparative analysis of different Bioinformatics software applications.

Butterfly area Software	Engineering Approach	Scientific Methodology	Scientific Application	Human Computer Interaction	Reference
BLAST	Scientific Software Engineering	Advanced (2 Hit method)	followed	Intuitive	S. F. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman (1990). Basic local alignment search tool. J. Mol. Biol. 215, 403-410.
FASTA	Traditional	1 Hit method	Steps to scientific application	Simple Command line	Pearson, W.R. & Lipman, D.J. (1988) "Improved tools for biological sequence comparison." Proc. Natl. Acad. Sci. USA 85:2444-2448.
Genbank	Traditional, but work in a team, iterative refinement	SQL	World-wide multiuser scenario	Command Line, web interface, BioPerl, SOAP, differ. downloads	Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW. GenBank. Nucleic Acids Res. 2010;38:D46-D51.
EBI databank	Traditional but work in a team, iterative refinement	XML	World-wide multiuser scenario	Command Line, SOAP, BioPerl, many download options	Kulikova T, Akhtar R, Aldebert P, Althorpe N, Andersson M, Baldwin A, Bates K, Bhattacharyya S, Bower L, Browne P, et al. EMBL nucleotide sequence database in 2006. NAR 2007;35:D16-D20.
EMBOSS	Traditional, but work in a team	AJAX Command Definition (ACD files) ANSI C	General software design rules for knowledgeable users	´Jemboss´, Java based Interface	Rice,P. et al. (2000) EMBOSS: the european molecular biology open software suite. TIG, 16, 276–277. Carver, T. J. and Mullan, L. J. (2002), Website Update: A new graphical user interface to EMBOSS. Comp Funct Genom, 3: 75–78. doi: 10.1002/cfg.136
Bioperl suite	Traditional	Perl	Followed general software design rules for knowledgeable users	Command Line	Stajich JE, Block D, Boulez K, Brenner SE, Chervitz SA, Dagdigian C, Fuellen G, Gilbert JG, Korf I, Lapp H, Lehväslaiho H, Matsalla C, Mungall CJ, Osborne BI, Pocock MR, Schattner P, Senger M, Stein LD, Stupka E, Wilkinson MD, and Birney E. The Bioperl toolkit: Perl modules for the life sciences. Genome Res. 2002 Oct;12(10):1611-8.
KEGG databank	Scientific Software Engineering	Oracle	Bottom-up and top-down effort, coordinated by Prof. Kanehisa	Charts, maps, Intuitive user interface, Export options, new KEGG api is restricted	Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., and Kanehisa, M.; KEGG: Kyoto Encyclopedia of Genes and Genomes. NAR. 27, 29-34 (1999). Kanehisa M, Goto S, Sato Y, Kawashima M, Furumichi M, Tanabe M. Data, information, knowledge and principle: back to metabolism in KEGG. Nucleic Acids Res. 2013
COPASI	Traditional	C++	General software design	Copasi GUI Command Line	Hoops S., Sahle S., Gauges R., Lee C., Pahle J., Simus N., Singhal M., Xu L., Mendes P. and Kummer U.

			rules for expert s	(CopasiSE)	(2006). COPASI: a COmplex PAthway SImulator. Bioinformatics 22, 3067-74.
COBRA	Traditional but large-scale team effort, multiple user-feedback	COBRA For Python COBRA Toolbox For MATLAB		Command Line Options	Ebrahim A, Lerman JA, Palsson BO, Hyduke DR. 2013 COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Syst Bio 7:74. Schellenberger J, Que R, Fleming RMT, Thiele I, Orth JD, Feist AM, Zielinski DC, Bordbar A, Lewis NE, Rahmanian S, Kang J, Hyduke DR, Palsson BØ. 2011 Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. Nature Protocols 6:1290-1307.
Rasmol	Traditional	C	Simple In PDB file Submissio n		Sayle, R. and Bissell, A. (1992) RasMol: A Program for Fast Realistic Rendering of Molecular Structures with Shadows. In Proceedings of the 10th Eurographics UK 1992 Conference, University of Edinburgh
Pymol	Traditional small team	C and Python		Command Line Options	The PyMOL Molecular Graphics System, Version 1.5.0.4 Schrödinger, LLC.
GROMOS package	Traditional but team work	FORTRA N77			W. R. P. Scott, P. H. Hünenberger, I. G. Tironi, A. E. Mark, S. R. Billeter, J. Fennen, A. E. Torda, T. Huber, P. Krüger, and W. F. van Gunsteren, The GROMOS Simulation Package, J. Phys. Chem. A 103 (1999) 3596-3607.

Table 2. Comparative analysis of different scientific software applications.

Application s / Comparativ e Measures	**C13**	**Metatool**	**BioOp t**	**Fiatlu x**	**ReMatc h**	**Biolayou t**	**LS-MIDA**	**Dro-LIGHT**	**Isotopo**
SSE?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
App. Type	Desktop	Desktop	Deskto p	Deskto p	Web	Desktop	Deskto p	Desktop	Desktop
Data Manageme nt	No DM Sys.	No DM Sys.	No DM Sys.	No DM Sys.	DB	No DM Sys.	File based	File based	File based and DB
Script or Prototype	Script	Script	Prototy pe	Script	Prototy pe	Prototyp e	Prototy pe	Prototy pe	Prototy pe
Algorithm Type	Parallel	Sequenti al	Sequen tial	Paralle l	Sequent ial	Parallel	Sequen tial	Parallel	Sequent ial
Algorithm / Methodolog y	Isotopic Labellin g	Schuster Algorith m	Mass Balanc e Equati on	Isotopi c Labelli ng	Carbon Mappin g	Markov Clusterin g	Least Square	Circadia n Rhythm s	Partial Least Square
Running Mode	Interacti ve	Interactiv e	Batch	Interac tive	Interacti ve	Interactiv e	Interac tive	Interacti ve	Interacti ve
Publishing, licensing	Publish ed, Free	Publishe d, Free	Publis hed,	Publis hed,	Publish ed, Free	Publishe d, Free	Publis hed,	Publish ed, Free	Publish ed, Free

			Free	Free			Free		
SDLC Information	Not Provided	Not Provided	Not Provided	Not Provided	Not Provided	Not Provided	V-Model	Spiral	V-Model
HCI Information	Not Provided	Not Provided	Not Provided	Not Provided	Not Provided	Not Provided	HCI Patterns Implemented	HCI Patterns Implemented	HCI Patterns Implemented
User Friendly	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Easy to configure	No	No	No	No	Yes	Yes	Yes	Yes	Yes
Easy to train	No	No	No	No	No	No	Yes	Yes	Yes
Software Re-Engineering	Yes	No	Yes	Yes	Yes	No	Yes	No	No
Cyclic or Repetitive	No	No	Yes	Yes	Yes	No	Yes	No	Yes
Easy to learn and Use	Yes	Yes	No	No	Yes	No	Yes	Yes	Yes
User Training	No	No	No	No	No	No	No	Yes	No

Conclusions

Conscious adaptation of SSE principles as exemplified here by the suggested butterfly design and its multilayered architecture, might look like an increase in developmental workload in comparison to many current bioinformatics programming methods. However, on the long run, it will reduce the burden by making the scientific application well designed, flexible, structured and reusable. It allows a product line development, is analytical and allows qualitative software improvement. Furthermore, HCI concepts make it user friendly, easy to learn and to deploy.

Conflict of Interests

The authors declare no conflict of interest.

References

1. Mahmood S, Lai R. A Component-Based System Requirements Analysis Language. Comput. J. Vol. 2013; 56: 901-922.
2. Szyperski C, Gruntz D, Murer S. Component Software—Beyond Object-Oriented Programming, Addison-Wesley, 2002.
3. The computer - your Virgil in the world of atoms. http://www.nobelprize.org/nobel_prizes/chemistry/laureates/2013/press.html (12 November 2013, date last accessed)
4. Boehm B. Software Engineering. IEEE Trans. on Com., 1976; 12: 1226-1242.
5. Rook P. Controlling Software Projects. Software Engin. J., 1986; 1: 7-16.
6. Benediktsson O, Dalcher D, Thorbergsson H. Comparison of software development life cycles: a multiproject experiment. IEE Proc. Soft., 2006; 153: 87-101.
7. Munassar NMA, Govardhan A. A Comparison Between Five Models Of Software Engineering. Int. Jr. Comp. Sci., 2010; 7: 94-101.
8. Ahmed Z. Towards Performance Measurement and Metrics based Analysis of PLA Applications. Int. J. Soft. Engin. App., 2010; 1: 66-80.

9. Ahmed Z, Majeed S. Measurement, Analysis with Visualization for better Reliability. Artificial Intelligence and Hybrid Systems: iConcept Press, 2013.
10. Ahmed Z, Majeed S. Towards Increase in Quality by Preprocessed Source Code and Measurement Analysis of Software Applications. IST Tran. Inf. Tech. Theo. App., 2010; 1: 8-13.
11. Ahmed Z. Measurement Analysis and Fault Proneness Indication in Product Line Applications (PLA). In: Sixth International Conference on New Software Methodologies, Tools, and Techniques, Rome, Italy, 2007; 391-400.
12. Ahmed Z. Integration of variants handling in M-system NT. Blekinge Institute of Technology, Karlskrona, Sweden. 2006.
13. Lee J, Xue NL. Analyzing user requirements by use cases: a goal-driven approach. IEEE Soft., 1999; 16: 92–101.
14. Sommerville I. Integrated requirements engineering: a Tutorial. IEEE Soft., 2005; 22: 16–23.
15. van Lamsweerde A, Darimont R, Letier R. Managing conflicts in goal-driven requirements engineering. IEEE Trans. Soft. Eng., 1998; 24: 908–926.
16. Kaur H, Singh P. UML (Unified Modeling Language): Standard Language for Software Architecture Development. In: International Symposium on Computing, Communication, and Control, Singapore, 2011.
17. Garlan D, Shaw M. An introduction to software architecture. Adva. Soft. Eng. Know. Eng., 1993; 2: 1-39.
18. Garlan D. Formal Approaches to Software Architecture. In: Workshop on Studies of Software Design, UK, 1993; 64-76.
19. Garlan D, Notkin D. Formalizing design spaces: Implicit invocation mechanisms. In: Fourth International Symposium of VDM Europe on Formal Software Development, UK, 1991; 31–44.
20. Dashofy EM, Hoek A, Taylor RN. An infrastructure for the rapid development of XML-based architecture description languages. In: Twenty Fourth International Conference on Software Engineering, USA, 2002; 266-276.
21. Egyed A, Kruchten PB. Rose/Architect: A Tool to Visualize Architecture. In: Thirty Second Annual Hawaii Conference on Systems Sciences, USA, 1999; 8066.
22. Booch G, Rumbaugh J, Jacobson I, Unified Modeling Language User Guide. In: the 2nd Edition, Addison-Wesley Professional, 2005.
23. Jacobson I, Christerson M, Jonsson P, Oevergaard G. Object-Oriented Software Engineering: A Use Case Driven Approach. In: Reading, MA Addison-Wesley, 1992.
24. Dumas M, ter-Hofstede AHM. UML Activity Diagrams as a Workflow Specification Language, In: Fourth International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, UK, 2001, 76-90.
25. Bruza PD, van-der-Weide TP. The Semantics of Data Flow Diagrams, In: International Conference on Management of Data, 1993.
26. Latronico E, Koopman P. Representing Embedded System Sequence Diagrams as a Formal Language, In: Fourth International Conference on The Unified Modeling Language, Canada, 2001; 302-316.
27. Marilyn B, A guide for programmers, Prentice-Hall, 1978.
28. Berardi D, Calvanese D, Giacomo G E. Reasoning on UML class diagrams. Artif. Intell., 2005; 168: 70-118.
29. Haney F M. Module connection analysis- a tool for scheduling of software debugging activitie. In: Proceedings of Fall Joint Computer Conference, 1972; 173-179.
30. Moreton R. A Process Model for Software Maintenance. J. Info. Tech., 1990; 5: 100-104.
31. Kan SH, Basili VR, Shapiro LN. Software Quality: An overview from the perspective of total quality management, IBM Sys. J., 1994; 33.
32. Li W, Henry S. An Empirical Study of Maintenance Activities in Two Object-oriented Systems, J. Soft. Main. Res. Prac., 1995; 7: 131-147.
33. Pfleeger SL, Bohner SA. A Framework for Software Maintenance Metrics. In: the proceedings of Conference on Software Maintenance, 1990; 320-327.
34. Moreton R. A Process Model for Software Maintenance. J. Info. Tech., 1990; 5: 100-104.
35. Soong NL. A program stability measure. In: the proceedings of Annual ACM conference, Boulder Colorado, 1977; 163-173.
36. 36 Yau SS, Collofello JS, McGregor TM. Ripple effect analysis of software maintenance. In: the Proceedings of Annual International Computers, Software & Applications Conference, 1978; 60-65.
37. Black S. Automating ripple effect measurement. In: the 5th World Multiconference on Systemics, Cybernetics and Informatics, Florida, USA, 2001.
38. Davis A. Software Requirements: Analysis and Specification. Prentice-Hall, New Jersey, 1989.
39. Martin J, McClure C. Software Maintenance: The Problem and its Solutions. Prentice- Hall, London, 1983.

40. Parikh G. Some Tips, Techniques and Guidelines for Program and System Maintenance. In: Winthrup Publishers, Cambridge, Mass., 1982; 65-70.
41. Sharpley W K. Software Maintenance Planning for Embedded Computer Systems. In: the Proceedings of the Annual International Computers, Software & Applications Conference, 1977, 520-526.
42. Osborne W M. Building and Sustaining Software Maintainability. In: the proceedings of Conference on Software Maintenance, 1987; 13-23.
43. Yau S S, Collofello J S. Some Stability Measures for Software Maintenance, IEEE Trans. Soft. Eng., 1980; 6: 545-552.
44. Kent W. A simple guide to five normal forms in relational database theory. Commun. ACM, 1983; 26: 120-125.
45. Codd EF. Normalized data base structure: A brief tutorial. In: ACM SIG- FIDET Workshop on Data Description, Access, and Control. Nov. 1971.
46. Codd EF. Further normalization of the data base relational model. In: IBM Research Report, San Jose, California RJ909, 1971
47. Fagin R. Multivalued dependencies and a new normal form for relational databases. ACM Trans. Database Syst., 1977; 2: 262-278.
48. Fagin R. Normal forms and relational database operators. In: ACM SIG- MOD International Conference on Management of Data, 1979.
49. Ahmed Z, Ganti S K, Kyhlbäck H, Design Artifact's, Design Principles, Problems, Goals and Importance. In: Fourth International Conference of Statistical Sciences, Pakistan, 2008; 57-68.
50. Ahmed Z. Designing Flexible GUI to Increase the Acceptance Rate of Product Data Management Systems in Industry. Int. J. Comp. Sci. Emerg. Tech., 2011; 2: 100-109.
51. Klemmer SR, Lee B. Notebooks that Share and Walls that Remember: Electronic Capture of Design Education Artifacts. In: ACM Symposium on User Interface Software and Technology, 2005.
52. Ahmed Z, Zeeshan S, Huber C et al. Software LS-MIDA for efficient Mass Isotopomer Distribution Analysis. BMC Bioinf., 2013; 14.
53. Ahmed Z, Majeed S, Dandekar T. Unified Modeling and HCI Mockup Designing towards MIDA. Int. Jr. Emerg. Sci., 2012; 2: 361-382.
54. Ahmed Z, Helfrich-Förster C, Dandekar T. Integrating Formal UML Designs and HCI Patterns with Spiral SDLC in DroLIGHT Implementation. Rec. Pat. Comp. Sci., 2013; 6: 58-98.
55. Ahmed Z, Helfrich-Förster C. DroLIGHT: Real Time Embedded System towards Endogenous Clock Synchronization of Drosophila, Front. Neuroinf., 2013.
56. Ahmed Z, Helfrich-Förster C. DroLIGHT-2: Real Time Embedded and Data Management System for Synchronizing Circadian Clock to the Light-Dark Cycles, Rec. Pat. Comp. Sci., 2013; 6. [Accepted]
57. Ahmed Z, Majeed S, Dandekar T. Formal UML Modelling of Isotopo, Bioinformatical Software for Mass Isotopomers Distribution Analysis. Soft. Eng., 2012; 2: 147-159.
58. Wiechert W, de Graaf AA. Bidirectional reaction steps in metabolic networks: I. Modeling and simulation of carbon isotope labeling experiments. Biotechnol Bioeng 1997; 55: 101-117.
59. Schuster R, Schuster S. Refined algorithm and computer program for calculating all non-negative fluxes admissible in steady states of biochemical reaction systems with or without some flux rates fixed. Comput Appl Biosci 1993; 9: 79-85.
60. Cvijovic M, Olivares-Hernández R, Agren R et al. BioMet Toolbox: genome-wide analysis of metabolism. Nucleic Acids Research 2010; 38: 144-149.
61. Zamboni N, Fischer E, Sauer M. FiatFlux - a software for metabolic flux analysis from 13C-glucose experiments. BMC Bioinformatics 2005; 6: 209.
62. Pitkänen E, Akerlund A, Rantanen A et al. ReMatch: a web-based tool to construct, store and share stoichiometric metabolic models with carbon maps for metabolic flux analysis. Journal of Integrative Bioinformatics 2008; 5: 1-13.
63. Klamt S, von Kamp A. An application programming interface for CellNetAnalyzer. Biosystems 2002; 105: 162-8.
64. Ahmed Z, Zeeshan S, Dandekar T. Developing sustainable software solutions for bioinformatics by the "Butterfly" paradigm. F1000Research. 3:71.
65. Ahmed Z, Zeeshan S. Cultivating Software Solutions Development in the Scientific Academia. Recent Patents on Computer Science. 7:1.
66. Ahmed Z, Michel M, Saman Z, Dandekar T, Mueller M J, Fekete A. Lipid-Pro: A computational lipid identification solution for untargeted lipidomics on data-independent acquisition tandem mass spectrometry platforms. Bioinformatics. 31:7, 2015.

67. Ahmed Z, Saman Z, Fleischmann P, Roessler W, Dandekar T. Ant-App-DB: A smart Solution for Monitoring the Arthropods' Activities, Experimental Data Management and Solar Calculations without GPS in Behavioural Field Studies. F1000Research. 3:311.
68. Ahmed Z, Saman Z, Peschel n, Dandekar T. GenomeVX: Bioinformatics Solution towards Understanding the Genome-Wide Comparative Analyses of Different Human Populations Front. Neurosci. Conference Abstract: Neuroinformatics 2015. doi: 10.3389/conf.fnins.2015.91.00059