Bioinformatics
Software
Engineering

Delivering Effective Applications

Paul Weston

Woodcock Stewart Ltd
Bioinformatics
Software
Engineering
Bioinformatics
Software
Engineering

Delivering Effective Applications

Paul Weston

Woodcock Stewart Ltd
Contents

Preface ix
List of figures xi
Acknowledgements xiii

Part 1 Introduction 1

1 What You Need to Know 3
The software development process: what is it you do all day? 3
The application lifecycle: from concept to construct 4
How to deliver effective bioinformatics applications 6

2 What Is Software Engineering? 9
Software engineering: applying formal rules to creativity 9
The benefits of discipline: reproducibility, stability, solidity, reliability 11
A limitation of engineering 11
On the edge: agile computing and evolutionary development 12

Part 2 Before Beginning 15

3 Project Definition 17
What you have got to do, in one sentence? 17
When have you got to deliver? 18
How will success be measured? 18
Who will measure success? 19
Overall vision? Expected results? Required functionality?
    You need them all 19
Work with what you have 20
4 Requirements Capture
When your customers are talking to you about what they want, stay focused
User stories
Avoid ‘blue-skying’ in meetings – summarize, don’t improvise
Pencil > keyboard: screen drawings
You will need to know what your sample datasets and expected results will be
Yes, now is the time to start thinking about how to test for success
Documents describing staged deliverables need to be agreed to, if not formally signed off
Get requirements capture right at this stage, so that top-priority requirements can go into the functional specification

5 Separating Function, Interface and Implementation
User stories will tell you most of the functionality that is required
From screen drawings to screens
Your implementation has to create and connect the interface and the functions

6 Implementation Considerations
Languages
Platforms
Operating systems
Prototyping tools
Debugging tools

7 Proof of Concept, Prototyping and Buy-in
Explaining the development process with prototypes and proofs of concept
‘Too much information! Too much information!’
Markets: how to sell yourself and your work, and why you need to
HyperCard, genetic algorithms, evolutionary development and ‘doing science’
Using adaptive development to encourage customer buy-in

Part 3 Getting it Done

8 Data in, Data out and Data Transformation
Begin, process, end: process flow diagrams
Boxes in boxes: data structure diagrams
9  Where to Start? 63
   You will need to get in data 64
   You will need to cope with errors, because they will happen 64
   You will need to report results 65
   Known and unknown coding 67
   From designs to pseudocode 67
   From pseudocode to code 71

10 Functional, then Optimized 73
   Get it out of the door, and in front of your customers, as soon
   and as often as safely possible 73
   Planning shows 74
   When not to display work in progress – getting them used to
   having it 75
   Customer retention and repeat buyers 76
   Optimization: benchmarking, assessment, refinement, hardware 77
   Focus on essentials, because your customers will 78

11 Coding Style 79
   Don’t write it if you don’t have to 79
   KISS – Keep It Simple, Stupid 80
   When you look at your code, does it make your eyes hurt, or does
   it look like poetry? 80
   Naming of parts 81
   Going gracefully into the darkness 82
   Prevention is better than cure 83
   Flexibility is vital 83
   Keeping track of what you are doing 84

Part 4  For Some Values of Done . . . 87

12 Writing the Friendly Manual 89
   Start as you mean to go on: make your code clean and clear 89
   Using what you have done so far: it’s half-written already 90
   Writing documentation from functional specifications,
   and vice versa 91
   Ask yourself, and others, who still needs to know what? 91

13 Testing – What and When 93
   Test plans 93
   Writing a test plan 94
   Expect the unexpected 95
   Why developers shouldn’t be their own testers 95
   Involving users in the testing process 96
   Some things to watch out for 97
   When to stop testing 99
So, what is this book all about, then?

Real bioinformatics – building adaptable tools to process, manage, analyze and display biological information – requires not just an appreciation of the underlying science, but also the ability to write efficient computer programs. *Bioinformatics Software Engineering* aims to build on these basic capabilities. It explains the principles and techniques that will help you to give your customers the software they need.

*Bioinformatics Software Engineering* shows you how you can save time and trouble by doing the right thing, at the right time, in the right way. It provides practical, helpful information that you can apply to your own work.

This book goes beyond coding to examine the whole project lifecycle. It guides you through the application development process from start to finish, without tying you to a particular methodology, paradigm, or platform. I hope it will give you the confidence to take on new challenges.

*Bioinformatics Software Engineering* covers:

- Agreeing requirements
- Design tools and techniques
- Defining appropriate documentation
- Hints and tips on development and coding
- Test planning and execution
- Rollout and delivery
• Support, maintenance and enhancement
• Project signoff

If some of these terms seem unfamiliar, don’t worry. They are explained at the start of the next chapter, and are discussed in more detail throughout the book.

Who is the book written for?

Both bioinformaticians looking to enhance their development skills, and scientists seeking to understand more about how the applications they use are created, should benefit from this book.

How do I find what I need to know?

Bioinformatics Software Engineering is arranged according to the stages in the development lifecycle, so you can read it straight through, or go to the section that covers your problem area.

Why did you write this book? And who are you, anyway?

I have tried to write the book that I didn’t realize I needed when I began my computing career. I hope that it helps you avoid some of the mistakes I made.

I started developing software tools for scientists at the UK’s Human Genome Mapping Project Resource Centre in 1996. I have also project-managed the development of an online game, and then been European Technical Consultant for a San Diego-based distributed computing company.

As well as building tools, I now provide consultancy and training in application development and software engineering, and I have a particular interest in genetic algorithms. Outside work, I have been a rhythm guitarist and songwriter for twenty years, I enjoy gardening, and I have followed Liverpool Football Club since 1987. And as far as George and Lillie are concerned, I’m just Dad.

Paul Weston
List of Figures

Figure 4.1 Screen drawing: desktop after program installation 26
Figure 4.2 Screen drawing: program start-up screen 26
Figure 4.3 Screen drawing: simple dialogue box 27
Figure 4.4 Screen drawing: selecting options 27
Figure 4.5 Screen drawing: confirming the choice 28
Figure 4.6 Screen drawing: program finish screen 28
Figure 8.1 Process flow diagram: linear process 56
Figure 8.2 Process flow diagram: repeated process 57
Figure 8.3 Process flow diagram: selection process 57
Figure 8.4 Process flow diagram: complex process 58
Figure 8.5 Data structure diagram: simple assembly structure 61
Figure 8.6 Data structure diagram: complex assembly structure 62
Figure 9.1 Process flow diagram: looping until success or exit 66
Figure 9.2 Process flow diagram: report construction 68