

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 <i>do</i> 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
<i>What</i> you have got to do, in one sentence?	17
<i>When</i> have you got to deliver?	18
<i>How</i> will success be measured?	18
<i>Who</i> will measure success?	19
Overall vision? Expected results? Required functionality?	
You need them all	19
Work with what you have	20

4	Requirements Capture	21
	When your customers are talking to you about what they want, stay focused	21
	User stories	22
	Avoid 'blue-skying' in meetings – summarize, don't improvise	24
	Pencil > keyboard: screen drawings	24
	You will need to know what your sample datasets and expected results will be	29
	Yes, now <i>is</i> the time to start thinking about how to test for success	30
	Documents describing staged deliverables need to be agreed to, if not formally signed off	30
	Get requirements capture right at this stage, so that top- priority requirements can go into the functional specification	31
5	Separating Function, Interface and Implementation	33
	User stories will tell you most of the functionality that is required	33
	From screen drawings to screens	37
	Your implementation has to create and connect the interface and the functions	37
6	Implementation Considerations	39
	Languages	39
	Platforms	40
	Operating systems	40
	Prototyping tools	42
	Debugging tools	42
7	Proof of Concept, Prototyping and Buy-in	45
	Explaining the development process with prototypes and proofs of concept	45
	'Too much information! Too much information!'	46
	Markets: how to sell yourself and your work, and why you need to	47
	HyperCard, genetic algorithms, evolutionary development and 'doing science'	48
	Using adaptive development to encourage customer buy-in	50
Part 3	Getting it Done	53
8	Data in, Data out and Data Transformation	55
	Begin, process, end: process flow diagrams	55
	Boxes in boxes: data structure diagrams	59

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 <i>not</i> 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

14	Rollout and Delivery	101
	You will need separate development, test and production environments	102
	Delivery notes	104
	Verbal handovers	105
	Building the installation package	105
	Instant gratification mode	106
	The useful/stable balance	107
15	Support and Feedback	109
	When do you start?	109
	Pre-emptive support	109
	What are your local customs?	110
	Development costs v. support costs	110
	Watch out for unanticipated feedback requiring urgent priority readjustment	111
16	Planned and Unplanned Enhancements	113
	Good applications are <i>never</i> finally finished	113
	Things not to say in meetings, No. 94: 'Oh, you found <i>that</i> one ...'	114
	'It's not a bug, it's an opportunity to further enhance the user experience'	114
	Big shoes: managing change in the workplace	114
	Priority 2: It'll be along real soon now, when we're all less busy	115
	Slippage	115
17	Project Signoff	117
	Dealing with bad stuff: focusing on the successfully delivered objectives	118
	Identifying potential endpoints and agreeing an exit strategy	119
	Where now?	119
	Index	121