ACCU 2010 Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Bonus:

John Lakos – The Birthday Problem

Kevlin Henney – Birthday Surprise

ACCU 2010 Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Anti-Parrallelism

Why and Where have "We" Gone Wrong™: A Personal, Passionate and Pragamatic Polemic.

Jason McGuiness[1]

[1]accu2010@hussar.demon.co.uk

ACCU, 2010

Planning To Fail...

- Because "the boss says so".
  - The incumbents know best, don't they? They wrote the code!
  - Did they enjoy using "mangletons"?
    - Shared-state is evil!
    - In the C++ code I've seen, I've not yet seen a properly implemented singleton!
  - Test cases? "We've heard of 'em." (We Don't Use Them 'Coz We're Not Whimps)
- Actually the code-base is truly evil, and not your friend!
  - Grok this, my friend.

More False Prophets...

- Because "it's too slow and parallelism will make it faster" say the bearded-gurus...
  - It's a magic bullet, isn't it?
  - *Of course* we believe the lore that it is slow due to not being parallel. So naturally we won't measure.
  - *Assumptions are the root of all evil!*

But But I Want To, *Please*...

- Because we can or just fancy doing it - we're geeks and it's just *too cool*. *;-}P>*
- Come on: writing parallel code makes you a guru, doesn't it?
  - You'll earn more, and all your desires will be satisfied by an adoring team and better still - management!!
  - Anthony Williams (of just::thread & boost.thread fame) is looked up to as one of the Great Gurus, isn't he?
    - I'm not taking his name in vain, just making the point that good thread libraries are hard to come by and harder to write, and he's doing not one but *two*.

For Further Reading I

- Intel "Thread Building Blocks".
- OpenMP in MSCV 2010
- "Efficient Parallel Algorithms" Gibbons & Rytter.
- "Parallel Algorithms" Casanova, Legrand & Robert.


http://www.justthread.com/


http://www.boost.org/


http://libjmmcg.sf.net/

ACCU 2010 Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

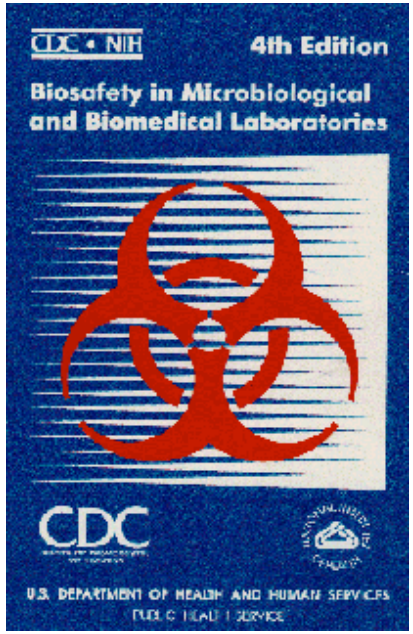
Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Researching Risky Software

- Many people research malware, but there are no widely accepted protocols.
- **Biological research has defined levels with associated practices, safety equipment, and facilities.**
- **Some approaches are**
  - **Weakened programs (auxotrophs)**
  - **Programs that ALERT**
  - **Outgoing firewalls**
  - **Isolated networks**



ACCU 2010 Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The "Also" Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

ACCU 2010 Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

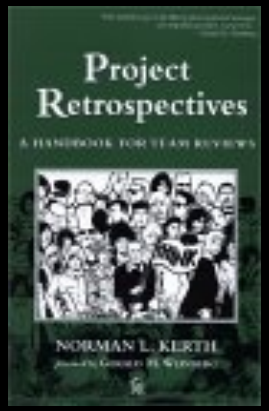
Learn from Experience with Retrospectives

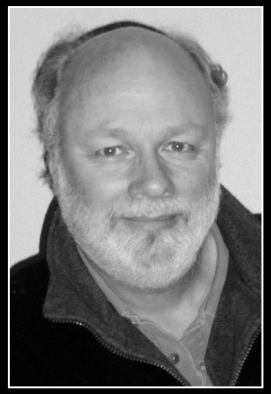
Rachel Davies
rachel@agilexp.com

Sankofa bird - “go back + fetch it”

What is a Retrospective?

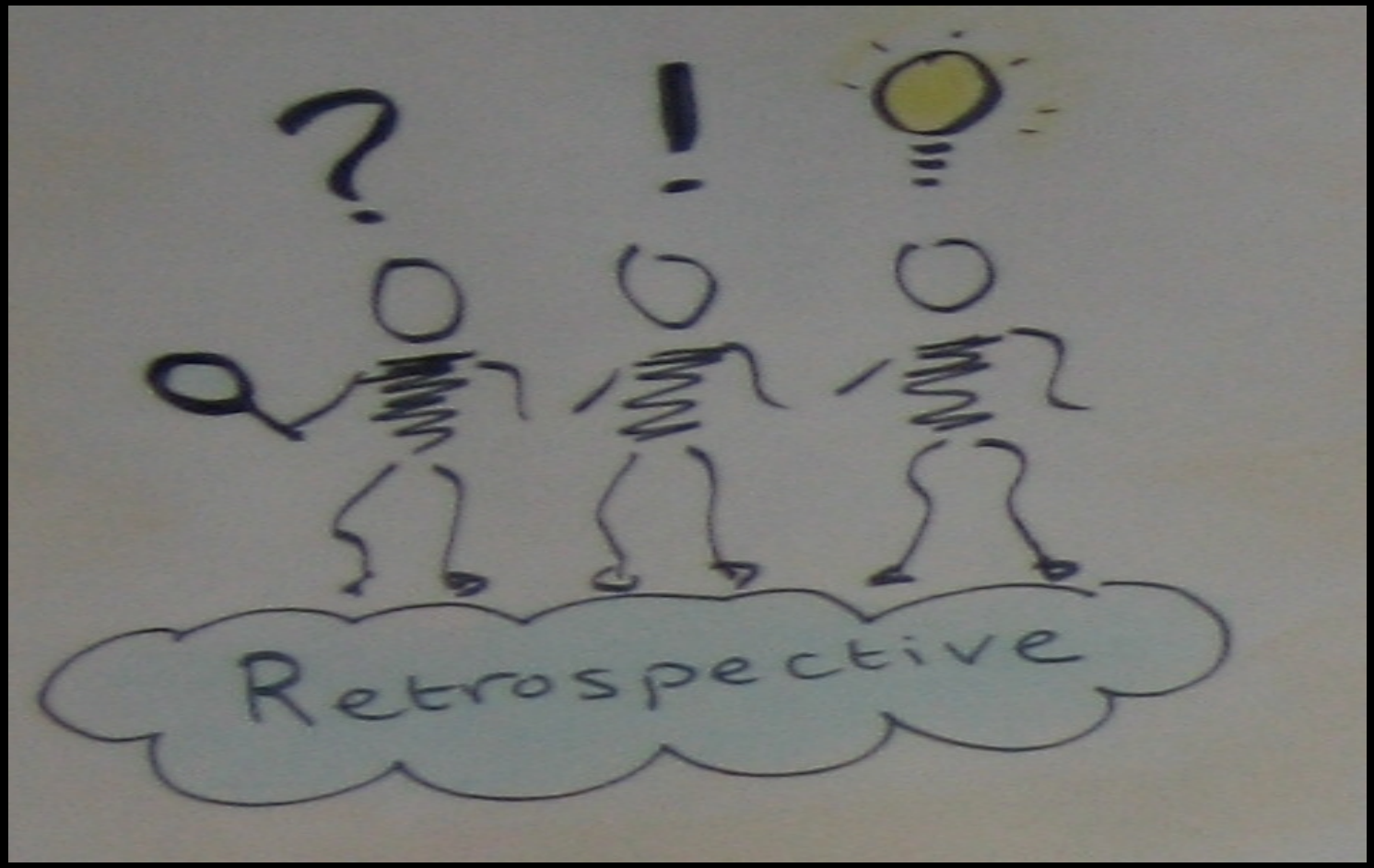
A meeting where a team looks back on a past period of work so that they can learn from their experience and apply this learning to future projects





Pioneered by Norm Kerth

Hmm? Really? So?



James Bach's 3 words to trigger critical thinking

THOSE WHO DON'T LEARN FROM THE MISTAKES IN THE PAST ARE DESTINED TO REPEAT THEM

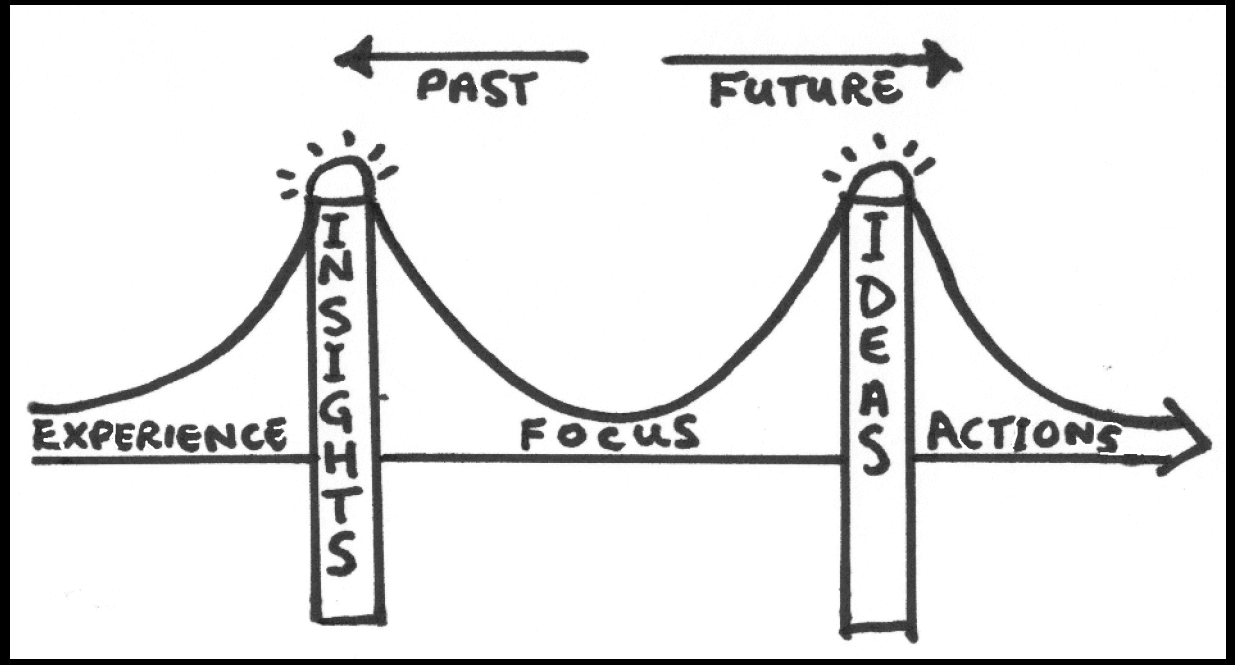
No time to improve?!

Groundhog Day

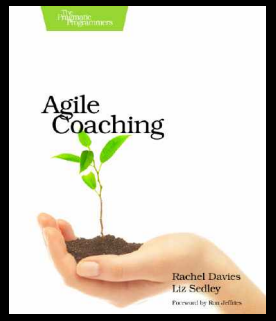
"Without retrospectives you will find that the team keeps making the same mistakes over and over again."

Henrik Kniberg

Structure Flow of Conversation



From “Agile Coaching” book



Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Recycle Bin 101

3 Products That Have Erroneously Achieved Immortality

Chris Oldwood

gort@cix.co.uk

Visual SourceSafe

- 10 Analyze.exe –F, 20 GOTO 10
- More chatty than Alan Carr
- Working Folder randomiser

Visual C++ 6

- Visual Studio 1998 (10 is the new 6)
- #if _MSC_VER < 1300
- It's *Visual* C++

Internet Explorer 6

- The Corporate Standard
- MDI is so 90’s
- jQuery is fanning the flames

Recycle Bin 101

Chris Oldwood

gort@cix.co.uk

ACCU 2010 Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Agility is the Tool, Not the Master:
Gilb's
Ten Key Agile Principles
to deliver stakeholder value,
avoid bureaucracy
and give creative freedom

Tom@Gilb.com
www.Gilb.com (slides will be here)
http://www.gilb.com/tiki-download_file.php?fileId=389
Unicom Keynote 27th April 2010 London
Paper: To appear agilerecord.com Summer 2010.
http://homepage.mac.com/tomgilb/filechute/Agile%20Principles%20and%20Values%20for%20Agile%20Record%202010%20Gilb.doc

Summary

- Introduction
- What is Stakeholder Value?
- How does stakeholder value relate to business benefits?
- How does IT System Quality relate to stakeholder
- values?
- What does Scrum do about this? why is Scrum inadequate?
- What new front end do we need for Scrum – or any Agile variant?
- 10 Principles for Agile Value Delivery

Introduction

What is Stakeholder Value?

- Critical Stakeholders
  - Can determine system success or failure
- Stakeholder value
  - any things that stakeholders want, need, value
    - Independently of 'your' system
    - Independently of the cost to someone of delivering those values
    - For example (stakeholder values)
      - Save time
      - Easier to learn
      - More secure
      - Easier to get things changed

How does stakeholder value relate to business benefits?

Stakeholder values

- will to some degree *directly drive* business benefits
- To some degree stakeholder values are necessary to satisfy in order to *avoid constraining delivery* of business benefits
- To some degree are *irrelevant* to business benefits

How does IT System Quality relate to stakeholder value?

If stakeholder value is ‘save my time’

then we can satisfy their needs in many ways

For example: (IT system design, requirements

- Higher availability of the system
- More usability
- Better performance (throughput, response)
- More integration with other subsystems
- Better automatic error detection and correction

What does Scrum do about this?

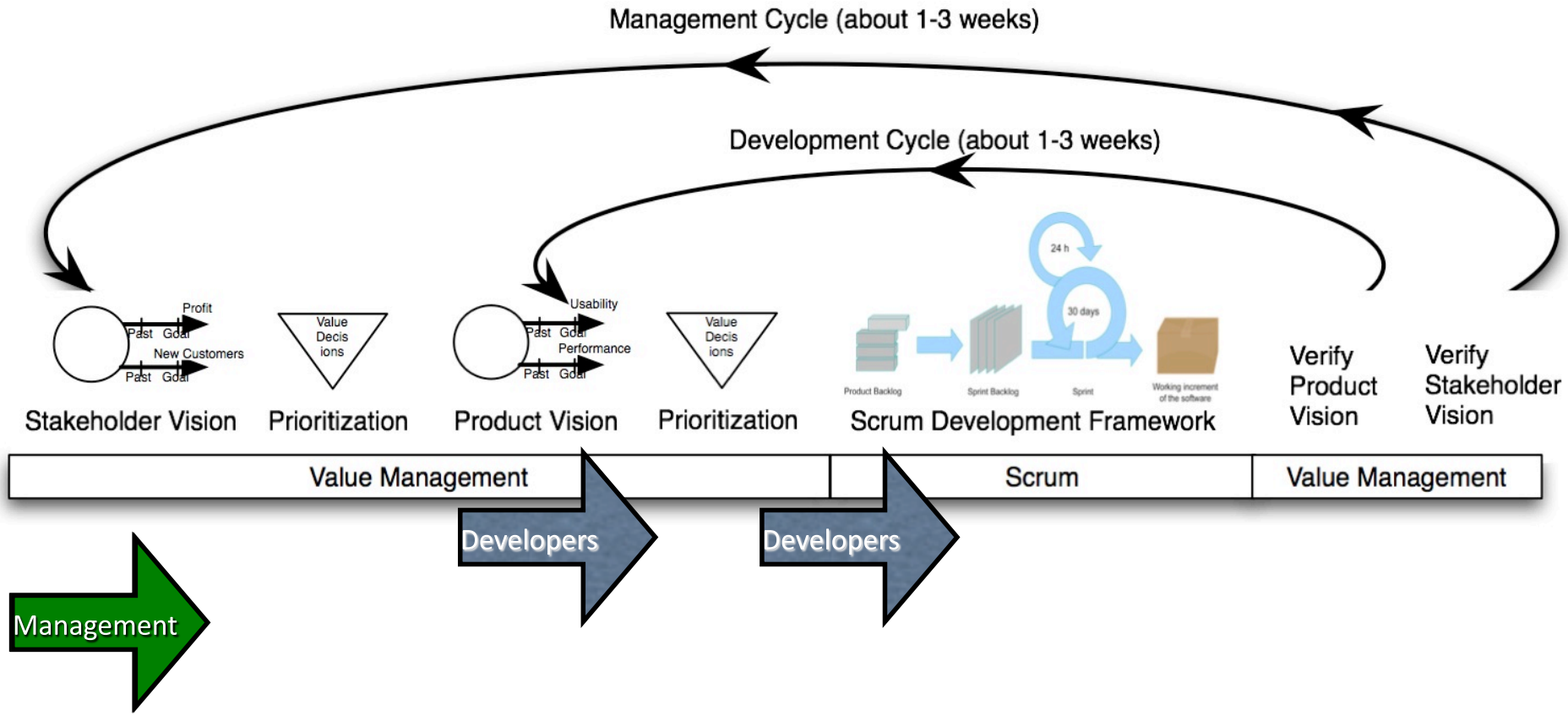
- Scrum is focussed on implementing designs, functions, features, use cases
  - As interpreted by a product owner
- It does not explicitly deal with business requirements, stakeholder values, product qualities.
- You need to add explicit 'front ends' to Scrum in order to deal with values
- You can think of this as an extension of what the Product Owner needs to be taught to do.
- http://www.gilb.com/tiki-download_file.php?fileId=353
  - Is a free set of slides designed to teach Product Owners how to do this
  - Developed with Gabrielle Benefield (Scum Alliance) Oct 2009

What new 'front end' do we need for Scrum – or any Agile variant?

The 'Business End' to Scrum (and other Agile variants) needs to:

- Explicitly, quantitatively, **define business objectives**, and constraints
  - Example: Increase Business Orders and sales
- Identify **stakeholder values**, quantitatively, that are directly related to the business values
  - Example: Help potential customers (Stakeholder) find what they want to more quickly
- Identify and define quantitatively the IT **system quality and performance requirements** needed to satisfy the prioritised stakeholder values
  - Example: Usability requirement: reduce time needed to correctly identify the correct transportation service to average under 50 seconds.
- Identify and define the **technical designs** needed to satisfy the product qualities and performance requirements
  - Design 1: Radical improved User Interface
  - Design 2: train website content providers to write clearer and more product-distinctive texts
- Tie these 4 levels together logically using 3 levels of Impact Estimation Tables, 4 level hierarchy

Value Management



http://www.gilb.com/tiki-download_file.php?fileId=277
slides May 09 Posten

Value Decision Tables

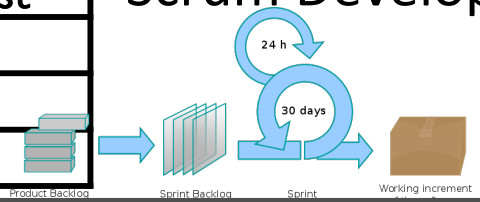
Business Goals	Stakeholder Value 1	Stakeholder Value 2
Business Value 1	-10%	40%
Business Value 2	50%	10%
Resources	20%	10%

Stakeholder Val.	Product Value 1	Product Value 2
Stakeholder Value 1	-10%	50 %
Stakeholder Value 2	10 %	10%
Resources	2 %	5 %

Product Values	Solution 1	Solution 2
Product Value 1	-10%	40%
Product Value 2	50%	80 %
Resources	1 %	2 %

Prioritized List
1. Solution 2
2. Solution 9
3. Solution 7

Scrum Develops



We measure improvements
Learn and Repeat

So, what are Agile methods missing?

- Stakeholder Focus
  - Real projects have dozens of stakeholders
    - Not just a customer in the next room
- Results Focus
  - It is not about writing code, it is about delivering value to stakeholders
  - It is not about programming, it is about making systems work for real people
- Systems Focus
  - It is not about coding - again
  - It is about reuse, data, hardware, training, motivation, sub-contracting, Outsourcing, help lines, user documentation, user interfaces, security
  - So, a systems engineering scope is necessary to deliver results.
  - Systems Engineering needs quantified performance and quality objectives,
    - to synchronize all necessary disciplines so that they deliver the results.

Gilb's Ten Key Agile Principles

to avoid bureaucracy and give creative freedom (summary)

1. **Control projects by quantified critical-few results. 1 Page total !**
   (not stories, functions, features, use cases, objects, ..)
2. **Make sure those results are business results, not technical**
3. **Align your project with your financial sponsor's interests!**
4. **Give developers freedom, to find out *how* to deliver those results**
5. **Estimate the impacts of your designs, on *your* quantified goals**
6. **Select designs with the best impacts for their costs, do them first.6. Decompose the workflow, into weekly (or 2% of budget) time boxes**
7. **Change designs, based on quantified experience of implementation**
8. **Change requirements, based in quantified experience, new inputs**
9. **Involve the stakeholders, every week, in setting quantified goals**
10. **Involve the stakeholders, every week, in *actually using* increments**

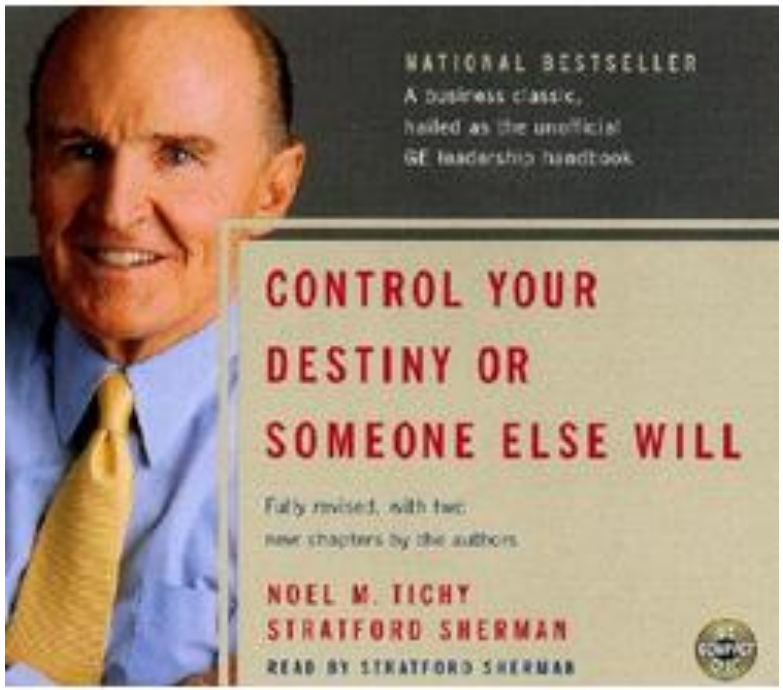
1. **Control projects by quantified critical-few results.**

1 Page total !

(not stories, functions, features, use cases, objects, ..)

Few *Clear* Top Goals

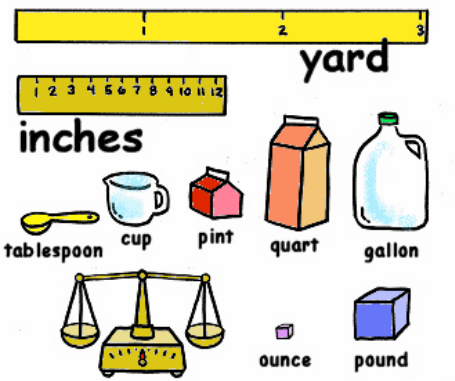
- ***Instead of directing business according to detailed...strategic plan,***
  - ***[Jack] Welch [General Electric CEO]***
- ***believed in setting only a few clear, overarching goals.***
- ***Then, on an ad hoc basis,***
  - ***his people were free to seize any opportunities***
  - ***they saw***
  - ***to further those goals.*** –



- Noel Tichy and Stratford Sherman, "Control Your Own Destiny or Someone Else Will"

Summary of Top '8' Project Objectives NOT 'clear'!

- **Defined** Scales of Measure:
  - Demands ***comparative* thinking**.
  - Leads to requirements that are unambiguously **clear**
  - Helps Team be **Aligned** with the Business



Real Example of ***Lack*** of Clarity

*1. Central to The Corporations business strategy is to be the world's **premier** integrated_<domain> service **provider**.*

*2. Will provide a much more efficient **user** experience*

*3. Dramatically scale back the **time** frequently needed after the last data is acquired to time align, depth correct, splice, merge, recompute and/or do whatever else is needed to **generate** the desired **products***

*4. Make the system much **easier** to **understand** and **use** than has been the case for previous system.*

*5. A primary goal is to provide a much more **productive** system **development** environment than was previously the case.*

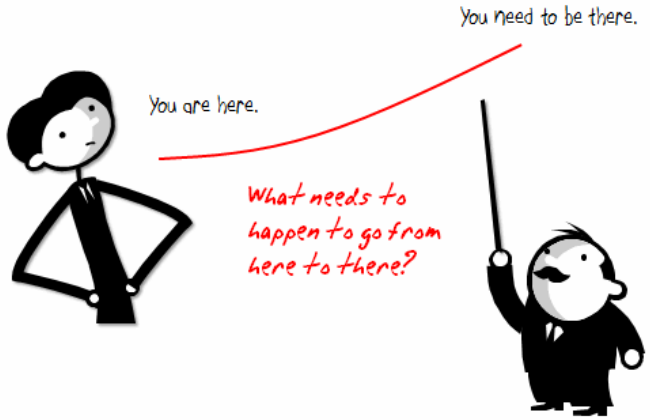
*6. Will provide a richer set of functionality for **supporting** next-generation logging **tools** and applications.*

*7. **Robustness** is an essential system requirement (see rewrite in example below)*

*8. Major improvements in **data quality** over current practices*

This lack of clarity cost them $100,000, 000 to $160 mill.

WHY are we doing this?
Part of Platform Rationalisation Initiative, with below **Main Objectives.**



- **Rationalize into a smaller number of core processing platforms. This cuts technology spend on duplicate platforms, and creates the opportunity for operational saves. Expected 60%-80% reduction in processing cost to Fixed Income Business levies.**
- **International Securities on one platform, Fixed Income and Equities (Institutional and PB).**
- **Global Processing consistency with single Operations In-Tray and associated workflow.**
- **Consistent financial processing on one Accounting engine, feeding a single sub-ledger across products.**
- **First step towards evolution of "Big Ideas" for Securities.**
- ***Improved development environment*, leading to increased capacity to enhance functionality in future.**
- **Removes duplicative spend on two back office platforms in support of mandatory message changes, etc.**

How can we improve such bad specification? ('Planguage')

Development Capacity:

Version: 3 Sept 2009 16:26

Type: Main <Complex/Elementary> Objective for a project.

Ambition Level: radically increase the capacity for developers to do defined tasks. <- Tsg

Scale: the Calendar Time for defined [Developers] to Successfully carry out defined [Tasks].

Owner: Tim Fxxx

C**alendar Time**: defined as: full working days within the start to delivery time frame.

Past [2009, {Bxx, Lxx, Gxx}, If QA Approved Processes used, Developer = Architect, Task = Draft Architecture] **15 days** ±4 ?? <- Rob

Goal[2011, { Bxx, Lxx, Gxx }, If QA Approved Processes used, Developer = Architect, Task = Draft Architecture] **1.5 days** ± 0.4 ?? <- Rob

Justification: Really good architects are very scarce so we need to optimize their use.

Risks: we use effort that should be directed to really high volume or even more critical areas (like Main Objective).

2. Make sure those results are business results, not technical,

What *level* are these objectives?

Business, User stakeholder, IT Technical?

- *1. Central to The Corporations business strategy is to be the world's **premier** integrated_<domain> service **provider**.*
- *2. Will provide a much more efficient **user** experience*
- *3. Dramatically scale back the **time** frequently needed after the last data is acquired to time align, depth correct, splice, merge, recompute and/or do whatever else is needed to **generate** the desired **products***
- *4. Make the system much **easier** to **understand** and **use** than has been the case for previous system.*
- *5. A primary goal is to provide a much more **productive** system **development** environment than was previously the case.*
- *6. Will provide a richer set of functionality for **supporting** next-generation logging **tools** and applications.*
- *7. **Robustness** is an essential system requirement (see rewrite in example below)*
- *8. Major improvements in **data quality** over current practices*

3. Align your project with your *financial* sponsor's interests!

- The Golden Rule:
  - He who has the gold, rules
- Find out and document exactly what the *project financial sponsors* are expecting for their budget for your project
  - They are a key stakeholder
  - They will expect you to satisfy several other stakeholders

Business Result Alignment: BRA:

- *Ambition:* ***Maximize delivery speed, and satisfaction level, of the Change the Bank Book of Work to achieve 'key business goals'***
- Scale: % of Planned Value actually Delivered to the Business by defined [Time].
- Past [Corp., Time = Deadline, 2007]: X% (guess
- X < 30%??) <- tg
- Goal [Corp., Time = Deadline, 2009]: < 50%, maybe much more?
- Issue: can The Tool be exploited to track Value?
-

Avoid Duplication:

- *Ambition:* ***eliminate corporate efforts that duplicate other corporate efforts.***
- Scale: % of project investment that is Duplicated
- Past [2007]: > 30%?? Wild guess
- Goal [2010] < 5% hope
-

Exploiting Existing Tools:

- *Ambition:* ***make use of existing tools, avoid reinventing the wheel.***
- Scale: % by Total Investment Value that Arguably could be avoided by Profitably making use of Existing Tools
- Past: 30%±30% ?? wild initial guess to start discussion tg
- Goal [2012?, Corp. Wide]: ~ 100%
-

Results MIS:

- *Ambition:* ***deliver high-significance real-time metrics, on critical aspects, of project results and resources.***
- Scale: % of defined [Key Project Data] available to management in real time.
- Key Project Data: default: {% of Goal Delivered to date, Stakeholder Satisfaction level, Value for Money}
- Past [Corp., 2007]: 0%
- Goal [Corp., 2010]: > 90%

4. Give developers freedom, to find out *how* to deliver those results

- Do not allow customers and salespeople to dictate to developers the technical solutions, such as screen layouts
- They are 'amateurs' at design, and will ruin the design for themselves and others
- They have no overview of the many requirements and constraints that a designer must consider simultaneously
- What you need to do is to establish the RESULTS valued by user stakeholders, and allow the developers (architects, user interface engineers, programmers as designers) to find and measure solutions that give the results desired
  - Such as: speed, correctness, input error detection and correction capability, ease of learning, leveraging other systems data.
- One of our clients (Confirmit/Firm, see Case gilb.com)calls this 'Empowered Creativity"

Quantified top level product objectives

4 product areas were attacked in all: 25 Qualities concurrently, one quarter of a year. Total development staff = 13

9

8

3

3

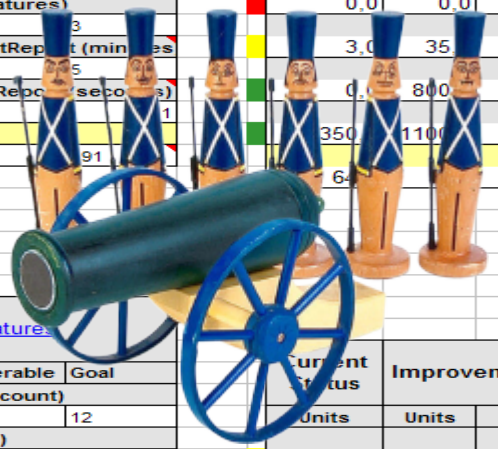
Impact Estimation Table: Reportal codename "Hyggen"

Current Status	Improvements		Reportal - E-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitivness (%)		
75,0	25,0	62,5	50	75	90
			Usability.Consistency.Visual (Elements)		
14,0	14,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	75,0	96,2	80	5	2
5,0	45,0	95,7	50	5	1
			Usability.Flexibility.OfflineReport.ExportFormats		
3,0	2,0	66,7	1	3	4
			Usability.Robustness (errors)		
1,0	22,0	95,7	7	1	0
			Usability.Replacability (nr of features)		
4,0	5,0	100,0	8	5	3
			Usability.ResponseTime.ExportRep t (min s)		
1,0	12,0	150,0	13	5	
			Usability.ResponseTime.ViewRepo seco s)		
1,0	14,0	100,0	15		1
			Development resources		
203,0			0		91

Current Status	Improvements		Survey Engine .NET		
Units	Units	%	Past	Tolerable	Goal
			Backwards.Compatibility (%)		
83,0	48,0	80,0	40	85	95
0,0	67,0	100,0	67	0	0
			Generate.WI.Time (small/medium/large seconds)		
4,0	59,0	100,0	63	8	4
10,0	397,0	100,0	407	100	10
94,0	2290,0	103,9	2384	500	180
			Testability (%)		
10,0	10,0	13,3	0	100	100
			Usability.Speed (seconds/user rating 1-10)		
774,0	507,0	51,7	1281	600	300
5,0	3,0	60,0	2	5	7
			Runtime.ResourceUsage.Memory		
0,0	0,0	0,0		?	?
			Runtime.ResourceUsage.CPU		
3,0	35	97,2	38	3	2
			Runtime.ResourceUsage.MemoryLeak		
0	800	100,0	800	0	0
			Runtime.Concurrency (number of users)		
350	1100	146,7	150	500	1000
			Development resources		
64			0		84

Current Status	Improvements		Reportal - MR Features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Replacability (feature count)		
1,0	1,0	50,0	14	13	12
			Usability.Productivity (minutes)		
20,0	45,0	112,5	65	35	25
			Usability.ClientAcceptance (features count)		
4,4	4,4	36,7	0	4	12
			Development resources		
101,0			0		86

Current Status	Improvements		XML Web Services		
Units	Units	%	Past	Tolerable	Goal
			TransferDefinition.Usability.Efficiency		
7,0	9,0	81,8	16	10	5
17,0	8,0	53,3	25	15	10
			TransferDefinition.Usability.Response		
943,0	-186,0	######	170	60	30
			TransferDefinition.Usability.Intuitiveness		
5,0	10,0	95,2	15	7,5	4,5
			Development resources		
0			0		48



http://www.gilb.com/tiki-download_file.php?fileId=32

Paper on case.

Confirmit Release 8.5

Trond Johansen

5 Estimate the impacts of your designs, on *your* quantified goals

- The only justification for a design is that it helps us reach our goals, as expressed by our requirement levels.
- It is critical that we have a fairly clear expectation of how powerful or useful a design will be for us.
- It is not an efficient practice to just select a promising design, and then try it out.
- You risk wasted energy. Better to face the bad news early – by estimating the power of the design, before you decide which design to try out.
- One problem is that the best solutions might also have bad side effects too tight security might destroy user friendliness.
- Another problem is that the costs of the design need to be estimated, and need to be compatible with overall resource budgets, and the resources needed for all the other designs!
- If you think the above is just good common sense, then recognize that in IT and software it is the exception. Designs are selected intuitively, and culturally – but there is no 'engineering' evaluation behind them. No wonder we fail so often!

Case of Estimating Impact of a Design on a Required Goal Level

Trond Johansen

- Impact Table for Market Research Web=product Project
- **Solution:** 'Recoding' (Market Research data recoding)
  - "Make it possible to recode variable on the fly from Reportal".
  - Estimated effort: 4 days
  - **Estimated'** Productivity' improvement: 20 minutes (50% way to Goal)
  - actual result 38 minutes (95% progress towards Goal)

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	66.7	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

confirmit

6. Select designs with the best impacts for their costs, do them first.

- **Designs should be chosen**
  - Based on their contribution to our requirements *Goal levels*.
  - And on their contribution to the *entire set* of critical objectives (top ten)
  - And on their total value for *total costs*
    - Both short-term and long-term costs, and resources

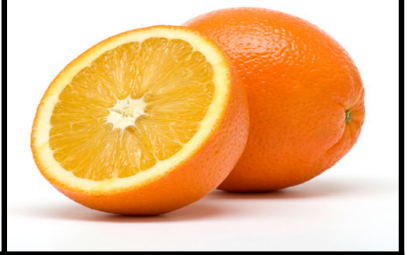
Value Decision Tables

Product Values	Solution 1	Solution 2
Product Value 1		
Product Value 2		
Resources		

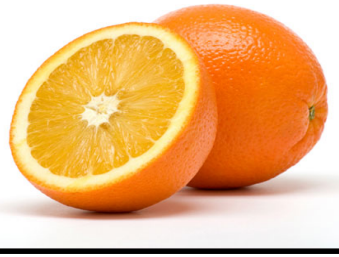
Value Decision Tables

Product Values		
Product Value 1		
Product Value 2		
Resources		

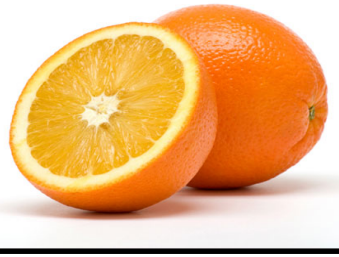
Value Decision Tables

Product Values		
Product Value 1		
Product Value 2		
Resources		

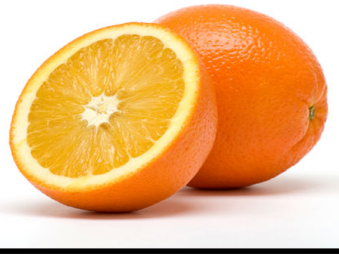
Value Decision Tables

Product Values			
Product Value 1			
Product Value 2			
Resources			

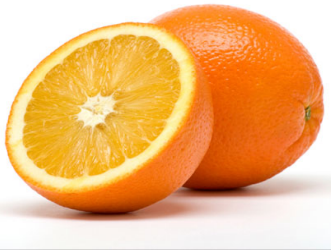
Value Decision Tables

Product Values			
Taste			
Resources			

Value Decision Tables

Product Values			
Taste			
Nutrition			
Resources			

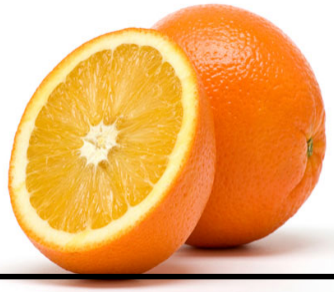
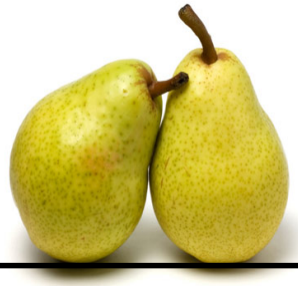
Value Decision Tables

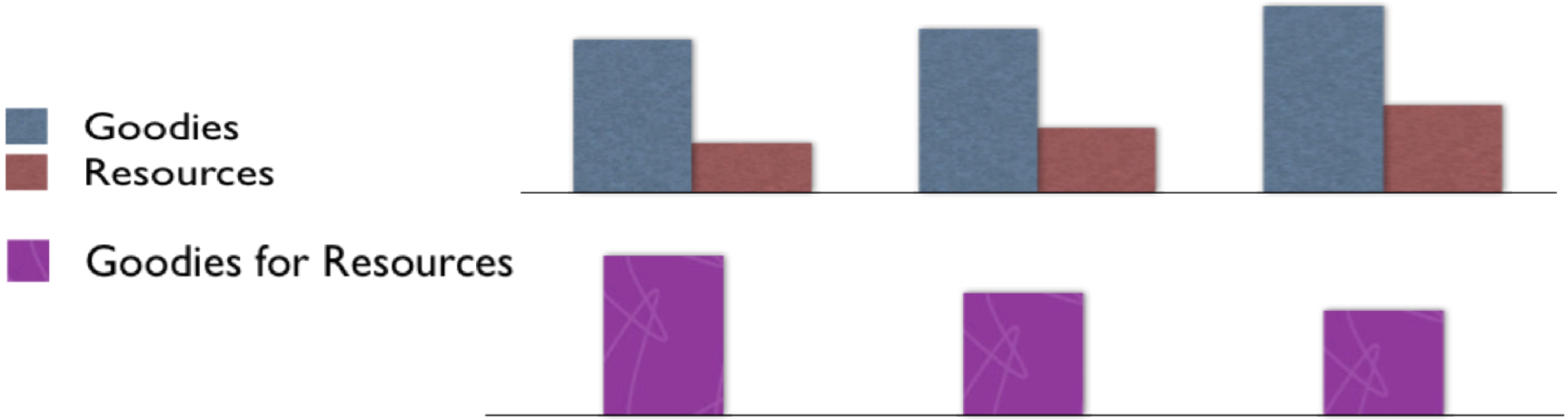
Product Values			
Taste			
Nutrition			
Shelf Life			
Resources			

Value Decision Tables

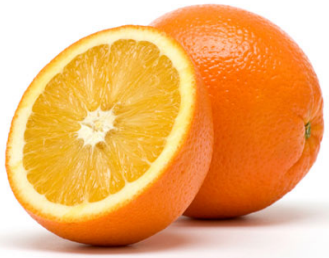
Product Values			
Taste			
Nutrition			
Shelf Life			
Sum Goodies			
Resources			

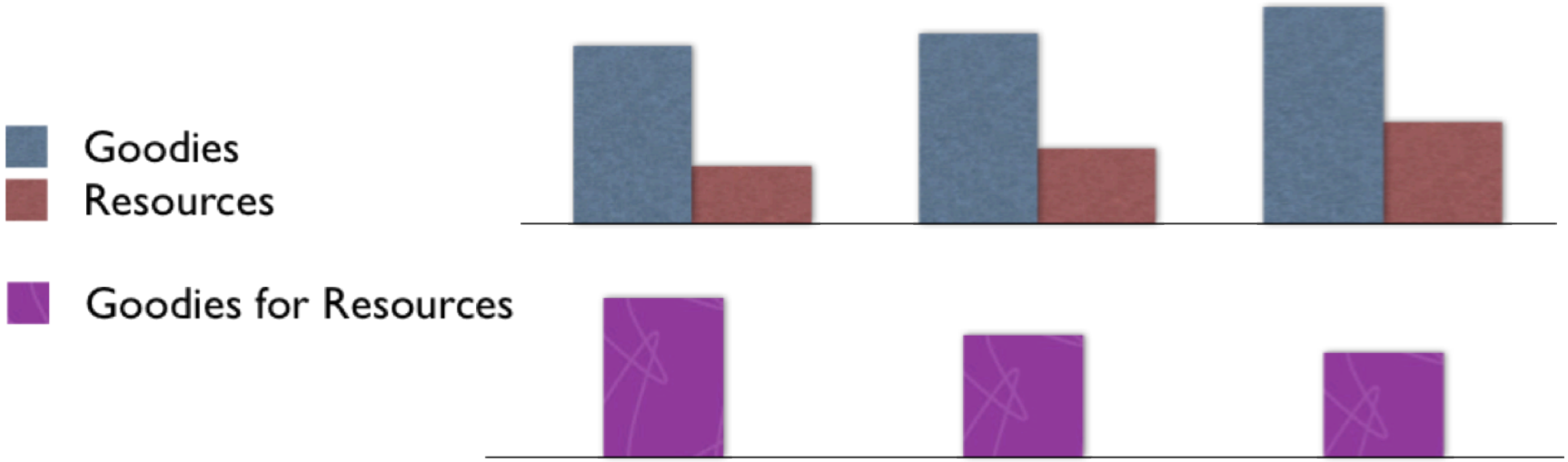
Value Decision Tables

			
Product Values	20 %	50 %	90 %
Taste	30 %	70 %	90 %
Nutrition	80 %	30 %	-10 %
Shelf Life	130 %	150 %	170 %
Sum Goodies	40 %	60 %	80 %
Resources			

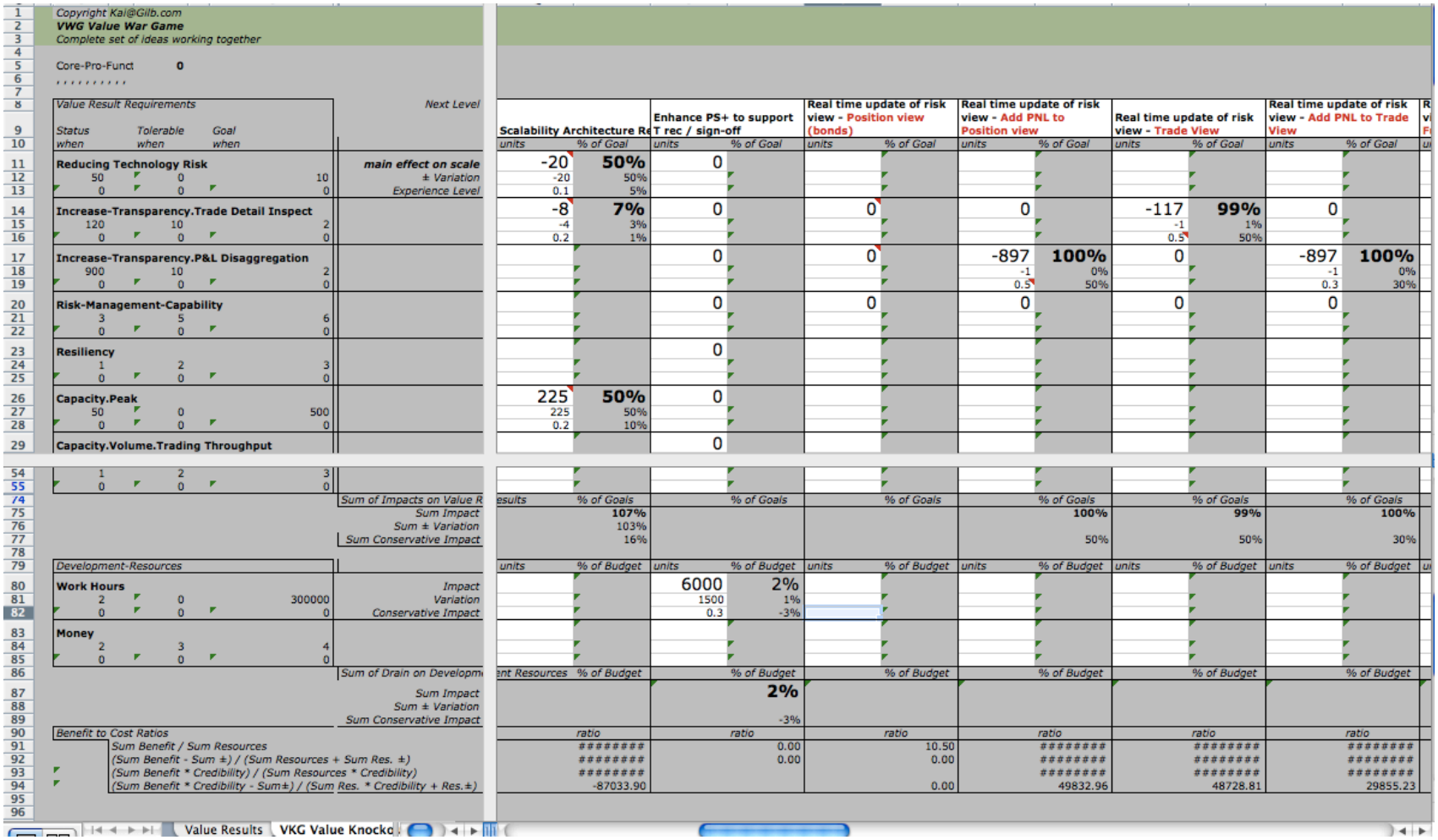


Value Decision Tables

Product Values			
Taste	20 %	50 %	90 %
Nutrition	30 %	70 %	90 %
Shelf Life	80 %	30 %	-10 %
Sum Goodies	130 %	150 %	170 %
Resources	40 %	60 %	80 %



Real Case of
Impacts for different designs



Copyright Kai@Gilb.com
VWG Value War Game
Complete set of ideas working together

Core-Pro-Funct 0

Value Result Requirements	Status when	Tolerable when	Goal when	Next Level
Reducing Technology Risk	50	0	10	*main effect on scale*
	0	0	0	± Variation
				Experience Level
Increase-Transparency.Trade Detail Inspect	120	10	2	
	0	0	0	
Increase-Transparency.P&L Disaggregation	900	10	2	
	0	0	0	
Risk-Management-Capability	3	5	6	
	0	0	0	
Resiliency	1	2	3	
	0	0	0	
Capacity.Peak	50	0	500	
	0	0	0	
Capacity.Volume.Trading Throughput	1	2	3	
	0	0	0	

	Scalability Architecture Re		Enhance PS+ to support T rec / sign-off		Real time update of risk view - Position view (bonds)		Real time update of risk view - Add PNL to Position view		Real time update of risk view - Trade View		Real time update of risk view - Add PNL to Trade View	
	units	% of Goal	units	% of Goal	units	% of Goal	units	% of Goal	units	% of Goal	units	% of Goal
	-20	**50%**	0									
	-20	50%										
	0.1	5%										
	-8	**7%**	0		0		0		-117	**99%**	0	
	-4	3%							-1	1%		
	0.2	1%							0.5	50%		
			0		0		-897	**100%**	0		-897	**100%**
							-1	0%			-1	0%
							0.5	50%			0.3	30%
			0		0		0		0		0	
			0									
	225	**50%**	0									
	225	50%										
	0.2	10%										
			0									

		Scalability Architecture Re		Enhance PS+ to support T rec / sign-off		Real time update of risk view - Position view (bonds)		Real time update of risk view - Add PNL to Position view		Real time update of risk view - Trade View		Real time update of risk view - Add PNL to Trade View	
		esults	% of Goals		% of Goals		% of Goals		% of Goals		% of Goals		% of Goals
Sum of Impacts on Value R	Sum Impact		**107%**						**100%**		**99%**		**100%**
	Sum ± Variation		103%										
	Sum Conservative Impact		16%						50%		50%		30%
Development-Resources		units	% of Budget	units	% of Budget	units	% of Budget	units	% of Budget	units	% of Budget	units	% of Budget
Work Hours	Impact			6000	2%								
	Variation			1500	1%								
	Conservative Impact			0.3	-3%								
Money													
Sum of Drain on Developm		ent Resources	% of Budget		% of Budget		% of Budget		% of Budget		% of Budget		% of Budget
	Sum Impact				**2%**								
	Sum ± Variation												
	Sum Conservative Impact				-3%								
Benefit to Cost Ratios			ratio		ratio		ratio		ratio		ratio		ratio
	Sum Benefit / Sum Resources		########		0.00		10.50		########		########		########
	(Sum Benefit - Sum ±) / (Sum Resources + Sum Res. ±)		########		0.00		0.00		########		########		########
	(Sum Benefit * Credibility) / (Sum Resources * Credibility)		########						########		########		########
	(Sum Benefit * Credibility - Sum±) / (Sum Res. * Credibility + Res.±)		-87033.90				0.00		49832.96		48728.81		29855.23

Value Results | VKG Value Knockd

7. Decompose the workflow, into weekly (or 2% of budget) time boxes

- Objectives of Decomposition
  - Early delivery of some value
  - Build credibility with stakeholders
  - Test out your development process
  - Reduce risk of loss to '2%'
  - Create value for money (ROI control)
- How to decompose?
  - By value
  - 1.1.1.1.1.1 method
    - 1 stakeholder, 1 value, 1% progress, 1 strategy, 1 week, 1 function
  - Use common sense and domain, technical knowledge

7. Decompose the workflow, into weekly time boxes
A Real Example of A Planned Step in Planguage (2010)

Reconcile Project X P&L Formulas.Vanilla Case 1:
Type: Evo Value Delivery Step
[Potentially Reusable for various positions and books]
Stakeholders: Dinesh, Developers who use it (Dan X), Neil?,
Step Owner: Michal X
Step Manager: Tom?
Status: first rough draft to see if we can define an Evo step at all
Approval: NOT YET
Approval Instance: Will
Version: April 9 2010 14:53
Summary: Identify and Reconcile Project X P&L Formulas for a Single Vanilla Position.
Detailed Step:
For
1 Position in 1 Simple (Vanilla Govt Bonds) Instrument,
1 Book,
1 Region
2 days in a row
intra-month, with no deals in progress
Do
1. Identify and consolidate info about P&L formulas in Project X
2. Put those formulas in Excel
3. Reconcile Excel with live Project X books
Part of Strategy Called:
P&L Documentation
Impacts Objectives called:
Primary: Increase The Transparency
% (to Goal) 1-3% ?? (very rough guess, not strictly on the I T T scale MG)
Issue: are we in fact missing some objectives? (MG thinks we are).
Secondary:
Negative:
Indirect Impacts Above:
P&L Consistency
Deliverables:
A spreadsheet (that increases our transparency)
Estimated Time:
1 week±? ?

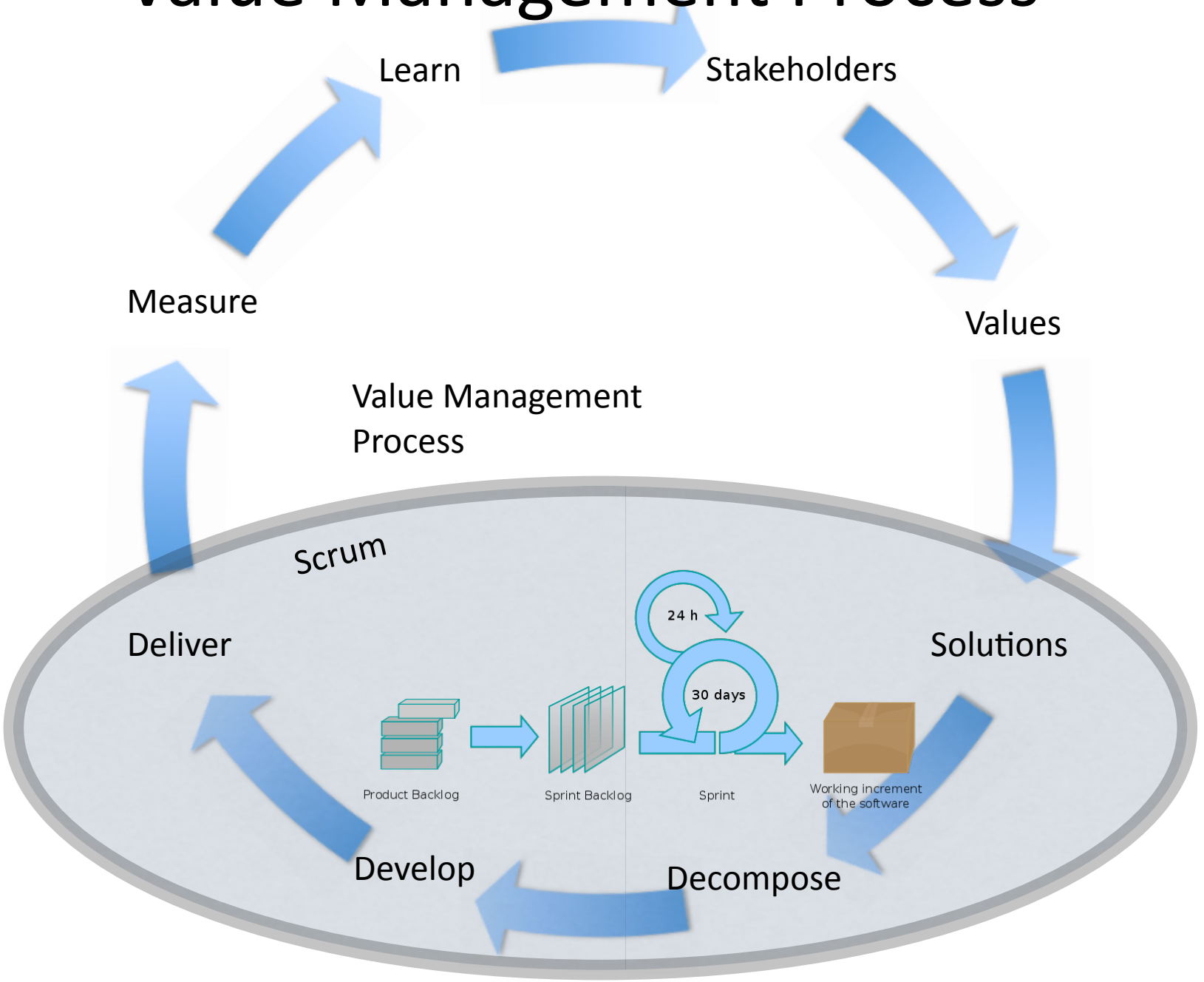
Necessary Resources:
Development
Time from Michal or equivalent
Neil X on holiday until Wednesday 14 April)
Ben?
From Krishna's group
Dinesh?
UAT 1 Environment
Assumption: generally available every day
Assumption: we do not need the downstream systems, at this stage
'I'd like it to be a copy of production" <-MG
I do not need a live system. <- MG
Dependencies:
D1:
Assumptions:
A1: we are just doing front to front.
Issues:
I1: does anyone know if the Project X methodology is the same across regions <- Eric (nobody was sure)
Resolution: Chris and …
I2: will any stakeholder really care? <- Eric
Risks:
R1: you might have to repeat this 3 times in order to get real value delivered <- Atul X
I might have to do it for every region, I am not sure there is any way to avoid it.
Future Step Variations: "not this week'!
Trades not yet settled
Forward starting trades
New issues
Fails
Cross end of month
Weekends, holidays
Late Trades/As Of

8. Change designs, based on quantified experience of implementation

Nobody can accurately predict the multiple effects of a design, which is added to a mix of other designs in a real world setting.

- It is too complicated, and we have too little knowledge to do so.
- In fact, like cooking, it is easier to taste the effect incrementally, to be sure.
- So, we are going to get some surprises
- And our only recourse is to learn quickly, and adjust quickly.
  - We don't want too much
  - And we don't want too little of the effects
  - Just right is fine.
- The reward for learning and for adjusting quickly is that we will reach more of our critical objectives, for less resource – or within our budgets and deadlines

Value Management Process



Quantified top level product objectives

NOTICE IN THE 9TH WEEK OF 12 THE % IMPROVEMENT IS FAR MORE THAN 75%
THIS IS ACHIEVED BY RAPID DYNAMIC FEEDBACK LEARNING AND REDEPLOYMENT

9

Impact Estimation Table: Reportal codename "Hyggen"

Current Status	Improvements		Reportal - E-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitivness (%)		
75,0	25,0	62,5	9	75	90
			Usability.Consistency.Visual (Elements)		
14,0	14,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	75,0	96,2	6	5	2
5,0	45,0	95,7	5	5	1
			Usability.Flexibility.OfflineReport.ExportFormats		
3,0	2,0	66,7	1	3	4
			Usability.Robustness (errors)		
1,0	22,0	95,7	7	1	0
			Usability.Replacability (nr of features)		
4,0	5,0	100,0	8	5	3
			Usability.ResponseTime.ExportReport (minutes		
1,0	12,0	150,0	1	13	5
			Usability.ResponseTime.ViewReport (seconds)		
1,0	14,0	100,0	15	3	1
			Development resources		
203,0			0		191

Current Status	Improvements		Survey Engine .NET		
Units	Units	%	Past	Tolerable	Goal
			Backwards.Compatibility (%)		
83,0	48,0	80,0	40	85	95
0,0	67,0	100,0	67	0	0
			Generate.WI.Time (small/medium/large seconds)		
4,0	59,0	100,0	63	8	4
10,0	397,0	100,0	407	100	10
94,0	2290,0	103,9	2384	500	180
			Testability (%)		
10,0	10,0	13,3	0	100	100
			Usability.Speed (seconds/user rating 1-10)		
774,0	507,0	51,7	1281	600	300
5,0	3,0	60,0	2	5	7
			Runtime.ResourceUsage.Memory		
0,0	0,0	0,0		?	?
			Runtime.ResourceUsage.CPU		
3,0	35,0	97,2	38	3	2
			Runtime.ResourceUsage.MemoryLeak		
0,0	800,0	100,0	800	0	0
			Runtime.Concurrency (number of users)		
1350,0	1100,0	146,7	150	500	1000
			Development resources		
64,0			0		84

8

3

Current Status	Improvements		Reportal - MR Features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Replacability (feature count)		
1,0	1,0	50,0	14	13	12
			Usability.Productivity (minutes)		
20,0	45,0	112,5	65	35	25
			Usability.ClientAcceptance (features count)		
4,4	4,4	36,7	0	4	12
			Development resources		
101,0			0		86

Current Status	Improvements		XML Web Services		
Units	Units	%	Past	Tolerable	Goal
			TransferDefinition.Usability.Efficiency		
7,0	9,0	81,8	16	10	5
17,0	8,0	53,3	25	15	10
			TransferDefinition.Usability.Response		
943,0	-186,0	######	170	60	30
			TransferDefinition.Usability.Intuitiveness		
5,0	10,0	95,2	15		4,5
			Development re		
2,0			0		48

3

http://www.gilb.com/tiki-download_file.php?fileId=32

Paper on case.

Confirmit Release 8.5

Trond Johansen

9. Involve the stakeholders, every week, in setting quantified goals

- When stakeholders experience that you really can deliver what they want
  - Then they will be more willing to spend time with you determining their real and immediate values.
  - Their values may have been changed by external events, since you last determined what they want
  - Resetting requirement levels, is bringing the requirements in line with current reality
    - Not locked into past misconceptions

10 . Involve the stakeholders, every week, in *actually using* increments

- ‘delivering working code to customers’ is not smart enough
- You need to deliver value increments to real stakeholder, like a clear time saving
- You need to spread from trial stakeholders towards all of them
- You need to measure reasonably well,
  - But not perfectly
  - Sometimes ‘early indicators’ (like speed for trial users) are more useful than the unrealistic dream of the final ‘lagging indicators’ (like time saved and staff reduction)
- You need to plan to capture other feedback in addition to the primary measures of value delivery

ACTUAL RESULTS IN *SECOND* 12 WEEKS OF USING 'Evo'

Evo's impact on Confirmit 9.0 product qualities

Product quality	Description	Customer value
Intuitiveness	Probability that an inexperienced user can intuitively figure out how to set up a defined Simple Survey correctly.	Probability increased by 175%
Productivity	Time in minutes for a defined advanced user, with full knowledge of 9.0 functionality, to set up a defined advanced survey correctly.	Time reduced by **38%**

Product quality	Description	Customer value
Productivity	Time (in minutes) to test a defined survey and identify 4 inserted script errors, starting from when the questionnaire is finished to the time testing is complete and is ready for production. (Defined Survey: Complex survey, 60 questions, comprehensive JScripting.)	Time reduced by **83%** and error tracking increased by 25%

MORE ACTUAL RESULTS IN SECOND 12 WEEKS OF USING 'Evo'

Evo's impact on Confirmit 9.0 product qualities

Product quality	Description	Customer value
Performance	**Max number of panelists that the system can support without exceeding a defined time for the defined task, with all components of the panel system performing acceptable.**	**Number of panelists increased by 1500%**
Scalability	**Ability to accomplish a bulk-update of X panelists within a timeframe of Z sec.**	**Number of panelists increased by 700%**
Performance	**Number of responses a database can contain if the generation of a defined table should be run in 5 seconds.**	**Number of responses increased by 1400%**

My 10 Agile Values?

- **Simplicity**
  - **1. Focus on real stakeholder values**
- **Communication**
  - **2. Communicate stakeholder values quantitatively**
  - **3. Estimate expected results and costs for weekly steps**
- **Feedback**
  - **4. Generate results, weekly, for stakeholders, in their environment**
  - **5. Measure all critical aspects of the improved results cycle.**
  - **6. Analyze deviation from your initial estimates**
- **Courage**
  - **7. Change plans to reflect weekly learning**
  - **8. Immediately implement valued stakeholder needs, next week**
    - ***Don't wait, don't study (analysis paralysis), don't make excuses.***
    - ***Just Do It!***
  - **9. Tell stakeholders exactly what you will deliver next week**
  - **10. Use any design, strategy, method, process that works quantitatively well - to get your results**
    - **Be a systems engineer, not a just programmer (a 'Softcrafter').**
    - **Do not be limited by your craft background, in serving your paymasters**

My 10 Agile Values?

- **Slides note: I am not going to detail these points yet as I suspect I wil do them in a single keynote.**
- **I have also not detailed the corresponding points in the Paper**
  - **Written for agilerecord.com**

Simplicity

1. Focus on real stakeholder values

Communication

2. Communicate stakeholder values quantitatively

3. Estimate expected results and costs for weekly steps

Feedback

4. Generate results, weekly, for stakeholders, in their environment

5. Measure all critical aspects of the improved results cycle.

6. Analyze deviation from your initial estimates

Courage

7. Change plans to reflect weekly learning

8. Immediately implement valued stakeholder needs, next week

- ***Don't wait, don't study (analysis paralysis), don't make excuses.***
- ***Just Do It!***

9. Tell stakeholders exactly what you will deliver next week

10. Use any design, strategy, method, process that works quantitatively well - to get your results

- **Be a systems engineer, not a just programmer (a 'Softcrafter').**
- **Do not be limited by your craft background, in serving your paymasters**

Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Are you getting enough?

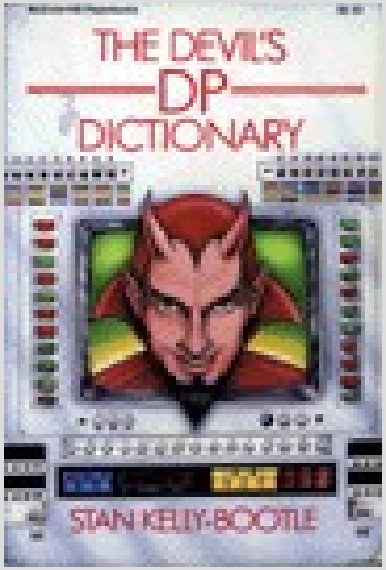
ACCU Oxford

accu-oxford@accu.org

http://lists.accu.org/mailman/listinfo/accu-oxford

http://www.oxfordfolkfestival.com/

http://www.oxfordfolkfestival.com/



Stan Kelly-Bootle

The Devil's DP Dictionary

Ambrose Bierce
The Devil's Dictionary

Man (n)**:**

An animal so lost in rapturous contemplation of what he thinks he is as to overlook what he indubitably ought to be. His chief occupation is extermination of other animals and his own species, which, however, multiplies with such insistent rapidity as to infest the whole habitable earth and Canada.

Major premise:

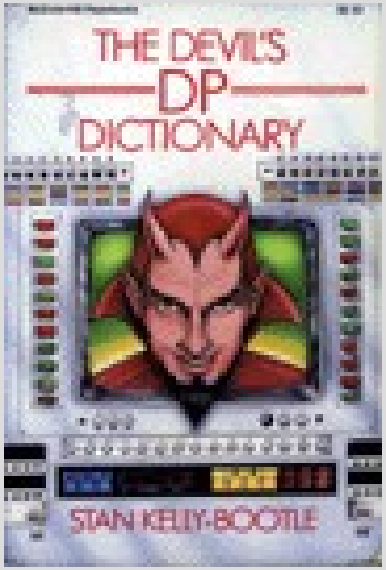
Sixty men can do sixty times as much work as one man.

Minor premise:

A man can dig a posthole in sixty seconds.

Conclusion:

Sixty men can dig a posthole in one second.



Stan Kelly-Bootle

The Devil's DP Dictionary

Infinite loop (n):

See: *Loop, infinite*

Loop, infinite (n):

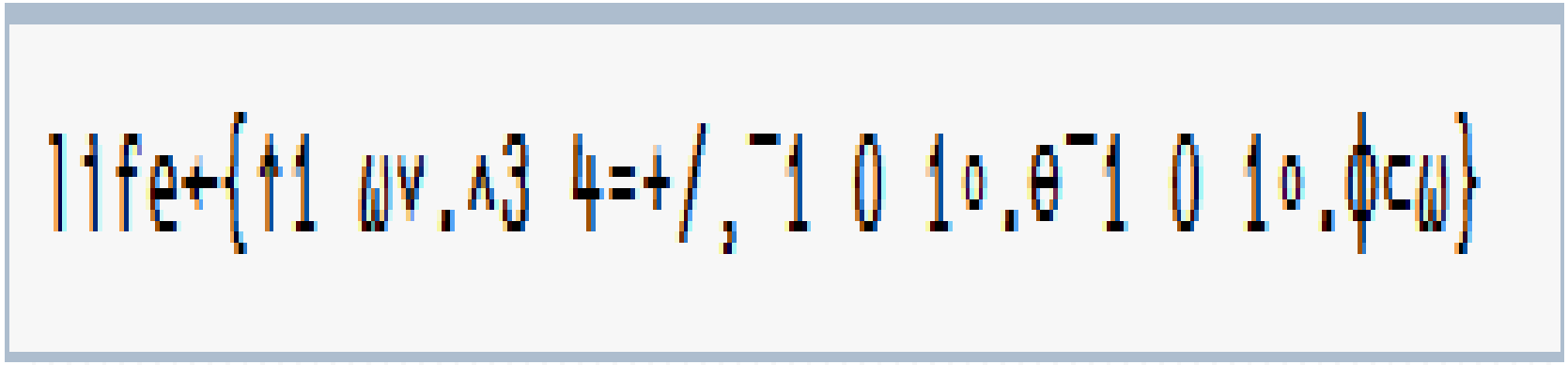
See: *Infinite loop*

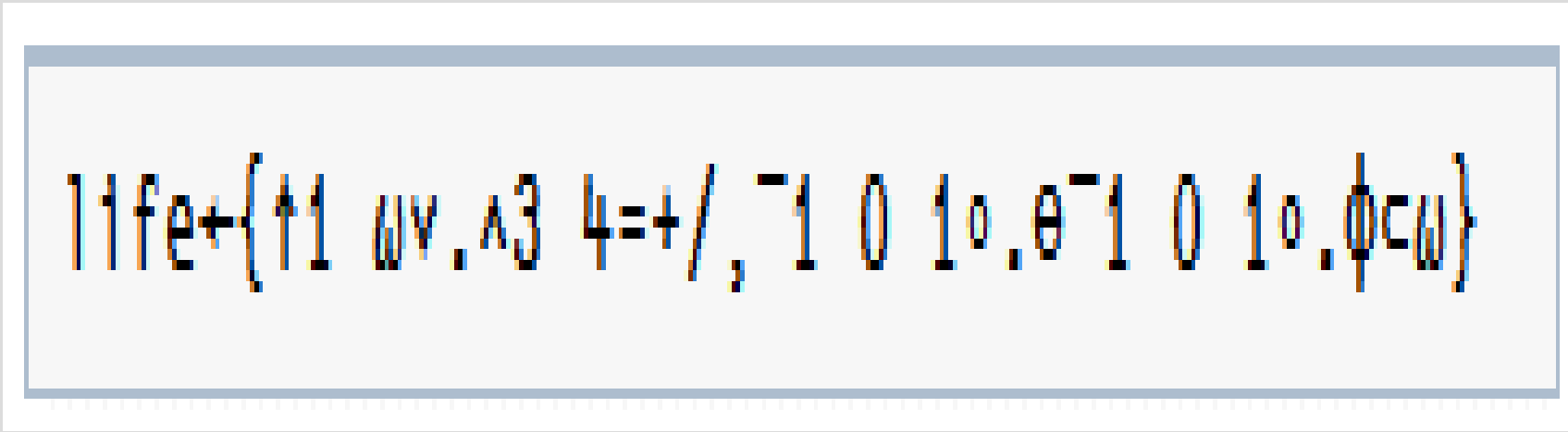
Recursion (n):

See: *Recursion*

Computer Science (n)*:*

A study akin to numerology and astrology, but lacking the precision of the former and the success of the latter.





There are three things a man must do
before his life is done;
Write two lines in APL,
And make the buggers run.

My program (n):

A gem of algorithmic precision, offering the most sublime balance between compact, efficient coding on the one hand, and fully commented legibility for posterity on the other. Compare *Your program*.

Your program (n):

A maze of non-sequiturs littered with clever-clever tricks and irrelevant comments. Compare *My program*.

Flowchart (v):

To obfuscate a problem with esoteric cartoons.

Implementation (n):

The fruitless struggle by the talented and underpaid to fulfill promises made by the rich and ignorant.

What makes programmers happy?

Algorasm (n):

[Origin: blend of *algorithm* + *orgasm*]

A sudden, short-lived moment of pleasure enjoyed by the programmer (and, for all we know, by the system) when the final *Kludge* rings the bell.

Have you had your 5-a-day?

Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Law of the Big Three (revised)

Dietmar Kühl
dietmar.kuehl@gmail.com
Bloomberg L.P. (London)

Law of the Big Three

from Marshall Cline's C++ FAQ

if one of the structors is need, so are the others:

- copy constructor
- copy assignment
- destructor

Motivation

```
struct big3
{
    big3(int v): ptr_(new int(v)) {}
    big3(big3 const& o):
      ptr_(new int(o.ptr_->v_)) {}
    ~big3() { delete this->v_; }
    big3& operator= (big3 const& o) {
      *this->v_ = *o.v_; return *this; }
    int* v_;
};
```

Example Type

```
struct example
{
    std::vector<int> array1_;
    std::vector<int> array2_;
};
```

- copy constructor and destructor are OK
- copy assignment is not!

Idiomatic Assignment

```
T& T::operator= (T const& other)
{
    T(other).swap(*this);
    return *this;
}
```

- of course, requires swap() being implemented
- ... which is wanted anyway

Exception Specifications

- BAD idea in ALL programming languages
- deprecated in C++0x

Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Lightning Talks

Jason McGuiness – Anti-Parallelism

Paul Black – Malware Research Protocol

James Bach – The “Also” Heuristic of Test Oracles

Rachel Davies – Learn From Experience With Retrospectives

Chris Oldwood – Recycle Bin 101

Tom Gilb – A Real Revolutionary Agile Manifesto

Jim Hague – Are You Getting Enough

Dietmar Kűhl – Law of The Big Three (Revised)

Phil Nash – What Will I Be Missing Out On If I Don't Develop For The iPhone or iPad

Boundary Conditions
“The Birthday Problem”

John Lakos

Friday, April 16, 2010

Copyright Notice

Abstract

In our component-based development methodology, virtually all of the software we design is rendered as components. When we say *component* in C++ we are referring to a `.h/.cpp` pair (of files) satisfying certain well-established, objective physical properties. Moreover, each of our developers is responsible for ensuring the correctness of the software he or she creates. Hence, along with each component developed, we require a standalone test driver to verify that all essential behavior implemented within that component behaves as advertised, i.e., according to the contract delineated in its component, class, and function-level documentation. In this very practicable talk, we will review the various categories of common classes (e.g., utilities, mechanisms, and value-semantic types). We will also review the basic principles of testing, various methods for systematically selecting test data, and the (four) primary implementation techniques for writing test cases. We will then discuss the organization of our self-contained (and delightfully light-weight) component-level test drivers. The substantial remainder of the talk will address (1) the details of how effective individual test cases are conceived, documented, and implemented, (2) how these test cases can be profitably ordered to leverage already proven component functionality, and (3) how similarities within the various class categories naturally lead to effective reusable testing patterns.

Test Data Selection Methods

How do we choose our test inputs ("test vectors")?

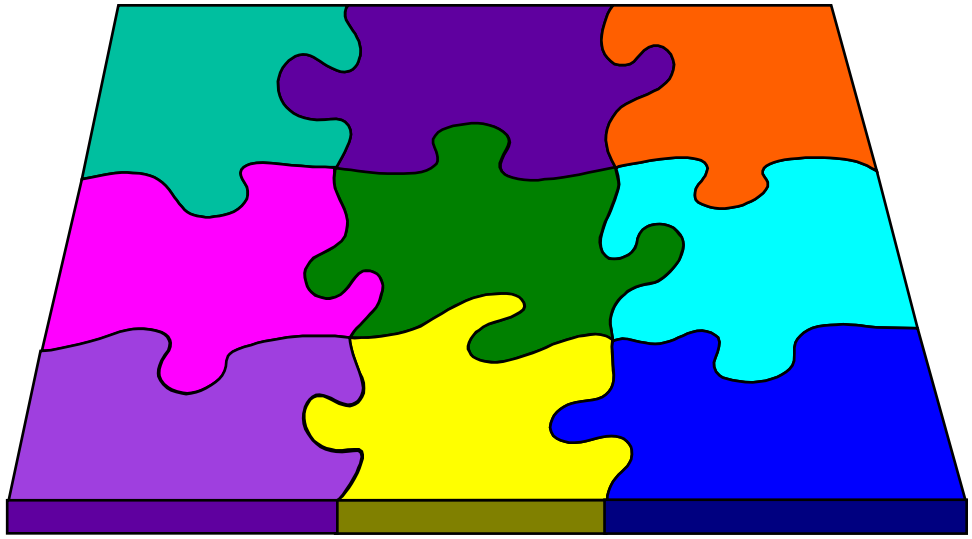
- **Ad hoc** – whatever makes sense
- **Boundary conditions** – look at the edges of our algorithms
- **Area testing** – try everything in an area or region
- **Orthogonal dimensions** – choosing a canonical value and then varying each dimension independently
- **Random/statistical** – relying on chance to detect unanticipated problems
- **Depth-Ordered Enumeration** – systematic testing around the origin of a design space
- **Category partitioning** – systematic testing based on equivalence classes

Test Data Selection Methods

How do we choose our test inputs ("test vectors")?

- **Ad hoc** – whatever makes sense
- **Boundary conditions** – look at the edges of our algorithms
- **Area testing** – try everything in an area or region
- **Orthogonal dimensions** – choosing a canonical value and then varying each dimension independently
- **Random/statistical** – relying on chance to detect unanticipated problems
- **Depth-Ordered Enumeration** – systematic testing around the origin of a design space
- **Category partitioning** – systematic testing based on equivalence classes

Data Selection: Boundary Conditions



Boundary Conditions

There are at least *three* kinds of boundaries to consider:

Those…

Boundary Conditions

There are at least *three* kinds of boundaries to consider:

Those…

- *Defined* by the **Interface**

Boundary Conditions

There are at least *three* kinds of boundaries to consider:

Those…

- *Defined* by the **Interface**
- *Created* by the **Implementation**

Boundary Conditions

There are at least *three* kinds of boundaries to consider:

Those…

- *Defined* by the **Interface**
- *Created* by the **Implementation**
- *Imposed* by the **Platform**

Boundary Conditions

Birthday Problem

Boundary Conditions

Birthday Problem

- What is the *minimum number* of people we would need in a room for the probability that "*at least two* of them have the same birthday" is *greater than 50%?*

Boundary Conditions

Birthday Problem

- What is the *minimum number* of people we would need in a room for the probability that "*at least two* of them have the same birthday" is *greater than 50%?*
- Simplifying Assumptions:
  - Only the day of the year matters.
  - All years have 365 days (no leap years)
  - Birthdays are uniformly distributed over the days of the year.

Boundary Conditions

Birthday Problem

- What is the *minimum number* of people we would need in a room for the probability that "*at least two* of them have the same birthday" is *greater than 50%?*
- Simplifying Assumptions:
  - Only the day of the year matters.
  - All years have 365 days (no leap years)
  - Birthdays are uniformly distributed over the days of the year.
- How does the probability vary with then number of people in the room?

Boundary Conditions

Birthday Function

- Let's design a function that provides a probability value in the range [0.0 .. 1.0] as a function of the number of persons in a room.
- What is the interface?
  - Function Name? Return Type? Parameter Name & Type?
- What is the contract?
  - What does it do? How *wide* should we make it?
  - *Essential Behavior*? (What must happen on valid input?)
  - *Undefined Behavior*? (What input values are not

Boundary Conditions

sameBirthday

```
double sameBirthday(int numPeople);
    // Return the probability that at least
    // two out of the specified (randomly-
    // chosen) 'numPeople' were born on the
    // same day of the same month.  People
    // born on February 29th are excluded.
    // The behavior is undefined unless
    // '0 <= numPeople'.
```

Boundary Conditions

sameBirthday

```
P(1) = 0
P(2) = 0/365 +  365/365   * 1/365
P(3) = 1/365 +  364/365   * 2/365
P(4) = P(3)  + (1 - P(3)) * 3/365
P(5) = P(4)  + (1 - P(4)) * 4/365
. . .
P(363) = P(362) + (1 - P(362)) * 362/365
P(364) = P(363) + (1 - P(363)) * 363/365
P(365) = P(364) + (1 - P(364)) * 364/365
P(366) = P(365) + (1 - P(365)) * 365/365 = 1
P(367) = 1
```

Boundary Conditions

sameBirthday

```
double sameBirthday(int numPeople)
{
    assert(0 <= numPeople);

    if (numPeople > 365) {
        return 1.0;
    }

    double probability = 0.0;

    for (int i = 1; i < numPeople; ++i) {
        probability += (1.0 - probability) * i / 365.0;
    }

    return probability;
}
```

Boundary Conditions

sameBirthday

- What kinds of values should we test first?

Boundary Conditions

sameBirthday

- What kinds of values should we test first?
  - The values near the boundaries.

Boundary Conditions

sameBirthday

- What kinds of values should we test first?
  - The values near the boundaries.
- Why?

Boundary Conditions

sameBirthday

- What kinds of values should we test first?
  - The values near the boundaries.
- Why?
  - Most likely to expose errors; easy to calculate by hand.

Boundary Conditions

sameBirthday

- What kinds of values should we test first?
  - The values near the boundaries.
- Why?
  - Most likely to expose errors; easy to calculate by hand.
- What are the *external* interface boundary values?

Boundary Conditions

sameBirthday

- What kinds of values should we test first?
  - The values near the boundaries.
- Why?
  - Most likely to expose errors; easy to calculate by hand.
- What are the *external* interface boundary values?
  - 0 and INT_MAX

Boundary Conditions

sameBirthday

- What kinds of values should we test first?
  - The values near the boundaries.
- Why?
  - Most likely to expose errors; easy to calculate by hand.
- What are the *external* interface boundary values?
  - 0 and INT_MAX
- What are the *internal* interface boundary values?

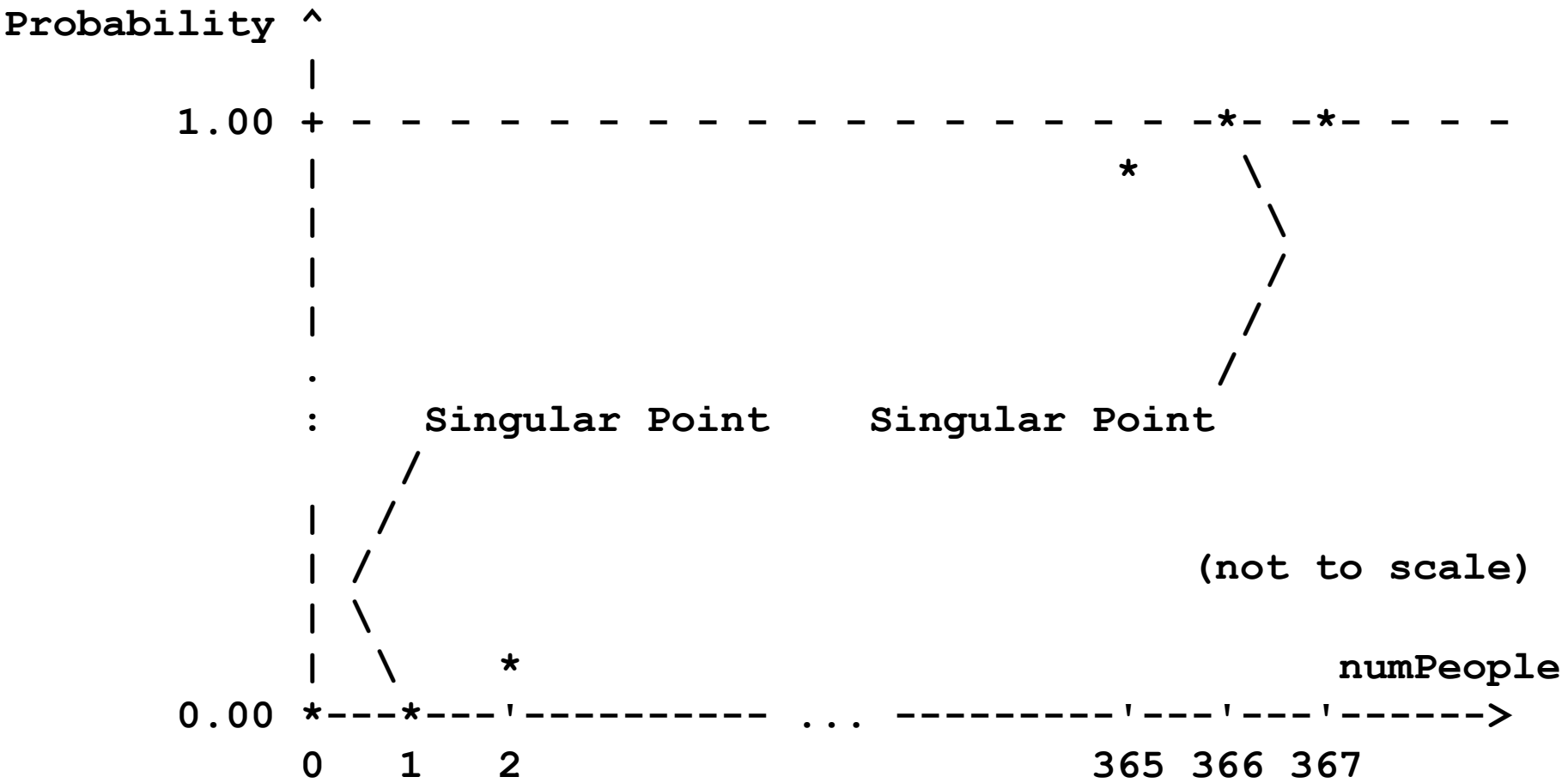
Boundary Conditions

sameBirthday

- What kinds of values should we test first?
  - The values near the boundaries.
- Why?
  - Most likely to expose errors; easy to calculate by hand.
- What are the *external* interface boundary values?
  - 0 and INT_MAX
- What are the *internal* interface boundary values?
  - 1 & 2 and 365 & 366 (367 *may* be an imp.

Boundary Conditions

sameBirthday



Boundary Conditions

sameBirthday

```
ASSERT(0.0 == sameBirthday(0)); // external interface boundary
ASSERT(0.0 == sameBirthday(1)); // internal singular point
ASSERT(0.0  < sameBirthday(2)); // internal interface boundary


ASSERT(1.0  > sameBirthday(365));// internal interface boundary
ASSERT(1.0 == sameBirthday(366));// internal singular point
ASSERT(1.0 == sameBirthday(367));// internal interface boundary

ASSERT(1.0 == sameBirthday(INT_MAX);
                                  // external interface boundary
                                  // also platform boundary
```

Boundary Conditions

sameBirthday

```
ASSERT(0.0 == sameBirthday(0)); // external interface boundary
ASSERT(0.0 == sameBirthday(1)); // internal singular point
ASSERT(0.0  < sameBirthday(2)); // internal interface boundary


ASSERT(1.0  > sameBirthday(365));// internal interface boundary
ASSERT(1.0 == sameBirthday(366));// internal singular point
ASSERT(1.0 == sameBirthday(367));// internal interface boundary

ASSERT(1.0 == sameBirthday(INT_MAX);
                                  // external interface boundary
                                  // also platform boundary
```

Boundary Conditions

sameBirthday

```
ASSERT(0.0 == sameBirthday(0)); // external interface boundary
ASSERT(0.0 == sameBirthday(1)); // internal singular point
ASSERT(0.0  < sameBirthday(2)); // internal interface boundary


ASSERT(1.0  > sameBirthday(365));// internal interface boundary
ASSERT(1.0 == sameBirthday(366));// internal                 point
ASSERT(1.0 == sameBirthday(367                              e boundary

ASSERT(1.                                                  // external interface boundary
                                                           // also platform boundary
```

Boundary Conditions

sameBirthday

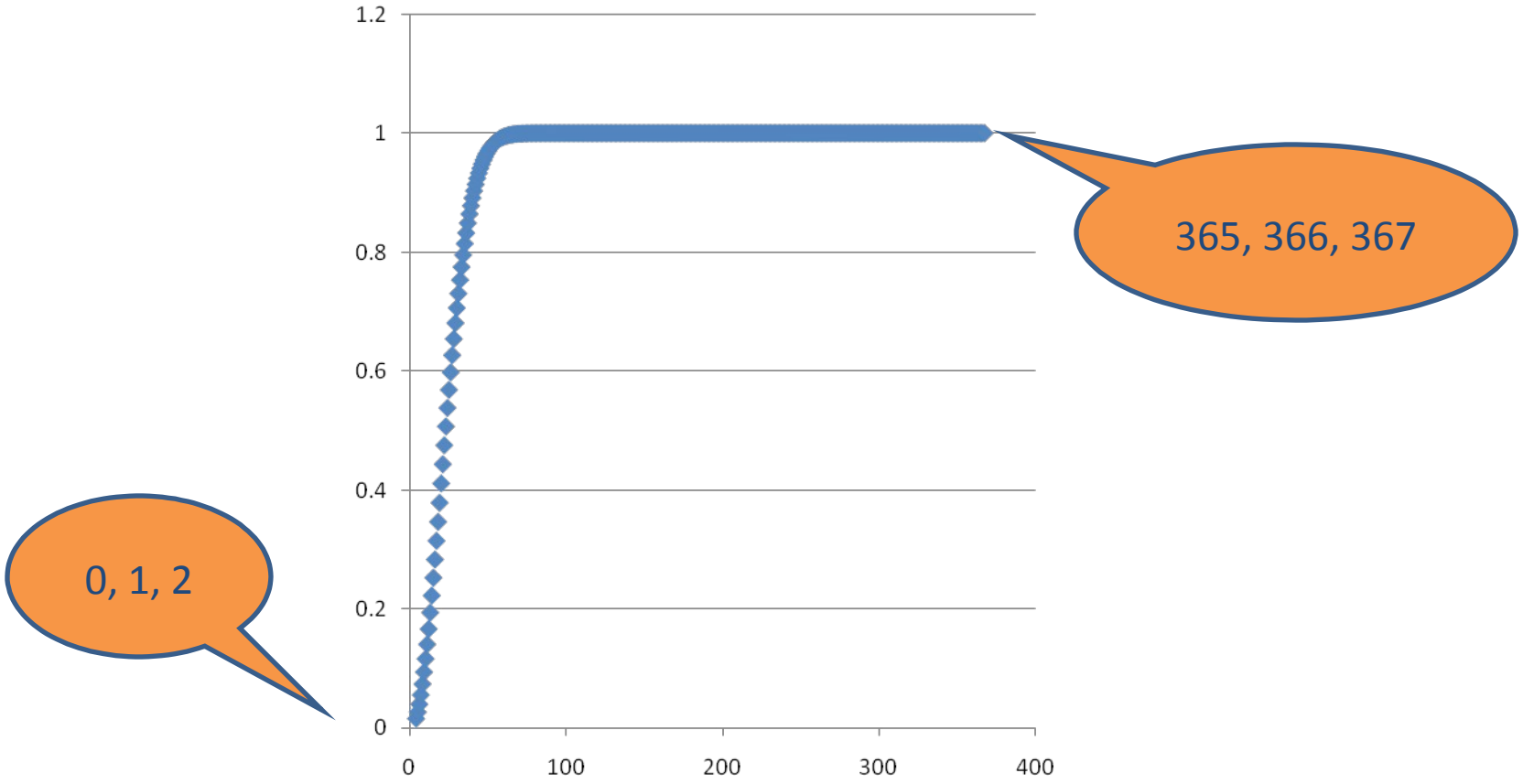
```
ASSERT(0.0 == sameBirthday(0)); // external interface boundary
ASSERT(0.0 == sameBirthday(1)); // internal singular point
ASSERT(0.0  < sameBirthday(2)); // internal interface boundary


ASSERT(1.0  > sameBirthday(365));// internal interface boundary
ASSERT(1.0 == sameBirthday(366));// internal                 r point
ASSERT(1.0 == sameBirthday(367                              e boundary

ASSERT(1.                                  // external interface boundary
                                           // a sc platform boundary
```

Assertion Failure!

WHY?

Boundary Conditions

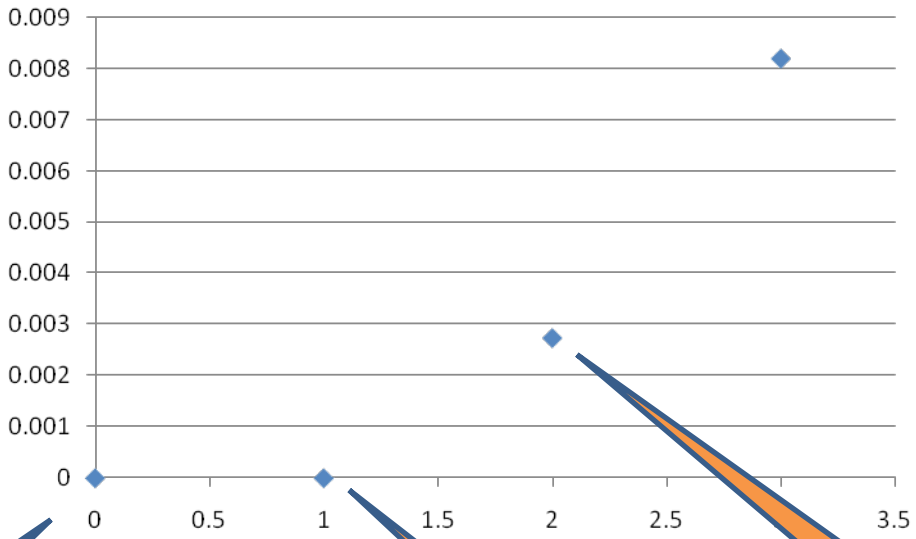
sameBirthday



Boundary Conditions

sameBirthday

0	0
1	0
2	0.00274
3	0.008204
4	0.016356



0: OK

1: OK

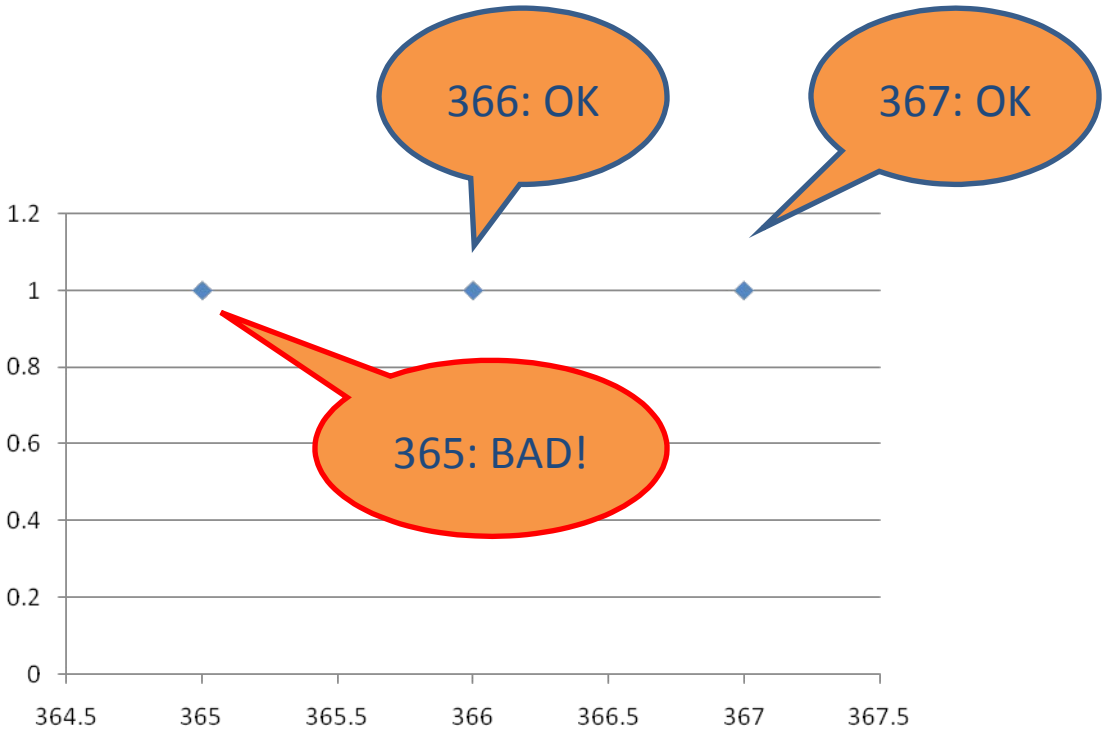
2: OK

2. Designing Component-Level (function) Tests

Boundary Conditions

sameBirthday

365	1
366	1
367	1



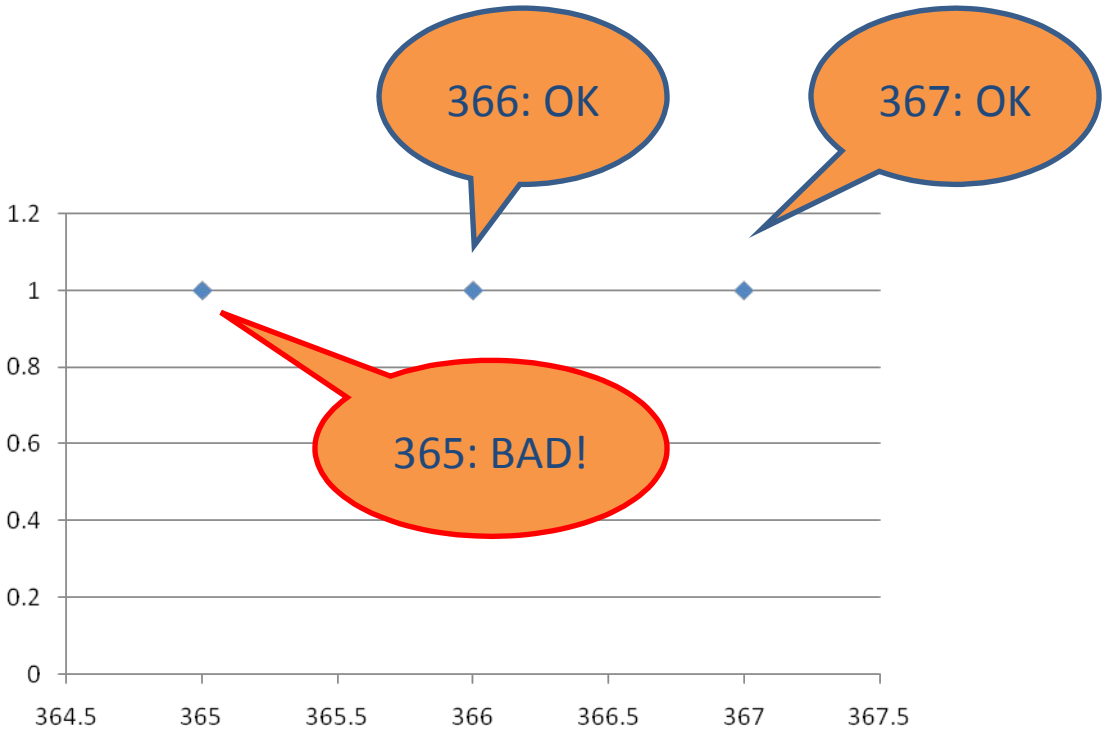

```
ASSERT(1.0 > sameBirthday(365)); // FAIL
```

2. Designing Component-Level (function) Tests

Boundary Conditions

sameBirthday

365	1
366	1
367	1

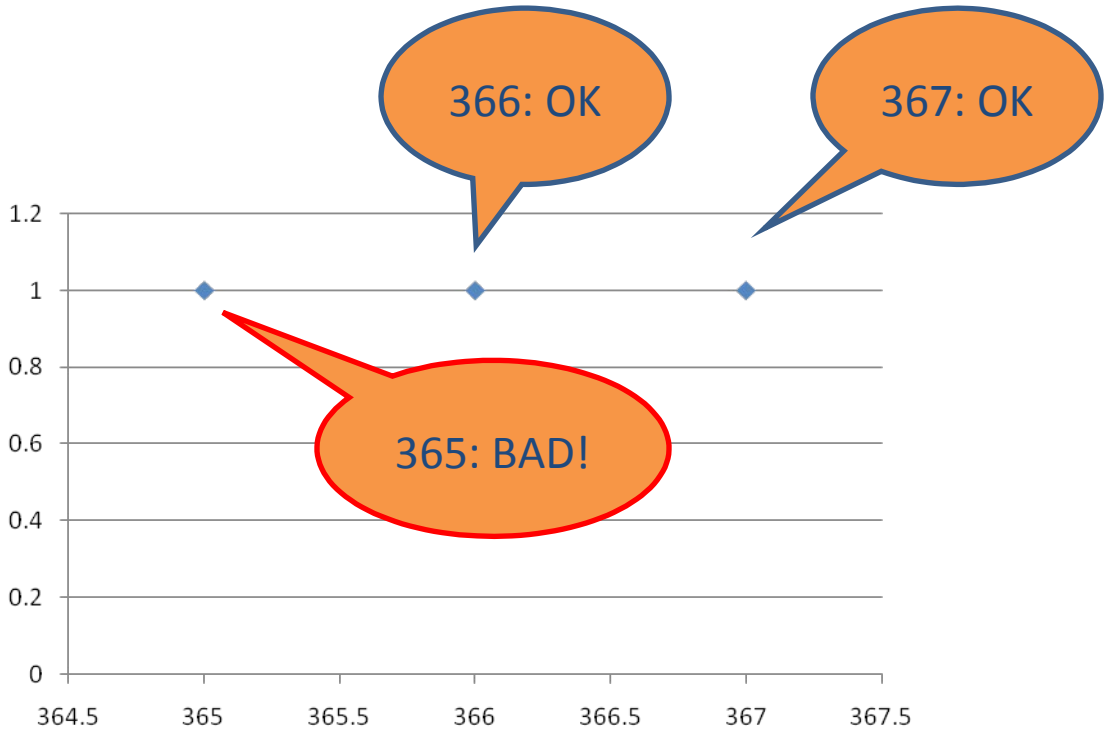



```
ASSERT(1.0 > sameBirthday(364)); // FAIL
```

Boundary Conditions

sameBirthday

365	1
366	1
367	1

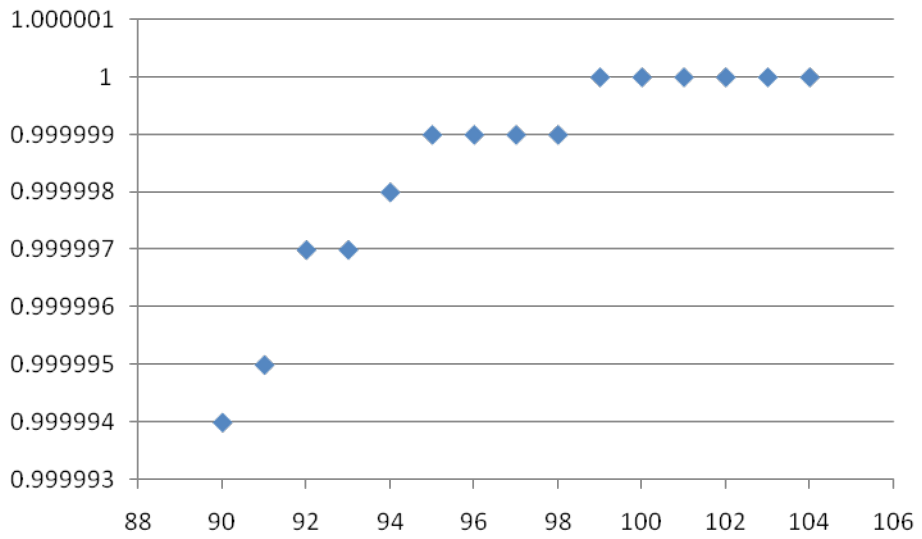



```
ASSERT(1.0 > sameBirthday(363)); // FAIL
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

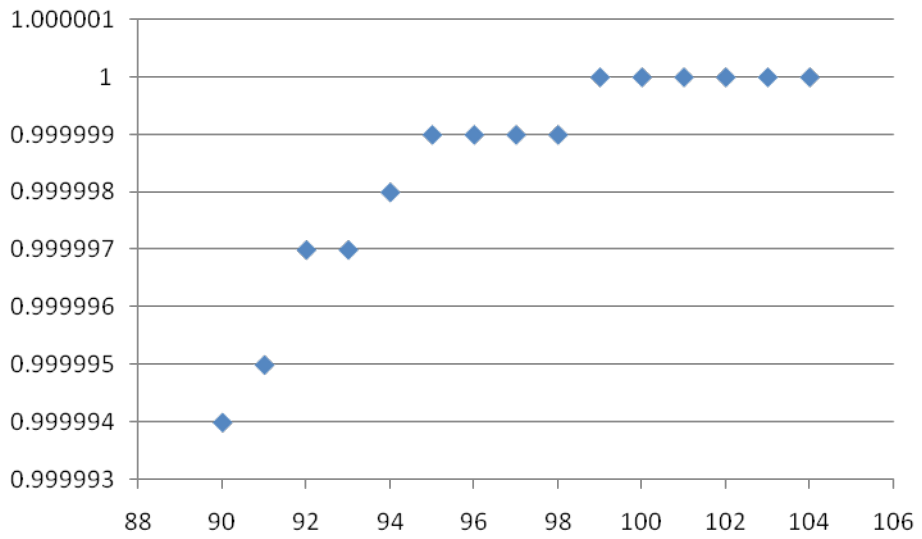



```
ASSERT(1.0 > sameBirthday(98)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

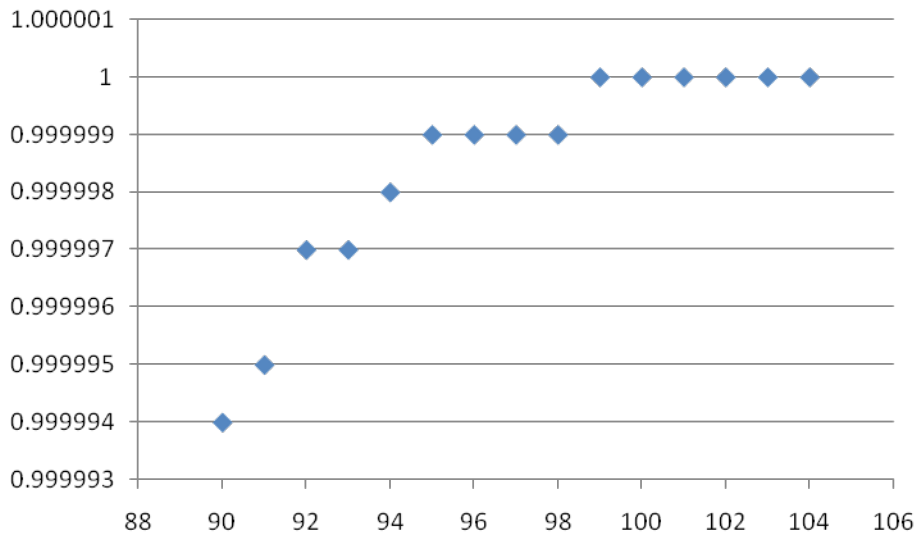



```
ASSERT(1.0 > sameBirthday(99)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

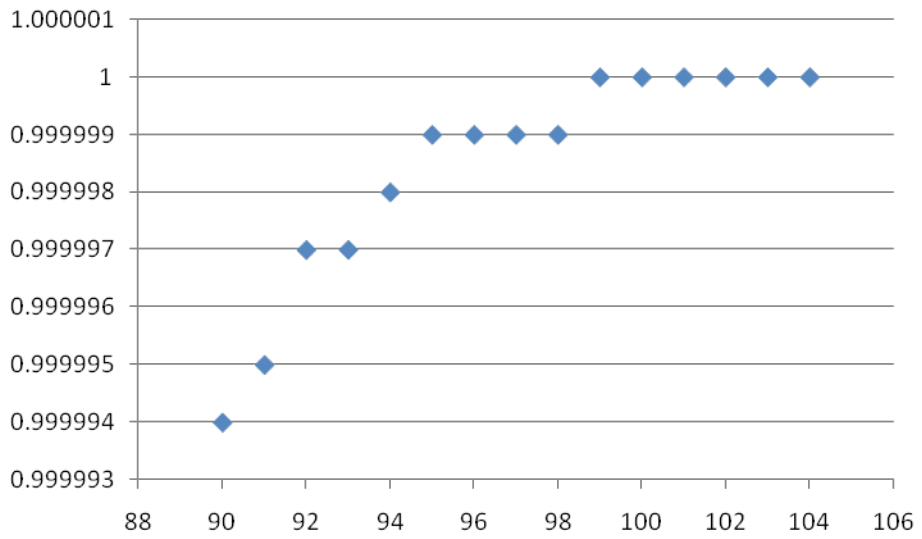



```
ASSERT(1.0 > sameBirthday(100)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

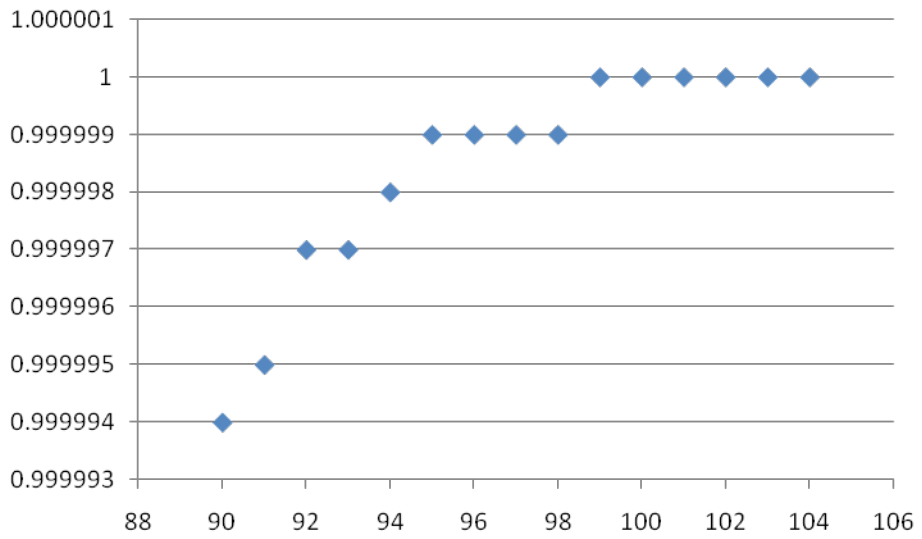



```
ASSERT(1.0 > sameBirthday(101)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

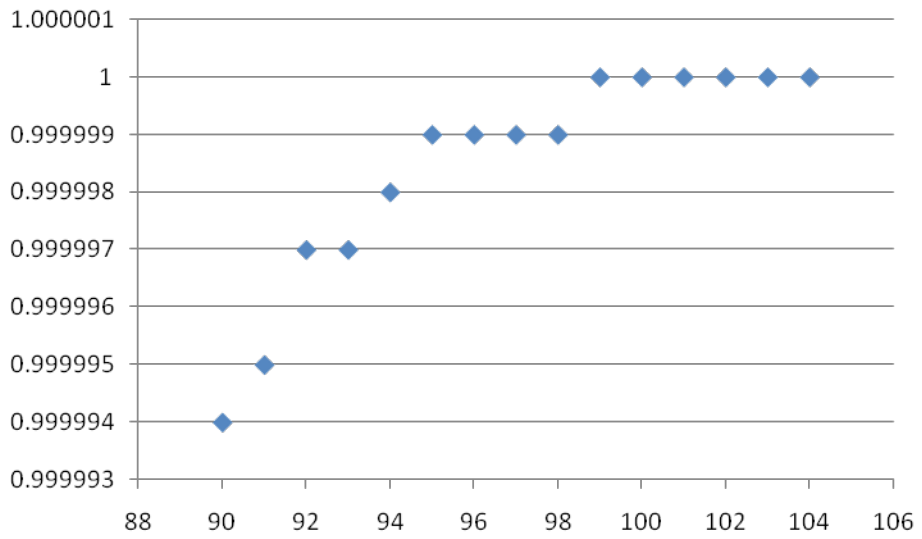



```
ASSERT(1.0 > sameBirthday(102)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

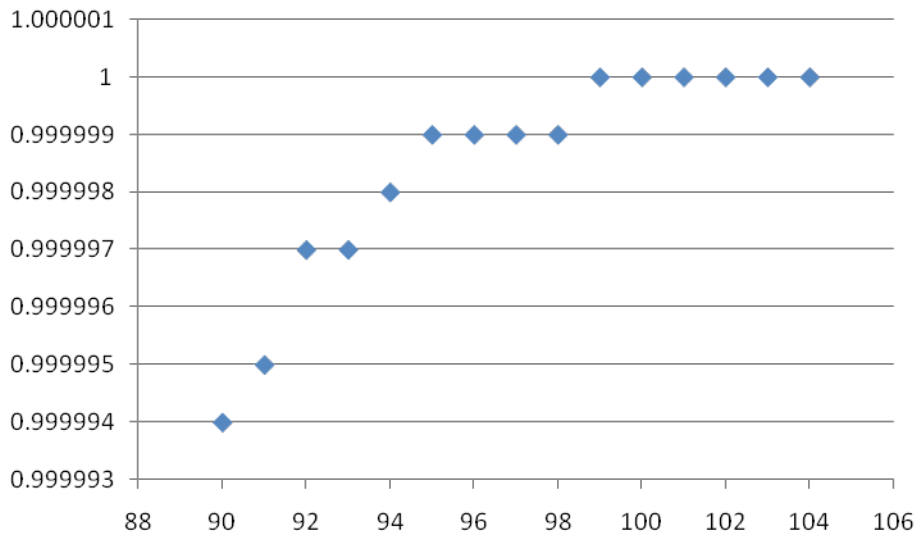



```
ASSERT(1.0 > sameBirthday(103)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

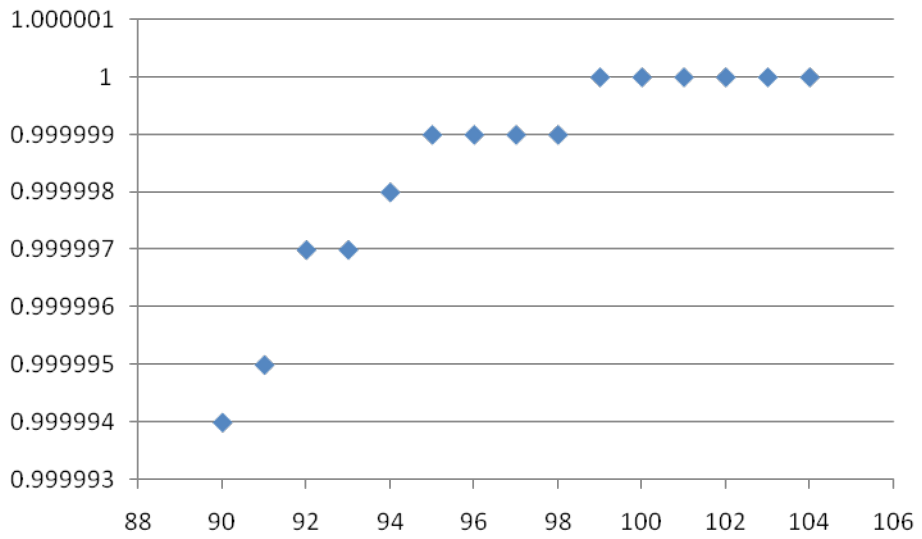



```
ASSERT(1.0 > sameBirthday(110)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

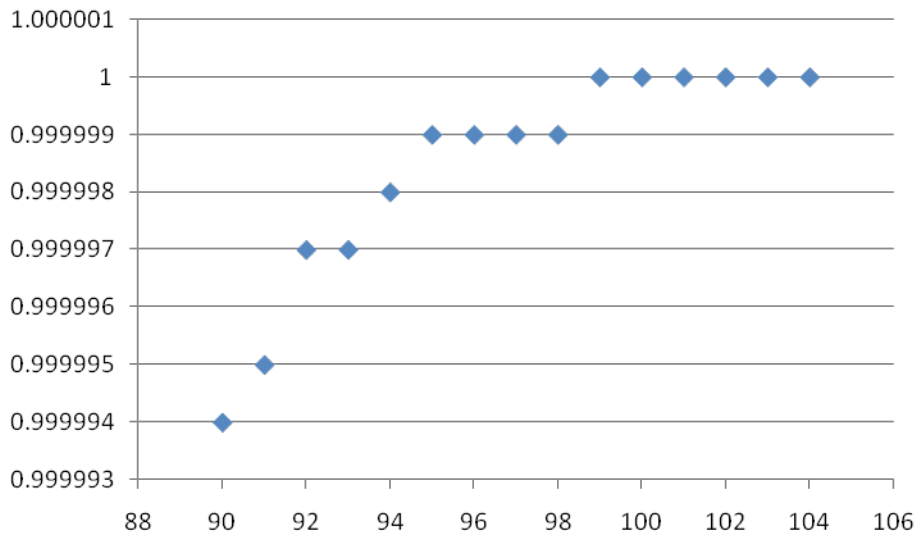



```
ASSERT(1.0 > sameBirthday(120)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

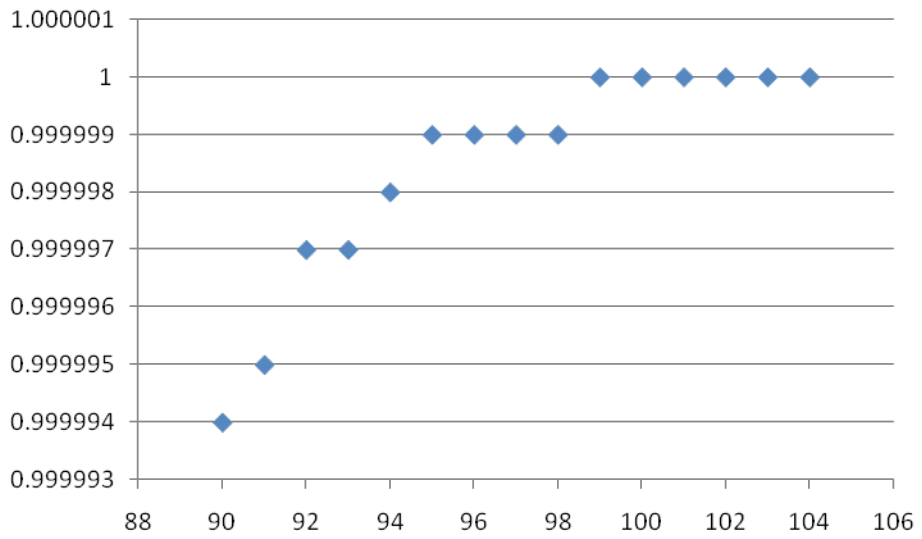



```
ASSERT(1.0 > sameBirthday(150)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

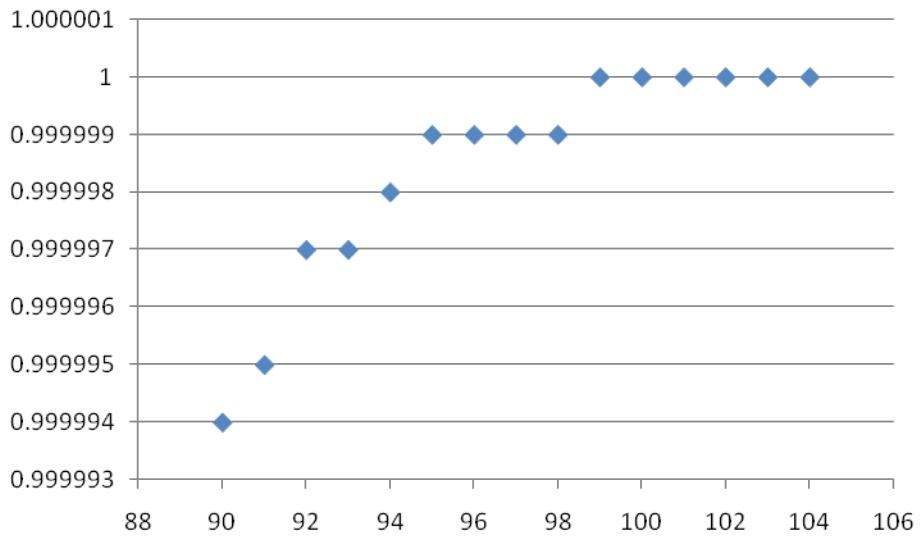



```
ASSERT(1.0 > sameBirthday(200)); // FAIL
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

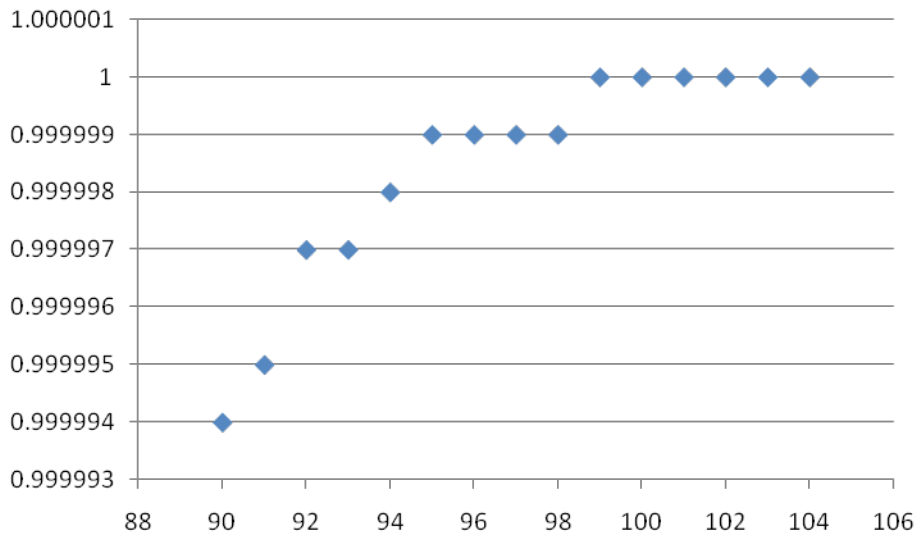



```
ASSERT(1.0 > sameBirthday(175)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

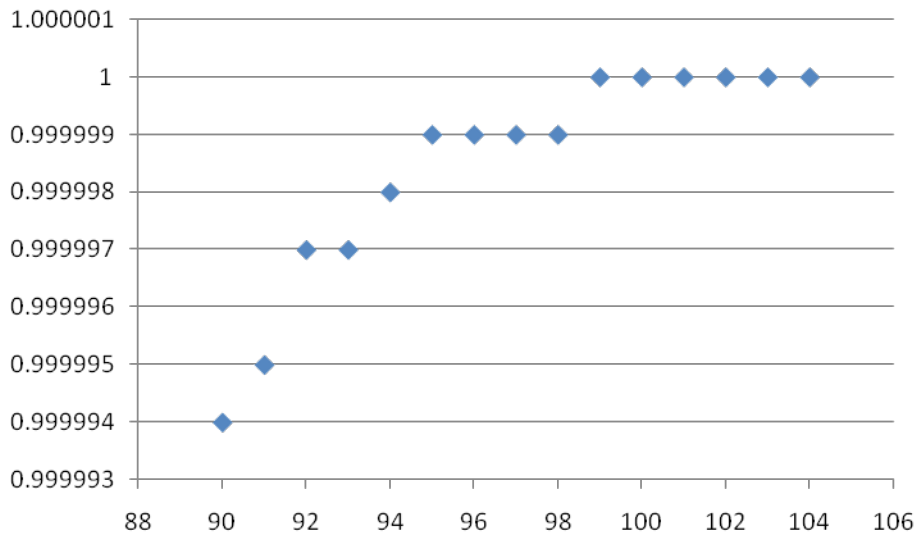



```
ASSERT(1.0 > sameBirthday(185)); // FAIL
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

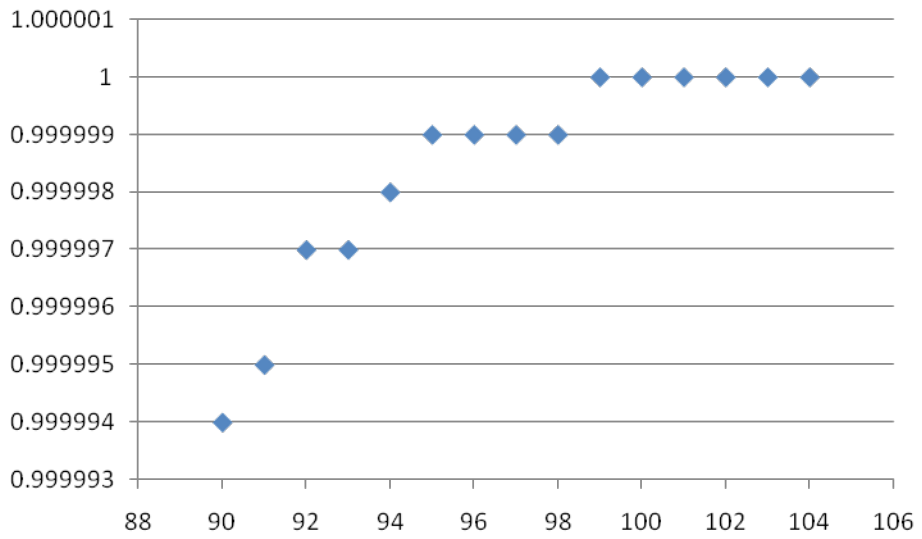



```
ASSERT(1.0 > sameBirthday(180)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

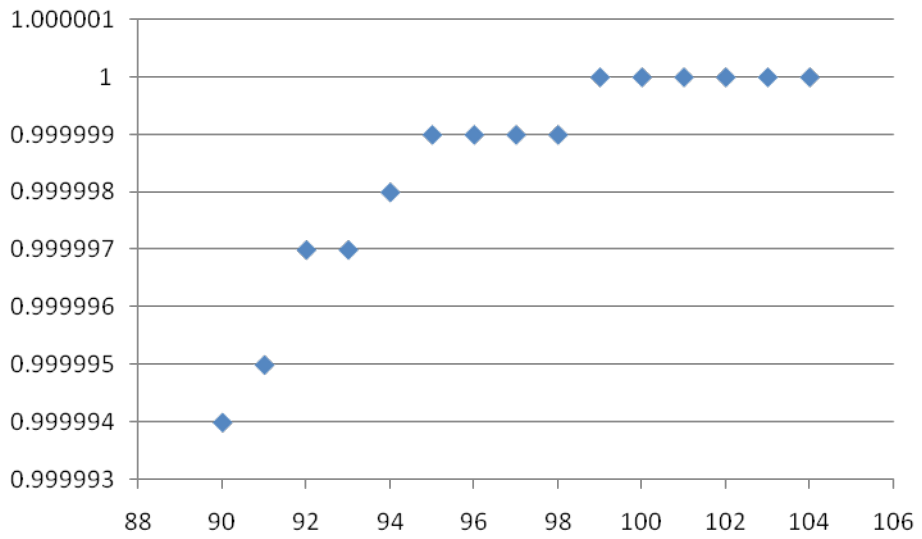



```
ASSERT(1.0 > sameBirthday(182)); // SUCCESS
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1

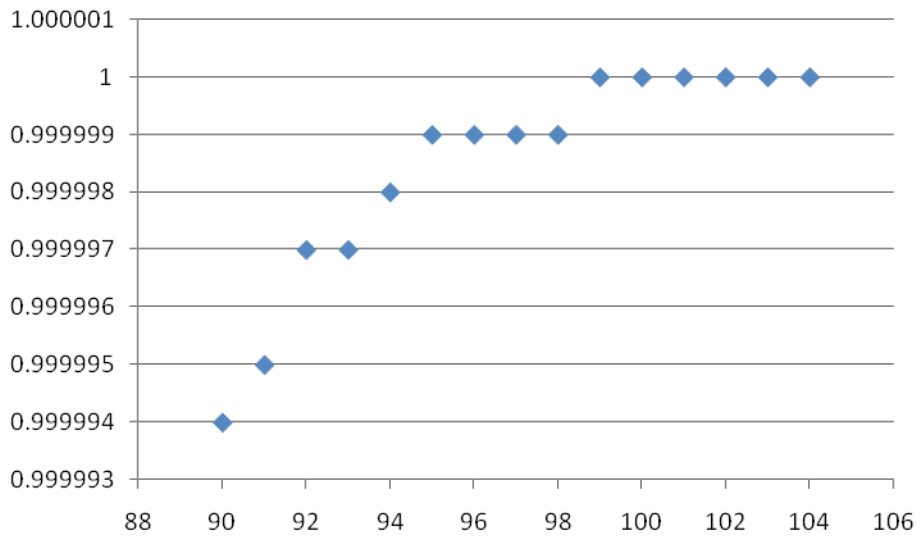



```
ASSERT(1.0 > sameBirthday(184)); // FAIL
```

Boundary Conditions

sameBirthday

90	0.999994
91	0.999995
92	0.999997
93	0.999997
94	0.999998
95	0.999999
96	0.999999
97	0.999999
98	0.999999
99	1
100	1
101	1
102	1
103	1
104	1
105	1




```
ASSERT(1.0 > sameBirthday(183)); // SUCCESS
```

Boundary Conditions

sameBirthday

So what's wrong with our implementation?

Boundary Conditions

sameBirthday

```
double sameBirthday(int numPeople)
{
    assert(0 <= numPeople);

    if (numPeople > 365) {
        return 1.0;
    }

    double probability = 0.0;

    for (int i = 1; i < numPeople; ++i) {
        probability += (1.0 - probability) * i / 365.0;
    }

    return probability;
}
```

Boundary Conditions

sameBirthday

We have encountered a ***platform-imposed*** boundary condition between 183 and 184 people!

Boundary Conditions

Using a `double` to represent "Almost One"

IEEE-754

```
sign-bit  11-bit exponent                 52-bit mantissa
 /       /                               /
0 01111111110 1111111111111111111111111111111111111111111111111111

+ 2^-1 * [ 1 + 1/2 + 1/4 + 1/8 + 1/16 + ...+ 1 * 2^-51 + 1 * 2^-52 ]

      Almost One: 1 - 2^-52 = 0.9999 9999 9999 9999


0 01111111111 0000000000000000000000000000000000000000000000000000

+ 2^0 * [ 1 + 0/2 + 0/4 + 0/8 + 0/16 + ... + 0 * 2^-51 + 0 * 2^-52 ]

      One: 1 = 1.0000 0000 0000 0000
```

Boundary Conditions

Using a `double` to represent "Almost Zero"

IEEE-754

```
sign-bit  11-bit exponent                 52-bit mantissa
 /      /                                 /

0 00000000000 0000000000000000000000000000000000000000000000000001

+ 2^-1022 * [ 0/2 + 0/4 + 0/8 + 0/16 + ... + 0 * 2^-51 + 1 * 2^-52 ]

      Almost Zero: 2^-1074 = 4.9406 5645 8412 466 * 10^-324


0 00000000000 0000000000000000000000000000000000000000000000000000

+ 2^-1022 * [ 0/2 + 0/4 + 0/8 + 0/16 + ... + 0 * 2^-51 + 0 * 2^-52 ]

      Zero: 0 = zero exponent and zero mantissa (sign is irrelevant)
```

Boundary Conditions

sameBirthday

- How do we fix our implementation for `sameBirthday`?

Boundary Conditions

sameBirthday

- How do we fix our implementation for `sameBirthday`?
  - ➢We can't: No viable implementation exists!

Boundary Conditions

sameBirthday

- How do we fix our implementation for `sameBirthday`?
  - We can't: No viable implementation exists!
- What should be do?

Boundary Conditions

sameBirthday

- How do we fix our implementation for `sameBirthday`?
  - We can't: No viable implementation exists!
- What should be do?
  - Rework our interface & contract.

Boundary Conditions

sameBirthday

- How do we fix our implementation for `sameBirthday`?
  - We can't: No viable implementation exists!
- What should be do?
  - Rework our interface & contract.
- How?

Boundary Conditions

sameBirthday

- How do we fix our implementation for `sameBirthday`?
  - We can't: No viable implementation exists!
- What should be do?
  - Rework our interface & contract.
- How?
  - Use what we've learned about non-uniformity in the dynamic range of IEEE-754 `double`

Boundary Conditions

uniqueBirthday

Boundary Conditions

uniqueBirthday

- The probability P(N) that no two of N people have the same birthday can be expressed as

  P(N) = 365/365 * 364/365 * 363/365 * 362/365 * … * (366 – N)/365

  = **365!/[N! * 365^N]**

Boundary Conditions

uniqueBirthday

- The probability P(N) that no two of N people have the same birthday can be expressed as

  P(N) = 365/365 * 364/365 * 363/365 * 362/365 * … * (366 – N)/365

  = **365!/[N! * 365^N]**

- **P(365)** = 365!/365^365 ~= e^-365 * sqrt(2 * pi * 365) *Sterling's approx.*

  ~= **1.45 * 10^-157**

  >> 4.9406 5645 8412 466 * 10^-324

Boundary Conditions

uniqueBirthday

- The probability P(N) that no two of N people have the same birthday can be expressed as

  P(N) = 365/365 * 364/365 * 363/365 * 362/365 * … * (366 – N)/365

  = **365!/[N! * 365^N]**

- **P(365)** = 365!/365^365 ~= e^-365 * sqrt(2 * pi * 365) *Sterling's approx.*

  ~= **1.45 * 10^-157**

  >> 4.9406 5645 8412 466 * 10^-324

- With **`double uniqueBirthday(int numPeople)`**, we can at least ***represent*** the results for the entire range of valid inputs.

Boundary Conditions

uniqueBirthday

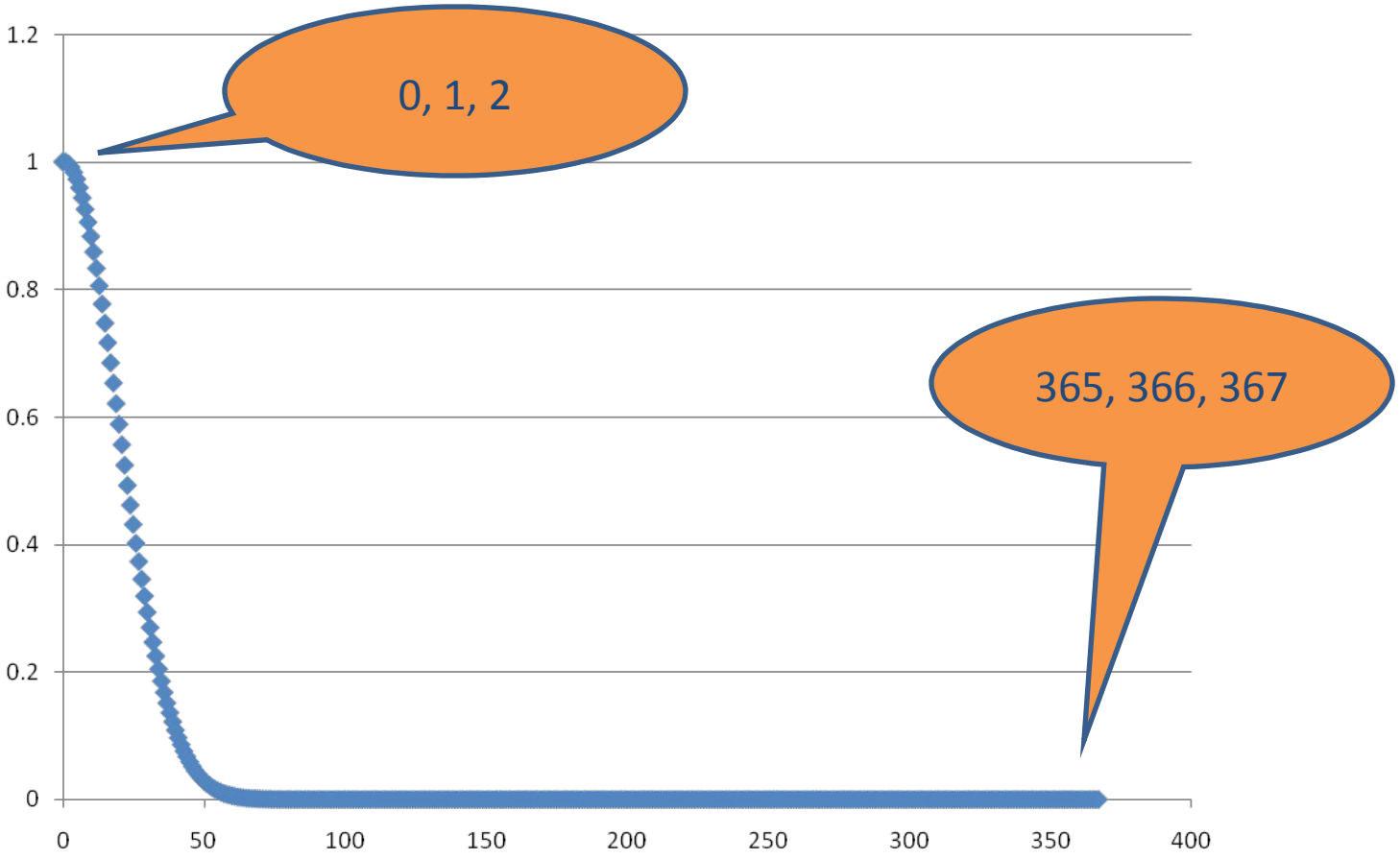
```
double uniqueBirthday(int numPeople);
    // Return the probability that no two
    // of the specified (randomly-chosen)
    // 'numPeople' were born on the same
    // day of the same month.  People born
    // on February 29th are excluded. The
    // behavior is undefined unless
    // '0 <= numPeople'.
```

Boundary Conditions

uniqueBirthday

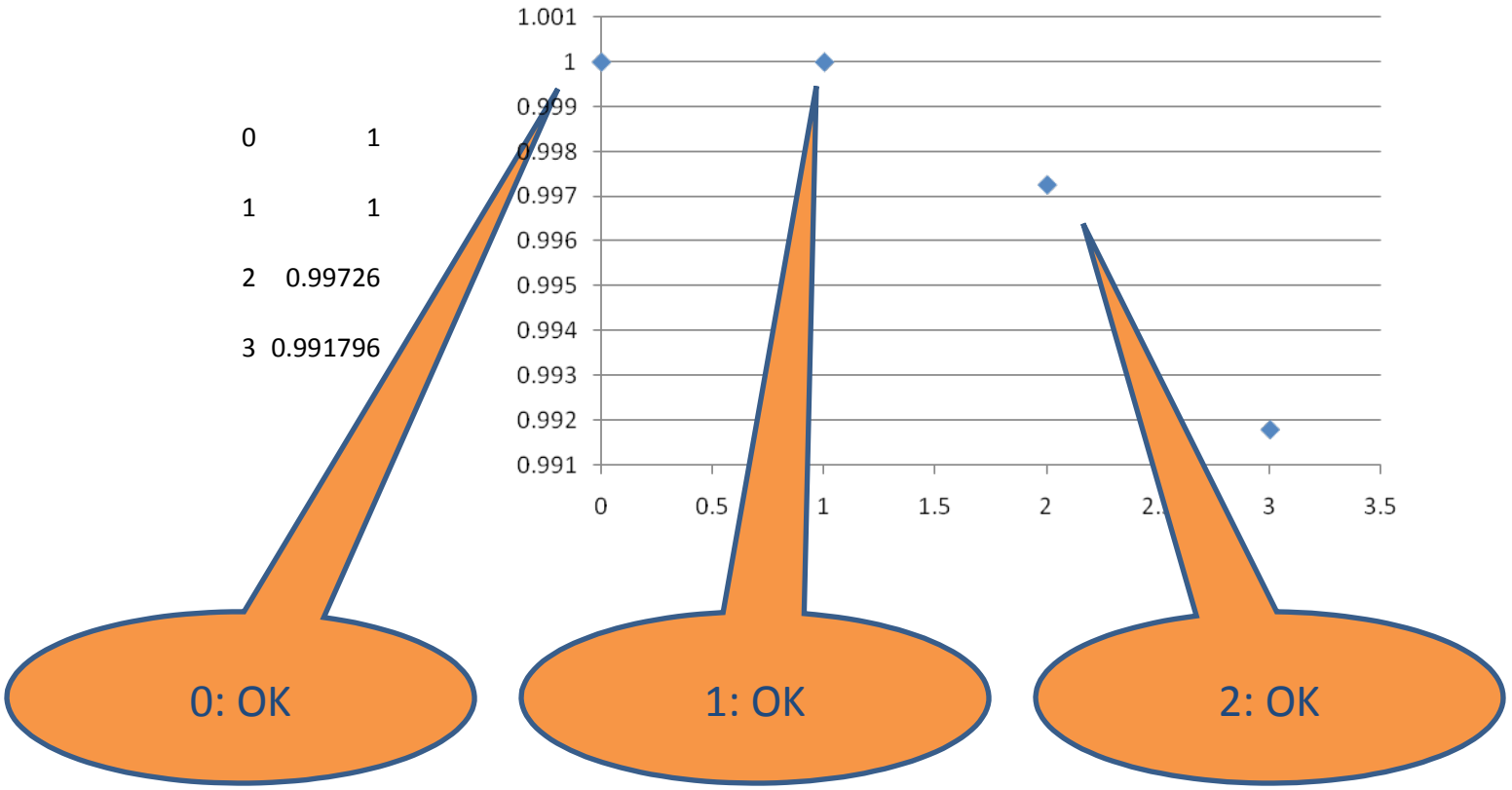
```
#include <assert.h>

double uniqueBirthday(int numPeople)
{
    assert(0 <= numPeople);

    if (numPeople > 365) {
        return 0.0;
    }

    double probability   = 1.0;
    int    lastNumerator = 366 - numPeople;

    for (int i = 364; i >= lastNumerator; --i) {
        probability *= i/365.0;
    }
```

Boundary Conditions



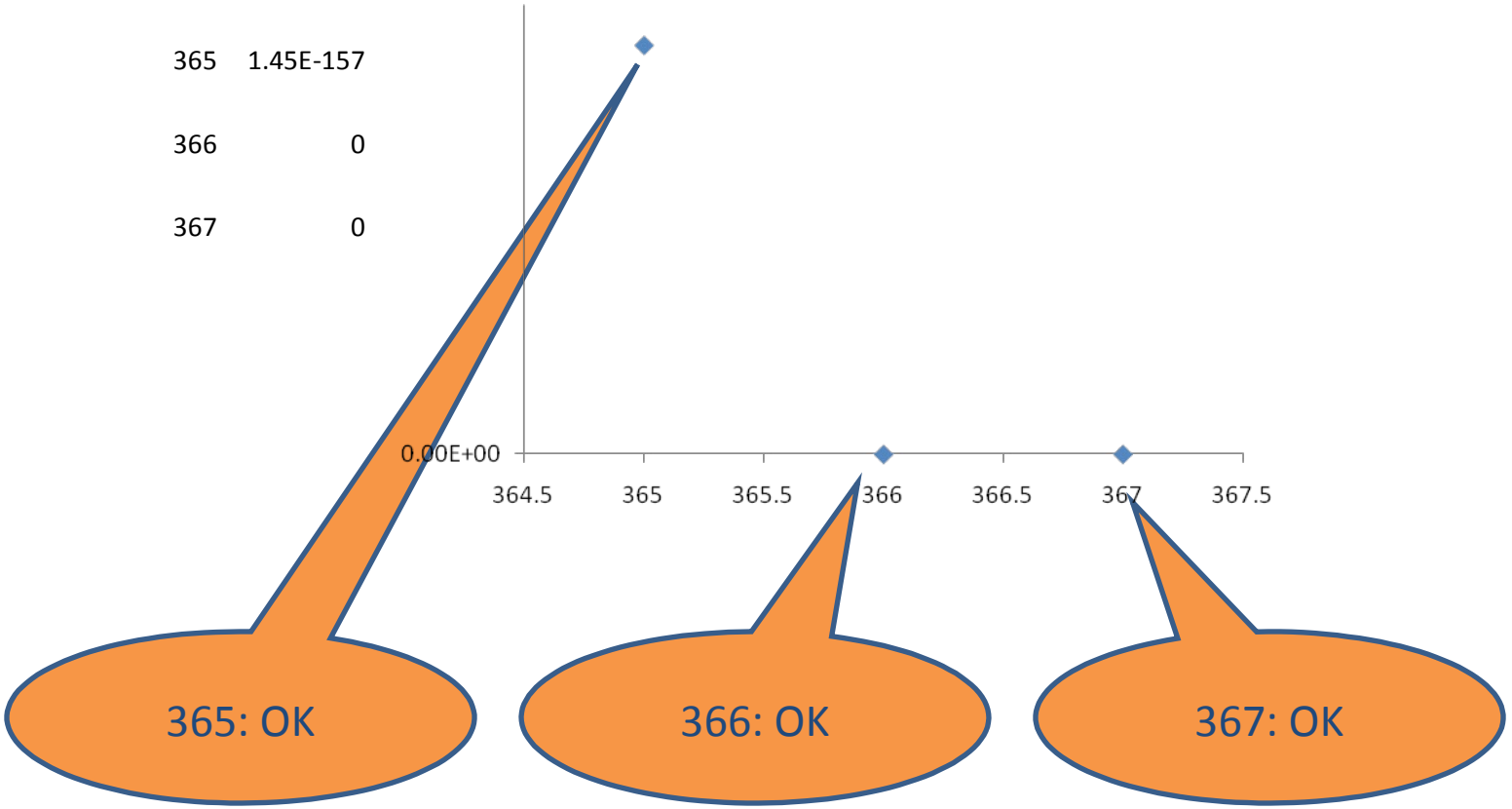
Boundary Conditions

uniqueBirthday



Boundary Conditions

uniqueBirthday



Boundary Conditions

Observations

Boundary Conditions

Observations

- We cannot expect to know everything ***a priori***.

Boundary Conditions

Observations

- We cannot expect to know everything ***a priori***.
- The purpose of our initial testing here was to verify certain basic assumptions.

Boundary Conditions

Observations

- We cannot expect to know everything ***a priori***.
- The purpose of our initial testing here was to verify certain basic assumptions.
- Creating just a few thoughtful tests at critical ***interface*** *boundaries* helped us to discover important ***platform*** *boundaries*, which, in turn, led us to redesign our function's interface.

Boundary Conditions

Observations

- We cannot expect to know everything ***a priori***.
- The purpose of our initial testing here was to verify certain basic assumptions.
- Creating just a few thoughtful tests at critical ***interface*** *boundaries* helped us to discover important ***platform*** *boundaries*, which, in turn, led us to redesign our function's interface.
- Thorough testing does not necessarily tell us how to solve a problem; it does, however, alert us to when we haven't done it yet.

Conclusion

Conclusion

The End

Conclusion

The End

Birthday Surprise

Kevlin Henney

kevlin@curbralan.com

@KevlinHenney

Is it possible to implement *sameBirthday* so that it satisfies the contract as defined?

Yes. The first version satisfied the contract as stated.

But it didn't pass the tests!

Indeed...

When a test fails, it may fail because of a bug in our code...

And it may reveal mistaken assumptions in the test case.

OK, so is it possible to write *sameBirthday* so that it also passes the tests?

Yes.

```
double sameBirthday(int numPeople)
{
    double probability;
    if(numPeople < 0)
    {
        errno = EDOM;
        probability = NAN;
    }
    else if(numPeople > 365)
    {
        probability = 1;
    }
    else
    {
        probability = 0;
        for(int i = 1; i < numPeople; ++i)
            probability += (1 – probability) * i / 365.0;
        if(probability == 1)
            --*(unsigned long long *) &probability;
    }
    return probability;
}
```

```
double sameBirthday(int numPeople)
{
    double probability;
    if(numPeople < 0)
    {
        errno = EDOM;
        probability = NAN;
    }
    else if(numPeople > 365)
    {
        probability = 1;
    }
    else
    {
        probability = 0;
        for(int i = 1; i < numPeople; ++i)
            probability += (1 – probability) * i / 365.0;
        if(probability == 1)
            --*(unsigned long long *) &probability;
    }
    return probability;
}
```

£20