A Cost Benefit Tool for Fire Protection Engineers: An Analysis

Chris Salter, Dino Bouchlaghem, Prof. G. Ramachandran

C.Salter@lboro.ac.uk, Loughborough University

Abstract. This paper outlines the background work for a cost benefit design tool which is aimed at those within the fire engineering design sector. The paper discusses potential programming languages, design features to watch out for and possible development methodologies. It also critically analyses previous design tools and provides methods to build upon them.

Keywords: fire engineering, cost benefit tool

1 Introduction

Fire engineers are employed by architects and design firms to allow proposed building plans to be designed more cost effectively than following the building regulations exactly would allow. Whilst the fire engineers are currently successful in this role, there is room for improvement on the savings that fire engineers can achieve. Previous work by Ramachandran [13,12] has identified aspects of fire engineering that benefit from the addition of cost effective installation of protection measures.

Building on the work done by Ramachandran, a cost effectiveness design tool is proposed that incorporates fire incident data as statistical background for the calculations within the tool. A cost effectiveness design tool has previously been constructed in work done by BRE [7], however, this tool focussed on schools and sprinkler systems. More recently, the National Institute of Standards and Technology (NIST) in America have released a cost benefit analysis tool for the installation of sprinkler systems in buildings [11]. The design tool proposed by the authors will be able to calculate costs benefits for all building types, with the exception of residential and dwellings, which are not part of the study.

2 Questionnaire Results

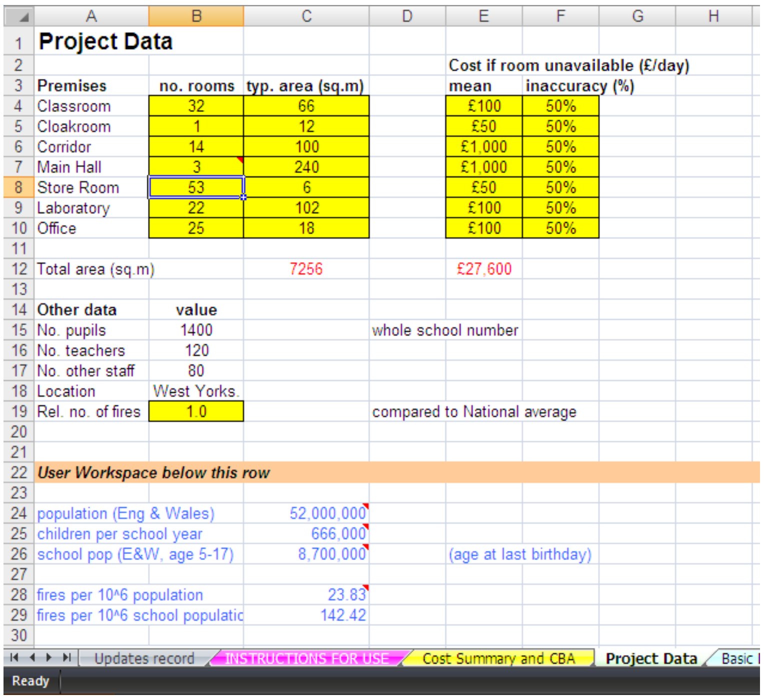
In a conference paper (currently in press), the results of a questionnaire and interviews carried out as part of this research are discussed. A summary of the findings of that paper are noted here.

- The majority of fire engineers consider other aspects other than life safety (such as property protection) in the design
- Fire engineers will mainly use building regulations and design guides to aid in validation of fire engineered solutions
- Cost was not a critical design factor in the majority of responses, yet the majority considered the costs of each individual system they specified for the building.
- A decision support system to aid cost benefit decisions does not currently exist, except the one described above that focusses on sprinklers and schools
- Fire engineers would be willing to use such a program if it existed.

Based on these results, the proposed design tool will take into account the individual systems that the fire engineers will potentially use, will be able to provide a cost overview of the proposed systems and offer cost benefit analysis on the proposed systems.

3 Previous Tools

As mentioned in the Introduction, there are a few different tools available to fire engineering professionals that focus on aspects that the proposed decision support system will cover. These have been programmed in a variety of different ways. The merits and drawbacks of these tools are discussed below.



	A	B	C	D	E	F	G	H
1	**Project Data**							
2					**Cost if room unavailable (£/day)**			
3	**Premises**	**no. rooms**	**typ. area (sq.m)**		**mean**	**inaccuracy (%)**		
4	Classroom	32	66		£100	50%		
5	Cloakroom	1	12		£50	50%		
6	Corridor	14	100		£1,000	50%		
7	Main Hall	3	240		£1,000	50%		
8	Store Room	53	6		£50	50%		
9	Laboratory	22	102		£100	50%		
10	Office	25	18		£100	50%		
11								
12	Total area (sq.m)		7256		£27,600			
13								
14	**Other data**	**value**						
15	No. pupils	1400		whole school number				
16	No. teachers	120						
17	No. other staff	80						
18	Location	West Yorks.						
19	Rel. no. of fires	1.0		compared to National average				
20								
21								
22	***User Workspace below this row***							
23								
24	population (Eng & Wales)		52,000,000					
25	children per school year		666,000					
26	school pop (E&W, age 5-17)		8,700,000		(age at last birthday)			
27								
28	fires per 10^6 population		23.83					
29	fires per 10^6 school populatic		142.42					
30								

Fig. 1. BRE Tool Data Input

3.1 BRE Sprinklers in Schools Tool

The tool developed and made by BRE [7] is developed in Excel. It's main drawback of this tool is the tools user interface. Figure 1 shows the main data input worksheet for the tool.
The colour of the font and highlighting of cells has been used to indicate to the user where changes should be made, however, no help is provided on the current worksheet to tell the user what each colour or highlighting corresponds to though the tool does contain instructions on this at the beginning of the document. Therefore, if you wish to consult the documentation during use of the tool, you are required to switch between the current worksheet and the instructions worksheet to view them. Putting the help on the same worksheet as the user is using prevents this switching action. This breaks rule 8 of Shneiderman and Plaisant golden rules on reducing short term memory load which are discuss in later in the paper.

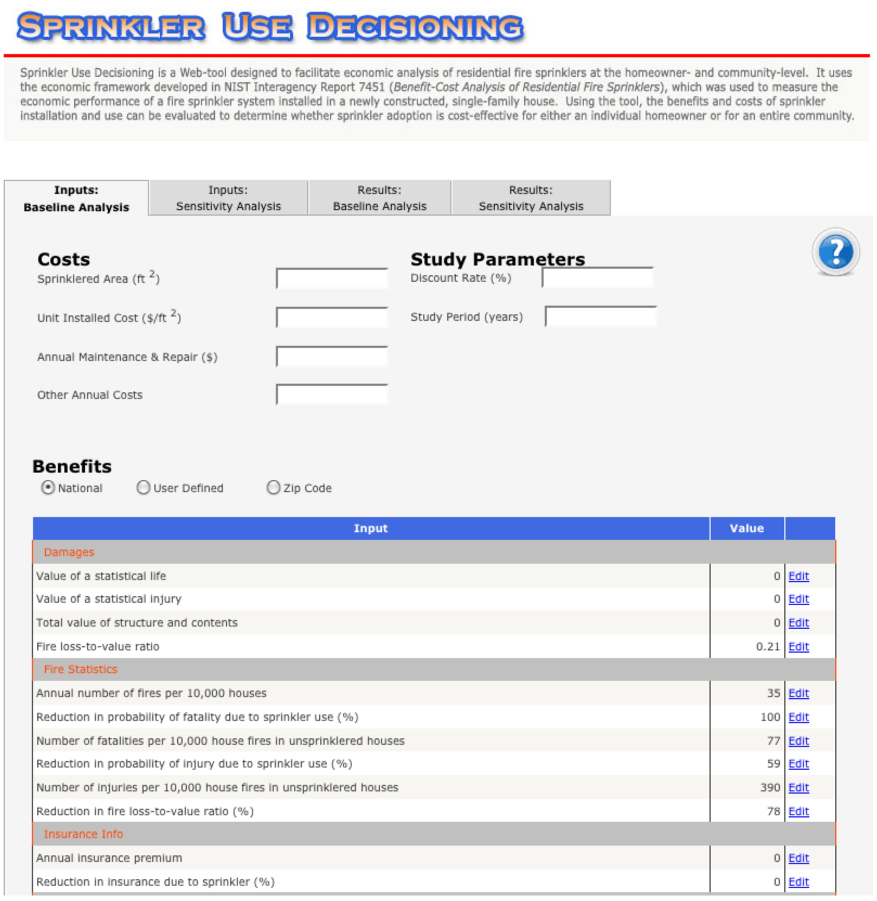
Data within the worksheets is not protected - a warning on the instructions worksheet states that values in red should not be changed - however no validation or write protection is built into the sheet and therefore a user can deliberately or accidentally change these values and give a completely different result than might be expected. Excel supports the locking of cell contents except those defined by the programmer to allow changing. In this manner, the sheet can be locked and the end user can only write in the cells that are unlocked - this would prevent errors due to accidental formula or base calculation figures being edited and is something that should be considered for a tool. However, the flexibility to add or change details should remain within the tool so that if required, an advanced user can change these values to suit.

An Excel document is very easy to save. When doing so, it will save everything within the file, such as inputs and the outputs. However, it is very easy to save over a document. Whilst the user and/or the business the user works for should keep backups of files, the program should make it easy for users to save the document and not overwrite any default data or if it occurs, make it easy to reset the sheet to the defaults. This worksheet does not accomplish that and only prompts the user to save a copy in the initial document page. If the user was to ignore this section, or at a later date, forget what was on this page, they may find themselves overwriting and deleting previous files that they wished to keep. Therefore the proposed design tool shall have a functionality to remind users either on save or make it impossible to save without creating a blank template copy.

However, whilst the sheet does have a list of problems, it does do the job of calculating the cost benefits - it's main let downs are the user interface which can be improved upon.

3.2 NIST Sprinkler Tool

The NIST sprinkler tool is different to the BRE one in the fact that the tool is a web based tool - this means that no other software other than a web connection and internet browser are required to use it. This is a different aspect than the options considered in this paper, which require software to be installed on the PC the user is using.


SPRINKLER USE DECISIONING

Sprinkler Use Decisioning is a Web-tool designed to facilitate economic analysis of residential fire sprinklers at the homeowner- and community-level. It uses the economic framework developed in NIST Interagency Report 7451 (*Benefit-Cost Analysis of Residential Fire Sprinklers*), which was used to measure the economic performance of a fire sprinkler system installed in a newly constructed, single-family house. Using the tool, the benefits and costs of sprinkler installation and use can be evaluated to determine whether sprinkler adoption is cost-effective for either an individual homeowner or for an entire community.

Inputs: Baseline Analysis | Inputs: Sensitivity Analysis | Results: Baseline Analysis | Results: Sensitivity Analysis

Costs

Sprinklered Area (ft ^{2})

Unit Installed Cost (\$/ft ^{2})

Annual Maintenance & Repair (\$)

Other Annual Costs

Study Parameters

Discount Rate (%)

Study Period (years)

Benefits

[✓ National] User Defined Zip Code

Input	Value	
Damages		
Value of a statistical life	0	Edit
Value of a statistical injury	0	Edit
Total value of structure and contents	0	Edit
Fire loss-to-value ratio	0.21	Edit
Fire Statistics		
Annual number of fires per 10,000 houses	35	Edit
Reduction in probability of fatality due to sprinkler use (%)	100	Edit
Number of fatalities per 10,000 house fires in unsprinklered houses	77	Edit
Reduction in probability of injury due to sprinkler use (%)	59	Edit
Number of injuries per 10,000 house fires in unsprinklered houses	390	Edit
Reduction in fire loss-to-value ratio (%)	78	Edit
Insurance Info		
Annual insurance premium	0	Edit
Reduction in insurance due to sprinkler (%)	0	Edit

Fig. 2. NIST Sprinkler Cost Tool

The ability to have instant access to the tool, regardless of the computer the consultant is using, makes this tool extremely flexible. It allows users of different operating systems and/or with access to a conventional piece of software installed on the machine to make use of the tool.

The tool is laid out well and it follows a natural progression - for example, the input boxes are related and it leads on from the previous question. Data is validated and you cannot progress to the next page if an item of information is missing or has been entered incorrectly (such as the addition of characters rather than digits). Wrongly entered sections are highlighted by red text until the correction has been made. Help for the user is provided by clicking a question mark icon on each page - however, this downloads a PDF document which then needs to be opened in a different program and then the help section for the application page then needs to be found. A more streamlined help system would be able to direct the user straight to the help section in the help file for the page the user is interested in. A javascript pop up or similar would have been able to provide the information *in situ* and thus require less swapping between programs. Again, this breaks rule 8 of Shneiderman's GUI design rules.

The tool allows the user to use the default settings that NIST have specified for the tool or they are able to put in their own settings. If the settings are changed, the user can easily return the settings to the defaults. This allows the user a greater control of the settings, whilst allowing allowing them to return the program to it's default state in case of error.

Whilst the online access may make the tool more viable for larger organisations (no need to do massive rollouts of IT software), it may also present a security risk. The tool does not contain any disclaimers saying that the data entered into the form will not be kept or used for any other purpose. For some firms, whilst no personal information is being uploaded, there may be issues regarding privacy because there are no disclaimers or even because the tool is a cloud application (cloud refers to the software not being present on the users machine but in the internet "cloud"). As NIST is American, the website is assumed to be hosted in America and thus falls under US data protection laws which differ from UK and EU rules - this again may cause concern, especially if the tool is updated and in the future asks for more details.

Finally, the last downside of the tool is that it does not allow for download of the data entered. To allow others to be able to replicate the results, it might be beneficial for the tool to allow users to export and import a plain text file with various settings the user might have customised. This would mean the tool does not have to be reprogrammed with data should custom values be required. It might have been possible to store data on this in a cookie (a small text file that most websites leave on your computer) but this does not occur so therefore a manual import/export system could have been implemented.

When viewed on a wide-screen monitor, it can be seen that the tool has been designed for a smaller, 4:3 ratio monitor - the sides of screen are taken up by large amounts of white space. This is something to bear in mind for designing of the tool.

4 Design Tool

The design tool, or more precisely, the decision support system, will allow the users to be able to determine the most cost effective method design and display these results to clients and architects. In the context of this research, this is "a computer program that is used by engineers to perform analysis of a building or its services (prior to realisation) for the purpose of making, modifying or evaluating design decisions" [8]. In the same paper, Lockley and Sun state that a design tool should be user friendly but the model behind the tool must be transparent to the end user, as well as making the program user friendly. This gives the end user confidence in the results generated by the tool as they can follow the process the tool has taken to reach its conclusions.

In regards to the user interface of the program, Shneiderman and Plaisant lay down 8 golden rules to interface design [15] that will be followed in the design of this tool. These are:-

1. **Strive for consistency**
   Program design should be consistent throughout, such as layout, fonts, design and where possible, user actions and terminology.
2. **Cater to universal usability**
   Recognise the needs of the users and design accordingly. Different users will use the tool differently (further discussed in a paper by Udema [18] which stated that different skill levels of users would use the tool differently, depending on the users level of experience in the field the tool is designed for). This means adding help for novice users and shortcut keys and faster pacing for more experienced users.
3. **Offer informative feedback**
   For each user action, the software should provide feedback though the feedback should follow the scale of actions (minor feedback for minor actions, major feedback for major actions)
4. **Design dialogues to yield closure**
   Sequences of actions should be organised into groups and should have a beginning, middle and end. Feedback should be given at the completion of a set of actions so the user knows the item is complete (for example, e-commerce sites show a checkout completion screen to let users know this set of actions has been completed)

5. **Prevent errors**
   Design the program to prevent errors, such as only allow numbers to be entered into a field that only needs numerical data entered. If an erroneous value is entered, provide feedback to the user and let them correct it. Allow them to only have to correct the erroneous value rather than redo the entire form.
6. **Permit easy reversal of actions**
   As much as possible, allow for easy reversal of errors. This allows users a sense of relief, knowing they can undo any error this allows for exploration of unfamiliar options.
7. **Support internal locus of control**
   Experienced operators desire the sense that they are in command of the interface and the interface should be designed accordingly. Surprising interface actions, tedious sequences of data entries or the inability to gather information will build dissatisfaction with the product.
8. **Reduce short term memory load**
   Human short term memory means that displays should be kept simple and short. Where appropriate, online access should be provided to command-syntax, abbreviations, shortcuts and other information.

4.1 Language

The tool will be programmed in either Microsoft Excel or in Visual Basic. Excel was chosen as the Microsoft Office suite is a common software within the corporate environment, with around 80 percent of companies using a version of it [9]. This allows the tool to be used by the maximum amount of people without purchasing extra software. Visual basic was also chosen as a potential language as it can provide very easy to use graphical user interface (GUI) programs quickly and simply. The programs created in Visual Basic can be downloaded and run with no additional software, allowing anyone running a version of Microsoft Windows to run the final program. The following show the pros and cons about each language:-

Microsoft Excel

✓ Easy to program basics
✓ Graphing functions built in
✓ Tabbed function built in due to worksheets
✓ Cross Platform - Microsoft Office runs on Mac OS X and Microsoft Windows so potentially the tool can be cross platform
✓ Easy import/export of data - just save the file
✗ Security concerns with macro functions in a corporate environment
✗ Restricted functions in Excel - however, are expandable with Visual Basic coding
✗ GUI limited - The graphical user interface is limited by layouts
✗ Potentially difficult to export imported data and results into a different format
✗ Change in screen resolution could potentially change layout

Visual Basic

✓ Doesn't require Microsoft Excel to be installed or present
✓ Fully customisable layout
✓ Easy to hide parts of the program that the user doesn't need to access
✓ Easy to set size of layout - should look the same on all systems
✓ Easier to keep open source so others can look at source code and continue work/check for security concerns
✗ More difficult to learn to program and very little prior knowledge
✗ Possibly difficult to graph
✗ Possible reliance on third party tools (gnuplot for example)

In discussion with the university computer society (a student society setup by those with an interest in computers, programming and other computer related activities), it was suggested that the author investigates the language C# (C Sharp) - the language is similar to Visual Basic and like Visual Basic, it can create GUI applications easily and quickly but is more powerful behind the scenes. However at the time of this paper, no research has been carried out this language by the author.

4.2 Screen Resolution

Screen resolution describes the screen size that a user will be using the tool on in terms of pixels. Figure 3 shows the worldwide screen resolution trends over the past 12 months. The data for this was taken from Statcounter, an online tracking service that collects information about web browsing via code on websites [2]. The code can track at what resolution a user is browsing the web - whilst this might be skewed towards the home users where web browsing may be more common, Statcounter provides this information freely and allows a reasonable picture of common screen resolutions to be built up so that web designers (and in this case, programmers) to design to suit the most common resolutions. Statcounter

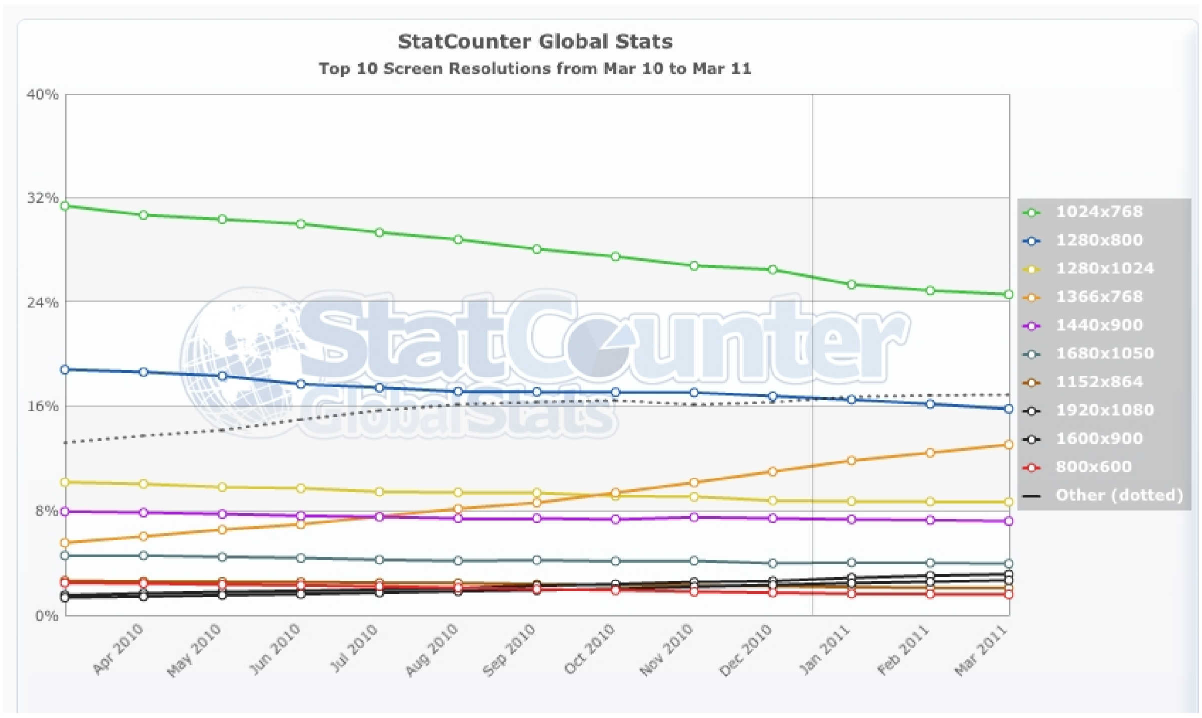


Fig. 3. Worldwide Screen Resolution From March 2010 Until March 2011

also allows for the screen resolution to be split further down into countries (this will be done by looking up IP addresses and finding the country the IP resides in). Due to the method of collection, these figure will be subject to different variables and may potentially be misleading due to people within the UK browsing through a proxy and thus not appear in the UK stats whilst they would appear in the worldwide stats, whilst conversely, people in non UK countries may be browsing through a proxy situated in the UK and thus will appear on the UK statistics instead of their country of residence. As can be seen from the UK specific results, there is a difference between the resolutions used in comparison to world wide though the spread between the resolutions used is lower, with the most popular resolution in the UK (1280x800) being used by less than the most popular one worldwide (1024x768).
The most common resolution in the UK are wide-screen formats - 1280x800 and 1366x768 - this could potentially be attributed to the large laptop market compared to the desktop market, as laptops tend to have widescreen format screens. However, this hypothesis can be challenged as news of laptops outselling desktops has been reported at least 3 different years with different figures [1,16,10] from different studies.

From this, it can be seen that the major trends show that 1366x768 laptops are beginning to increase in number (another reason is possibly the fact that Blu-Ray and High Definition TV is taking off) and that other resolutions seem to be in decline or of little relevance due to the low percentage share.

4.3 Design Methodology

Whilst designing and programming the design tool, it would potentially be beneficial to follow a design methodology to ensure that the program is completed in time and to a high degree of quality. Various

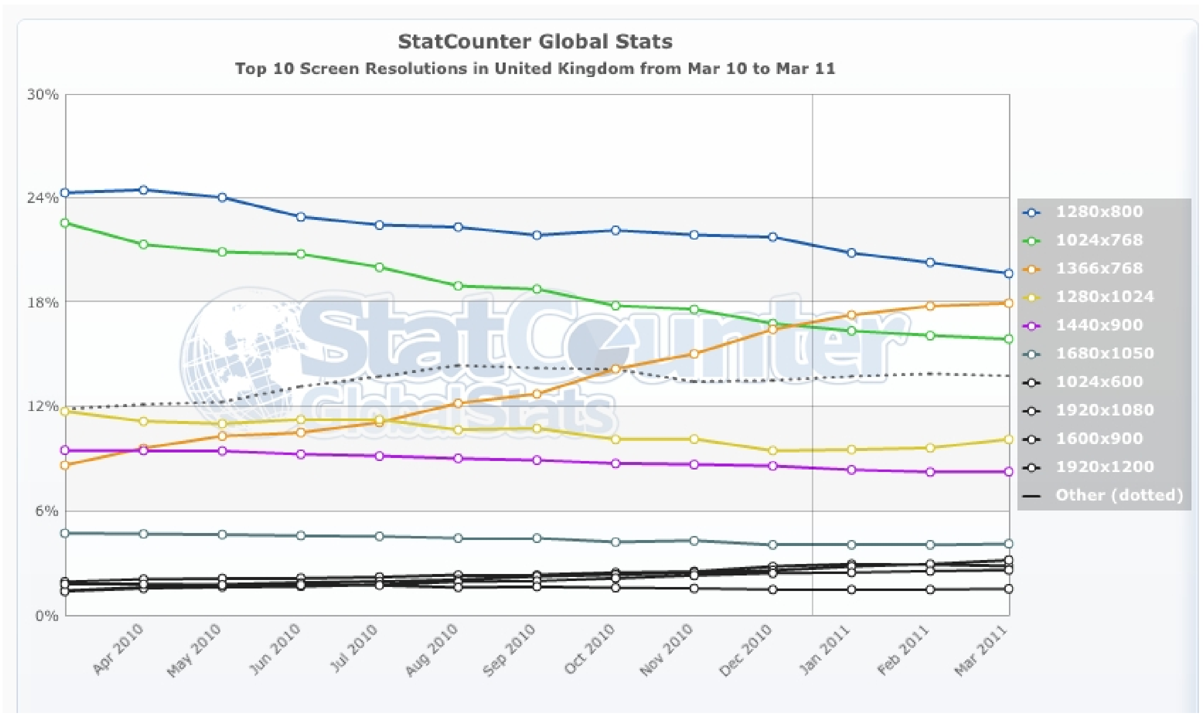


Fig. 4. United Kingdom Screen Resolution From March 2010 Until March 2011

methodologies are discussed below. Methodologies tend to fall into two groups, either waterfall (sequential) type or iterative/incremental methodologies [19]. In a sequential type strategy, all development steps are complete before the next step is done as shown in Figure 5. This means that all steps have

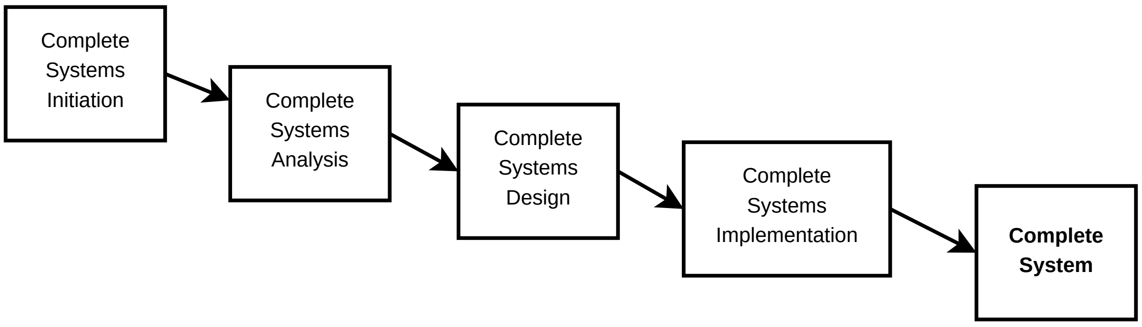

Complete Systems Initiation
Complete Systems Analysis
Complete Systems Design
Complete Systems Implementation
Complete System

Fig. 5. Sequential/Waterfall Methodology

to be fully completed and then the final product is release to the end user/customer. This methodology is a simple one but takes a longer time to see results over an iterative process [19]. Figure 6 shows an iterative process - this is the more common style of methodology used in software development today thanks to the ability to produce working prototypes and beta versions quicker than a sequential process. This allows the end user to test the software whilst it is still in development to offer any advice, make sure it fits the design brief and find any computer bugs present. Crucially, in industry, it allows for the customer so see progress being made on his/her contract.

RAD Rapid Application Development or RAD, is an iterative design methodology as it name implies for the rapid development of software. It focuses on getting the end users interacting with the program during the development so that each cycle or iteration can be tested. The principle behind the beta

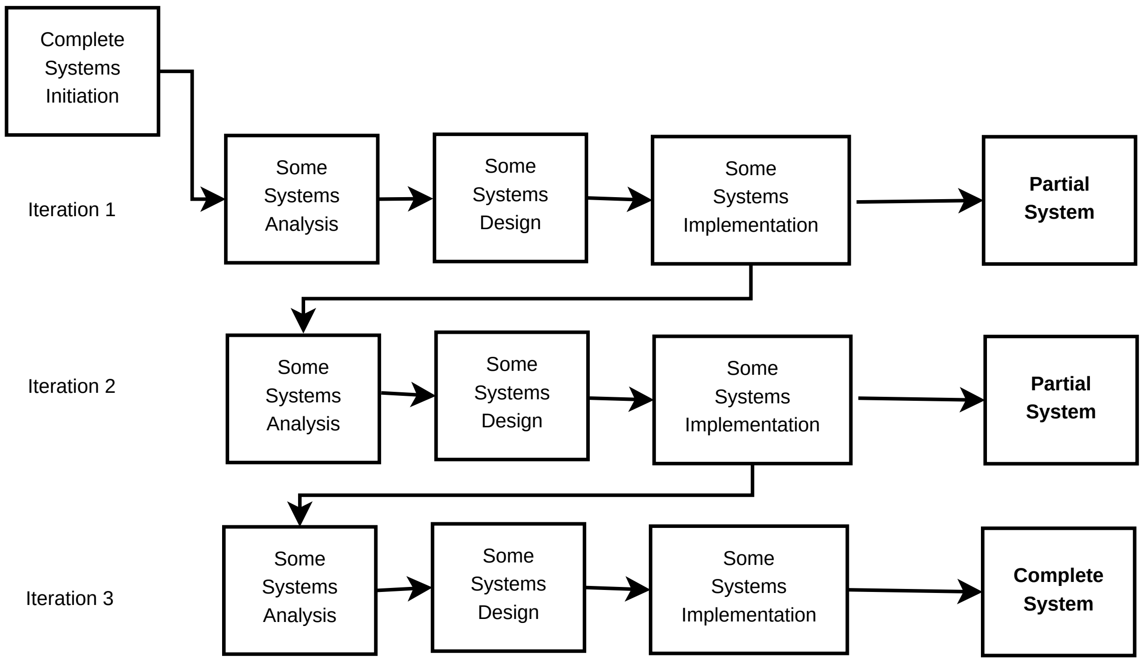


Fig. 6. Iterative/Incremental Methodology

versions and prototyping is that users will have a better idea of what they want when they see part of the program already working. RAD focuses on a few main points.

– Emphasis is on reducing time therefore phases of design and programming are consolidated and accelerated.
– In each iteration, only some design specifications will be considered.
– Assumption is that errors will be corrected in the next iteration.
– After each iteration, end users are invited to test the software.
– Based on feedback, designers will make changes until a version is deemed worthy of implementation.

The design phase within a RAD methodology is very short - the majority of design is done during each iteration.

Agile In 2002, a publication by VTT Publications stated that agile software development was an umbrella term for various different software methodologies [3]. All of them share the same principles of quick iterative design, similar to the RAD methodology described above. In 2001, an manifesto was setup to document what an Agile methodology should consist of. This can be found at the Agile manifesto page [5]. This states the following principles:

1. Individuals and interaction over process and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

These principles mean the various methodologies that fall under the Agile umbrella all share common roots.

Of the Agile methods, there are a few which stand out as being more common than other methodologies [6]. One of these is XP or Extreme Programming, put forward by Beck [4] and the other is Scrum, first described in 1986 [17].

XP Extreme programming is focussed on small team developments, with 2-10 people involved in the design team. XP seeks to prevent schedule slips by having short release cycles on software [4]. However, due to the nature of the methodology, of inter team communication and the need for co-location of the the design teams. It also has a very short iteration time of of about 2 to 4 weeks.

Scrum Scrum is another team driven methodology that relies on an iterative, incremental steps. Scrum relies on:

– Transparency - Ensures that all aspects that affect the outcome are visible to those managing the work.
– Inspection - Work must be inspected frequently.
– Adaption - If an inspection determines something is outside acceptable limits, the inspector will adjust the process.

This list was taken from [14]. The roles in a Scrum are those of the team (the programmers) and the ScrumMaster. The team are designed to optimise flexibility and productivity. Iterations are called sprints and these sprints are time boxed (done within a set time limit). During the sprint, the ScrumMaster makes sure that no changes are made that would affect the Sprint Goal. As the Sprints progress, the iterations come together to form the final product.

5 Conclusion

From the previous tools and the research conducted, a reasonable picture of the tool can be drawn up. A choice of programming languages has been proposed and put forward with both languages offering very good potential to create a a tool that builds upon those that have gone before it. The final choice of language is yet to be made however. By critically analysing the previous tools, the one proposed by the author can avoid the shortcomings of these other tools and build upon them.
By investigating methodologies for the development of the software, the software design and programming can be undertaken via a framework. The ones mentioned above, XP, Scrum, Agile and RAD all have benefits and weaknesses. However, RAD suits the style of development of this software and will be the framework to follow. This methodology was chosen over the others as both XP and Scrum focus mainly on team work and team projects and this project only has one programmer. However, the methodology chosen relies on the input of the end users, in this case, fire engineering consultants. Therefore the end users will have to be approached and asked to participate in the design and release cycles.
Finally, the analysis of the importance of the GUI and the screen resolutions has helped to inform the decision that the tool should be designed to accommodate wide-screen ratio monitors as these appear to be on the increase and therefore, following the 8 rules laid down by Shneiderman, specifically rule 8 to reduce short term memory load which scrolling would increase.

References

1. Technology Briefing: Hardware. Online (July 2003), `http://www.nytimes.com/2003/07/03/technology/03TBRF1.html?ex=1118030400`
2. Statcounter - Global Stats. Online (April 2011), `http://gs.statcounter.com/`
3. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile Software Development Methods. Tech. rep., VTT Finland (2002)
4. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional (2000)
5. Beck, K., Cockburn, A., Jeffries, R., Highsmith, J.: Agile manifesto. Online (2001), `http://www.agilemanifesto.org`
6. Cohen, D., Lindvall, M., Costa, P.: Agile Software Development. Tech. rep., Data and Analysis Center for Software (2003)
7. Fraser-Mitchell, J.: The Costs and Benefits of Sprinklers in Schools. In: Interflam 2010 Proceedings. pp. 1845 – 1856 (2010)
8. Lockley, S.R., Sun, M.: The Development of a Design Tool. Pattern Recognition 28(10), 1499 – 1506 (1995)
9. McLeish, S.: Enterprise Plans For Productivity Tools: Holding Out For Microsoft Office 2010. Online (June 2009), `www.forrester.com/go?docid=54702`
10. Murphy, D.: Notebook PCs Outsell Desktops, First Time Ever. Online (December 2008), `http://www.pcworld.com/article/155975/notebook_pcs_outsell_desktops_first_time_ever.html`
11. NIST: Sprinkler Use Decisioning. Online (April 2011), `http://ws680.nist.gov/firesprinkler/default.aspx`
12. Ramachandran, G.: Probability-Based Building Design for Fire Safety: Part 1. Fire Technology 31(3), 265–275 (August 1995), `http://dx.doi.org/10.1007/BF01039195`
13. Ramachandran, G.: The Economics Of Fire Protection. Taylor and Francis Group (1998)
14. Schwaber, K., Sutherland, J.: Scrum Guide. Tech. rep., Scrum.org (February 2010)

15. Shneiderman, B., Plaisant, C.: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Pearson Education, 4th edn. (2004)
16. Singer, M.: PC Milestone–Notebooks Outsell Desktops. Online (June 2005), `http://news.cnet.com/PC-milestone--notebooks-outsell-desktops/2100-1047_3-5731417.html`
17. Takeuchi, H., Nonaka, I.: The New New Product Development Game. Harvard Business Review 64(1), 137 – 146 (1986)
18. Uduma, L., Morrison, G.R.: How do Instructional Designers Use Automated Instructional Design Tool? Computers in Human Behavior 23, 536 – 553 (2007)
19. Whitten, J.L., Bentley, L.D., Dittman, K.C.: Systems Analysis and Design Methods. Irwin/McGraw-Hill, 6 edn. (2003)