Is there a role for Apps in the future of Safety Management?

R L Maguire CEng MIMechE MSaRS MBCS

RS2A Limited,Swindon, rlm@rs2a.com

Keywords: Mobile apps, Software, Safety.

Abstract

This paper seeks to describe mobile apps, their benefits and problems when they come to be used in the safety domain. This research compares and contrasts the development processes for a typical safety domain tool, against the development processes used for apps. It seeks to answer the question whether future safety practitioners will all be using hand-held, mobile devices with safety Apps on? The paper concludes with a set of guidelines, developed from these opening statements for prospective App developers and App users, such that potential Apps of the future might actually be useful and demonstrably safe to use in the future of safety management.

1 Introduction

All tools used in the development of safety relevant functions or services shall have sufficient safety assurance to ensure that they do not jeopardise the safety integrity of the function or service. Analysis shall define the safety assurance required of each tool with respect to its use on the system. This is a minor extension on the requirements for tool use in the UK MoD retired standard for requirements for safety related software in defence equipment [6]. But it could be from any software or safety standard.

"Writing and delivering an App can be quite a daunting task with many bases that need covering, including research, design, development, marketing, technical support, and more" – Rodney D. Cambridge (creator of 'Top-Ten' App 2008) [2]. So can the additional requirements of an App for the safety domain be satisfied as well? What are the requirements in the first place and is there a need or role for Apps in the future of safety management?

2 What Exactly is an App

The word App is of course short for 'Application', and like any computer programme, it is the app software that adds functionality to a mobile device [7]. Apps themselves are not that new. Google produced a successful suite of apps for online services in 2002; these included webmail, calendars, etc. However, the term has become somewhat hijacked by Apple and has become synonymous with games, maps, past-times and service ordering (e.g. taxi, pizza, flight times) that run on the multitude of mobile devices that exist now.

An app is a smartphone or tablet computer application that provides some functionality. As the shortened name implies, apps typically have less functionality that the more 'grown' up full applications that run on desktop or laptop personal computers [2]. So a small office (small 'o') app might allow you to create, view, edit and save MSWord documents, but it might not allow you to use macros, templates and track changes. It may not support all the fonts and languages that are usually available, and it may not allow mail merging, clip art or formatting for printing. The question to think about here is, 'Well, do I need all that anyway?'

So, Apps are usually just small scale, reduced functionality software programmes. There are a number of very good reasons for this as follows;

Screen size and resolution: Smartphones and tablets have much smaller screen size available than a full 21" high resolution desktop screen. This can make it difficult to appreciate and observe full graphic representations without frequent re-scaling and screen movements. Any table wishing to display multiple columns in high resolution could render the font size prohibitively small.

Demand Profile: The usage profile of an app is based on the app being something that is run and looked at for a very short time. It doesn't stay on all day constructing a technical paper for a safety conference (or did it?). They are launched for a short term objective – check the weather, order a taxi, look up a shop location. There are some exceptions – maps generally stay on for the length of the journey and some games, at least for older children and teenagers are designed to be played for hours – at night – after they've been sent to bed.

System Profile: Many i-things have a limited package volume in order to be more mobile. This reduces the space available for RAM and memory capacity. Desk top computers may have 8 or 16 GB of RAM, smart phones may only have 10% of this. Memory capacity is being satisfied by the use of cloud-based storage – although there are security and availability aspects to consider in these areas. Operating speeds on mobile devices are largely comparable with PC system at typically at or above 1GHz; so they can do very fast things – as long as they use a low amount of memory.

Development Profile: The development profiles are discussed in more detail in the next section of this paper. Essentially,

the app market is very dynamic and this week's fashionable app game, app service or app function, becomes next week's history. So the development profile for a commercial app has to be fast in calendar time – typically from idea to app store in less than six months. The amount of (quality) code development that can take place in this time is small. A bespoke App for a dedicated single or low number of safety functions does appear to be a viable development model, if the key aspect to the App is that it is mobile. Otherwise a full PC-based application would provide equal functionality.

Market Appetite: The market place drives app cost. There has been a noticeable 'race to the bottom' [2] for sell price of a commercial app. Many apps are totally free to download and use, usually through sponsorship or as part of a marketing or demonstration campaign. Most game apps retail for £1 or less. I downloaded 'Smart Office 2' for less than £10. This app allows for the opening, creation and wireless printing of many MS standards and pdf format. The likely number of sales and unit costs does reflect on the size and functionality of the app that is offered. But, if you only want an app to do just a few things anyway, the cost benefit analysis is easy.

One area that could influence the future development and price of Mobile-Apps is the strong open-source software communities within several safety relevant domains. For example "auvation" produce a freely available fault-tree analysis package 'OpenFTA'. This open source product has now been further developed by the producers into 'FormalFTA', as a commercial product, where its distribution is now being considered.

3 Apps in the Safety Domains

There are already a number of apps on the market that are safety relevant. Of course, if you don't have a mobile smart device or i-thing, you will not be exposed to their existence. There are several review publications for mobile apps, usually dedicated to particular equipment e.g. ipad or Android or Windows-touch etc. These publications typically review and list 400 or 500 current apps across a wide range of domains, for example, Gaming, Music Streaming, TV & Film, Immersive stories, Reference Libraries, Outdoor mapping and routing, Email, Task Management, Shopping, Document Management [8] and some/few bespoke industry apps. It is this final category that includes the safety relevant ones. As you might expect, they are not terribly popular so generally never make into the top 10 of anything.

App name and price	Safety relevant functions of the app
NHS ANT (free)	Provides updated guidelines to medical professionals on the dose and prescription guidance for the use of antibiotics to treat a range of 50 infections e.g. meningitis, pneumonia, clostridium difficile and MRSA.
Europe Warn (free)	Pushes warnings of severe weather in user selected regions of Europe. For use in real time vigilance warnings published by national meteorological service participants
HazMan3 ($2.00)	Allow the user to input floor plans of safety inspection areas and record hazard points using a standard set of markers. This can then form a report for demonstrating duty of care; producing a safety baseline; or constructing an accident report list.
HaZaP (£100.00)	Allows users to construct linked accident sequences including multiple intermediate events. Presents configuration controlled hazard and accident logs along with graphical, spread-sheet and pdf outputs

Table 1: Selection of safety relevant apps

In spite of their lack of popularity this paper presents a series of apps that are available now or shortly, that may be argued to have a relevance to safety. The discussion on each app shows how the author sees them as being safety relevant with regard to its functional use.

The Apps shown in Table 1 indicate that there is a safety relevance to many apps that are already in existence. It is by no means certain that there has been any risk assessment or evidence produced on the code development process or the testing regime. But does this mean that they are not fit for purpose? Or that they should be prohibited from use? Aspects of the app software development methodology discussed later in this paper will help to resolve these issues.

4 Review of Software Development Methodologies

In order to make any claim for safety integrity in any software, app or otherwise, it is necessary to identify a framework by which to judge that integrity. In full software applications this framework is provided by following recognised good practice based on a published, standard approach throughout the software development lifecycle.

The last half-century has seen a dizzying progression of technical advancement in the areas of computer, software, and communications technology. With each advance came rapid changes in the way society works and lives. The impact of technology is increasingly pervasive. Even as the current economic downturn limits capital investment, innovators and

entrepreneurs are pushing the limits in the areas of biotechnology and nanotechnology [1].

Several development methodologies have been published and used over time, but there is now a new approach that has become more popular in order to suit the market place for apps. These are discussed below, the author offers an apology if a particular favourite development methodology is not presented; the production of a complete list is not the intent of this part of the paper.

Code and Fix Model
This was the first basic methodology in the earliest days of software development, and it contained two basic steps – i) Write some code and ii) fix the problems. However, after even just a small number of fixes the code became so poorly constructed that future maintenance and modifications were prohibitively expensive [ibid.]

The Waterfall model was highly influential in the 1970s. Originally developed from the Stagewise Model of the 1960s, it provided for recognition of feedback loops between stages and allowed for an initial incorporation of prototyping in the software lifecycle via a build stage running on parallel with requirements analysis and design [ibid.]

The transform model assumes the existence of a capability to automatically 'transform' a formal specification of a software product into a programme satisfying the specification. This methodology by-passed the difficulty of having to modify code, since the modifications were made to the specification. This model does share some of the difficulties of earlier models, such as the assumption that the user's operational system will be flexible enough to support unplanned evolution paths. Additionally (as predicted in 1988) this model faces a formidable knowledge-based maintenance problem in dealing with the rapidly increasing and evolving supply of COTS software products (although the acronym COTS hadn't been coined at that time – the phrase at the time was "reusable software components and commercial software products"). [ibid.]

The spiral model evolved from experience with incremental refinements of the waterfall model as applied to large government software projects. The model reflects the underlying concept that each cycle involves a progression that addresses the same sequence of steps, for each portion of the product and for each of its levels of elaboration. The three primary areas of concern with this model involve matching the software to contract need, relying on risk-assessment expertise, and the need for further elaboration of spiral model steps [ibid.]

In spite of these models being available, many projects failed attempting to use the same techniques. Some projects got lost in the documents and never implemented any code, missing the window of opportunity for the software. Others did not leave enough time at the end for implementation and testing and delivered systems inconsistent with the documents and designs on which most of the project time was spent [5].

At the same time, numerous projects were very successful that did not follow methods with binders of documents, detailed designs, and project plans. Many experienced programmers were having great success without all these extra steps. The determining factor of project success seemed more and more to be the people on the project, not the technology or the methods that were being used [ibid].

This gave birth to a disciplined, yet lighter approach to software development, known as Agile Methodologies. Extreme Programming (XP) is the most widely used agile methodology. To many, XP is a set of 12 inter-dependent software development practices. Used together, these practices have had much success, initially with small teams, working on projects with high degrees of change. XP teams use a simple form of planning and tracking to decide what to do next and to predict when any desired feature set will be delivered. Focused on business value, the team produces the software in a series of small, fully integrated releases that pass all the tests that the Customer has defined.

A series of rules for Extreme Programming have been published via the extremeprogramming.org web community [9] covering the areas of Planning, Managing, Designing, Coding and Testing. The basis for the rules is somewhat flexible as they contain directions to 'Give the team a dedicated open work space' and that 'A stand-up meeting starts each day'; as well as more familiar types of coding rules about iteration planning and unit testing.

5 XP Development Values

The XP Values are Communication, Simplicity, Feedback, and Courage. The essence [of XP] truly is simple. Be together with your customer and fellow programmers, and talk to each other. Use simple design and programming practices, and simple methods of planning, tracking, and reporting. Test your program and your practices, using feedback to steer the project. Working together this way gives the team courage.
These values guide actions on the project. The practices leverage these values to remove complexity from the process [5].

This is how many apps are now produced. An XP project proceeds in iterations of typically a few weeks in length. Each iteration delivers fully developed and tested software that meets the most valuable small set of the full project's requirements. An app developed to this approach proceeds in a steady rhythm of delivering incrementally more functionality. The Customer determines at what point in time the app can be released and deployed.

The pace of change in the software development industry remains at high. People continue to push the boundaries of known techniques and practices in an effort to develop software as efficiently and effectively as possible. Extreme

Programming and Agile Software Methodologies have emerged as an alternative to comprehensive methods designed primarily for very large projects. Teams using XP are delivering software often and with very low defect rates [ibid.]

6 Benefits of developing apps

It is the author's belief that using these agile software development methodologies for small scale apps holds the potential for great improvements in safety relevant software. Many of the software tool concepts that we see around us in the safety industry could be implemented via small scale smart apps. The model I put forward now is one of a small core app carrying out, say, Event-tree Analysis or producing a Goal Structure Notation; coupled with a number of plug-in apps for additional functionality. This kind of model probably already exists but hasn't been explicitly identified or found yet.

The chief benefit of a Mobile-App is that it is mobile – you can use it where you can't take a desktop PC or where a full size laptop would present handling difficulties. The Hazman3 App from table 1 allows geo-location specific hazards to be recorded and tagged to locations as part of a mobile working party around a work site. The weather warning related Apps can be taken via a mobile device on a remote mission – providing signal coverage is sufficient. Whilst some laptops are available in 'ruggedized' versions, sometimes the operating environment is more rugged.

The benefits could be extraordinary. An app may be small enough in functional code size so that is may be tested exhaustively. Incremental functions available from open source areas can, and I stress can, become high integrity through their prior use, testability and through ownership of the code.

Bespoke apps can also be developed, indeed have been developed and are now being developed. They may not be commercially available from the i-store or from Amazon just yet, but they are multiplying in our domain. The benefits to users come from the closeness to the code developers, so that modifications and functionality can be developed, tested and in service within a few weeks. And, if that app doesn't do exactly what you want, you really can get it changed or commission / develop your own bespoke app. The NHSANT App from Table 1 can be updated very quickly when clinical advice changes, because there is a short and direct feedback route to the developer's from the Users, something that doesn't necessarily exist in full applications.

Although this section does say benefits, there is the darker side to apps that needs to be considered, and there are several bear-traps to avoid. These are taken from an Information Week Analytics survey of over 300 business technology professionals in May 2011 [3]. It gives the percentage of responders citing a top concern over the growing use of devices, apps and operating systems.

Security risks: 62%
Variety of devices and operating systems: 53%
Lack of user support by developers: 43%
Lack of centralized platform for managing devices: 39%
Cost of through life maintenance: 23%
Cost of management effort: 21%
Loss of control over processes: 20%

I imagine there was a similar survey on the role of business computers in the 1970s, but I haven't been able to find it yet!

Interestingly, none of the concerns mentioned were relating to low confidence in functionality or poor performance. These points, and the listed concerns, do need to be mitigated if the benefits of apps are to be realised in the safety industry – that is, if we want to have the benefits.

7 Good Practice for the Development of Safety Relevant Apps

In light of the concerns noted in industry, personal experience with software projects and as cited above the author has proposed a set of good practice requirements for the development and use of Mobil-Apps in the safety industry. Of course, an existing software standard may be followed to the proscriptive or goal based direction of that standard e.g. IEC61508 [4] or the recently published (and very recently FAA approved) DO178C [9], including all the cohort documents. Due to the development timelines of these standards, they have not been developed with Agile development methods acknowledged, so it will remain difficult for an agile-developed App to satisfy those standards in entirety. However, the reduced development, use and functional profile of Apps should equally allow for a proportionally reduced development requirement. Proposals for good practice for the development of safety-relevant Apps are as follows.

1. The App development shall follow any published, pre-defined development methodology.
2. The development methodology shall be made available for User or Auditor review.
3. The development process shall allow for the creation of an associated data pack.
4. The associated data pack shall be made available for User and Auditor review.
5. The associated data pack shall contain a description of all the functions that the app is required to do.
6. The associated data pack shall explain how the functions of the app have been implemented.
7. The associated data pack shall contain a description and the results of the testing done to demonstrate the functional performance of the app.
8. The associated data pack shall contain a history of the development process used for the app.
9. The associated data pack shall contain a record of the competence of the designer and developer.

10. The app shall come with a recommendation on its integrity, limits and fitness for purpose based on a risk assessment.

It is the last point that makes a cultural change in the use of software apps as opposed to full software applications. This would be akin to the use of a Certificate of Design, but perhaps would be named as a Certificate of Risk. Much like any equipment or service, once a risk assessment has been made, the safety case makes a recommendation for use or for the next development stage or for test flying or for limited release etc. It is up to the person holding the duty of care over the end users to accept that recommendation or not.

The Certificate of Risk shall contain the version description; the summary of the associated data pack items listed above; a summary of- and references to the risk assessment; and a statement of residual risk

One additional proposal for the associated data pack that would accompany Apps (but may also be equally applicable to full applications), is that the data pack should (and 'should' is chosen deliberately) not have more lines of information than there are lines of code in the App. This would force developers to be succinct in their evidence presentation, rather than bulking up evidence so that it 'looks' good. Responses on this point would be welcome from the community.

8. Is there a role for Apps in the future of Safety Management?

This paper has shown that there are identified benefits from the use of Mobile-Apps including their obvious mobility, but also their ability to be designed for targeted safety functions using rapid, agile development techniques. However, these benefits are not unique to the domain of safety management.

Open-source development approaches for the wider benefit of society may reduce the need to spend resources on App development – someone may already provide a suitable free version.

There are multiple concerns within the business domain in general concerning Mobil-App use and the associated infrastructure. However, these can be seen as concerns relating to generic Information Technology applications rather than being specific to just Apps themselves.

Ownership and developer feedback distance can be exploited to create bespoke Mobile-Apps that can be developed to meet specific needs, perhaps within programme management or wider-system progress cycles.

The requirement for a data pack of evidence concerning process and testing proofs can be significant within the safety critical domain. This may be commercially prohibitive in terms of finance and resource requirement. Within a safety-relevant domain, the data evidence obligation may be able to be reduced in proportion to the risk (pending contractual agreements).

In summary, this paper shows that there is a role for Apps in the safety management domain. However, that role is currently based on a relatively complex relationship between mobility, function (role), safety relevance and an agreement over the amount of evidence that the end User might require. For high-integrity, single-function Apps it may become possible to exhaustively test the software. For multi-function high integrity Apps, the evidence requirements may just as well justify a full application to be developed. For lower integrity single or multi-function Apps, the agile and lower-cost App development process may indeed provide a niche opening for their capability.

This potentially inverse relationship between integrity and functionality can be represented in equation (1) below, where F is 'App functionality' and I is 'App integrity';

$$F \propto 1/I \quad (1)$$

In this way a Mobile-App is likely to be able to do one function at high integrity or several functions at lower integrity.

The author believes that on balance, there is enough evidence to show that there is a role for some Apps in the future of safety management, but there will be limited opportunity at the higher end of the integrity spectrum.

References

[1] Boehm, B. W. "A Spiral Model of Software Development and Enhancement", TWR Defense Systems Group, Redondo Bech, California, USA, 0018-9162/88/0500-0061$01.00, IEEE, 1998

[2] Cambridge, R,D. "How not to write an App – A reality check for budding app developers", Amazon.co.uk Ltd, Marston Gate, London, 2011.

[3] Finneran, M. "Mobile App Development Needs a New Approach", Information Week Analytics, August 15, 2011.

[4] International Electrotechnical Commission (IEC) " Functional safety of electrical/electronic/programmable electronic safety-related systems", IEC-61508, Edition 2.0, 2010-04

[5] Lindstrom, L & Jeffries, R. "Extreme Programming and Agile Software Development Methodologies", pg41-52, Information Systems Management, Summer 2004

[6] Ministry of Defence. "Requirements for Safety Related Software in Defence Equipment", Defence Standard 00:55 Issue 2, August 1997.

[7] Osbourne-Walker, S. "What is an App?", pg6-7, The Stuff Guide to Apps 2013-2013, Haymarket Consumer Media, November, 2012

[8] Parsons, J. "iPhone, iPad and Android Apps magazine", Image Publishing, No. 34, June 2013

[9] RTCA/EUROCAE "Software Considerations in Airborne Systems and Equipment Certification", DO178C / Ed-12C, RTCA/EUROCAE, May 2012.

[10] Wells, D "The Rules of Extreme Programming", from website www.extremeprogramming.org/rules.html (retrieved 5th September 2013)