Evaluating Situational Applications Builders

Mark GREGORY
Department of Finance and Operations, ESC Rennes School of Business, 35065 Rennes Cedex, France;
mark.gregory@esc-rennes.fr

and

Dr. Mario NORBIS
Department of Management, Quinnipiac University, Hamden, Connecticut 06518, USA;
mario.norbis@quinnipiac.edu

ABSTRACT

This paper introduces a number of applications builders which can be used by non-technical users to build and deploy business information systems. The emphasis is on different ways to build applications which are deployable on the Web. The applications so created are frequently said to be hosted "in the cloud", that is, on Web servers which may not be under the direct control or ownership of the organisation which owns the information.

IBM has suggested a new name for this category of simple applications builders: "situational applications builders": see for example [1]

This paper reviews certain applications builders and puts forward methods to evaluate them.

Keywords: Situational applications, Web application builders, PaaS – Platform as a Service, SaaS – Software as a Service

1 WHY END-USERS MAY WANT TO BUILD AND DEPLOY SMALL-SCALE BUSINESS INFORMATION SYSTEMS

The ways in which a company can procure the business applications necessary for its operations include combinations of three basic approaches which are already widely accepted, and a fourth:

1. ***Bespoke*** (custom) development; this requires technical skills
2. Purchase and use of ***packaged*** applications
3. ***Systems integration*** – composing applications from different components found on the web; this requires technical skills
4. ***So-called "end user programming", which can be done by business professionals.***

There are many Information Systems requirements which are on a relatively small scale, and where reasonably-courageous individuals, sometimes referred to as "power users", set out to build their own small-scale Information Systems. They do this, rather than wait for a large-scale, no doubt better-engineered but much later solution produced by Information Systems professionals. Regan has written extensively on this phenomenon, sometimes known as ***"end-user programming"***; see for example [2]

Recap: Storing small amounts of structured data

The context of this research is end-users within medium and large enterprises. Its findings may also be applicable in small and very small enterprises.

We normally store data on computers:

- Either when we have many occurrences of a specific kind of record each with a well-defined structure, and we want to process specific records, or complete sets of records. This is sometimes referred to as structured data and it is on this that we concentrate in this paper.
- Or where we have complex, semi-structured information requirements at the personal or small-group level. This situation is addressed in a separate paper [3]

A widely accepted way of storing data, indeed, we may even refer to it as the "natural" way to store such data, is by means of ***two-dimensional tables***.

Many widely used office productivity programs or suites provide good facilities for storing two-dimensional tables. For example, when using Microsoft Office http://office.microsoft.com/, the most widely-used such suite, each program has specific strengths and weaknesses when it stores data in this way. Figure 1 summarises the approaches, and their comparative strengths and weaknesses.

Method	Advantages	Disadvantages
Word	Simple, well understood by people with weak computing skills	No formulae (or only very rudimentary ones)
	Excellent formatting options	Not a safe place to store critical or long-life data
	Powerful built-in outliner for structuring lists (e.g. of tasks)	
Excel	Some degree of structure – rows and columns	Poor support for queries – searching is slow and finding information again is imprecise
	Very powerful data manipulation using formulae	Size limits – 65535 rows (until Office 2007)
		No design methodology or coherence
		Not a safe place to store critical or long-life data
Access	Relational data model gives method and coherence	More difficult to learn
	Very powerful data structuring and querying	Requires thoughtful use and advance planning
	Safer critical or long-life data	Not as safe as MS SQL Server, etc.

Figure 1 Comparative strengths and weaknesses of data storage in two dimensional tables: Microsoft Office tools

Web-accessible end user computing

A new challenge has arisen in making information readily available within and outside the organisation. This is the increasing prevalence and expectancy that computerised applications will be available via the Web: whether internally, on an intranet, or externally on the Internet. What is novel is that applications built by what some suppliers refer to as "Web-savvy business users" (and not by information systems professionals) can also be deployed to be Web accessible. As [4] states:

"

The recent rise of grassroots computing among both professional programmers and knowledge workers has highlighted a different development approach (our comment: from SOA, service-oriented architecture, usually applicable to major corporate systems). In this approach, those who best understand the business problem at hand develop rapid solutions without the overhead and formality of traditional IT methods. The new breed of situational applications (SAs), often developed by amateur programmers in an iterative and collaborative way, shortens the traditional edit-compile-test-run development life cycle. SAs have the potential to solve immediate business challenges in a cost-effective way, capturing the part of IT that directly impacts end users and addressing the areas that were previously unaffordable or of lower priority.

"

Research questions addressed in this paper

- What Web-accessible applications builders (or situational applications builders) exist, and what are their characteristics?
- How can business users (practising professionals and students) evaluate the most appropriate Web-accessible applications builders (or situational applications builders) to employ in a given application context?

Background to this research: structure of this paper

To obtain an answer to these questions, it is first necessary to identify what types of computer application exist in this area (largely by Web-based secondary research), and to suggest ways in which they can be evaluated. The paper summarises the tools available. It goes on to suggest that not only are the applications themselves situational, or specific to the situation; so is the evaluation of which tool to use. Since this remains research in progress (the questions have not yet been answered well), it concludes with directions for further research.

Before considering how to make information available outside the organisation, we firstly recap on spreadsheets and databases before going on to discuss Web application builders, or situational applications builders, which concentrate on enabling the construction of online (Web-accessible) spreadsheets and databases.

2 SPREADSHEETS VERSUS DATABASES

Spreadsheets consist of an array of cells, each of which can store a value or a formula. A formula relates the value of the current cell to other cells which can be considered as exporting their value to be used in the formula. Spreadsheets are a very powerful combination of the nearest approach to widely available end-user computer programming so far invented; and ways of storing (more or less) structured data in which the relationship between items of data is imposed by the use of formulae.

Databases generally have a more limited remit which they fulfil with greater precision than do spreadsheets. The most widely accepted, implemented and used type of database is the so-called "relational" database. See [13] which suggests as an informal initial definition that

"

A relational system is one in which the data is perceived by the user as tables (and nothing but tables); and the operators at the user's disposal (e.g. for data retrieval) are operators that generate new tables from old. For example, there will be one operator to extract a subset of the rows of a table, and another to extract a subset of the columns – and of course a row subset and a column subset of a table can both be regarded as tables themselves. The reason such systems are called 'relational' is that the term 'relation' is essentially just a mathematical term for a table.

"

The role of spreadsheets in business

[14] documents the pervasiveness of spreadsheets, and confirms their value for processes such as one-off ad hoc reporting and the prototyping of requirements for what should subsequently be re-engineered into, or acquired ready made as, formal applications to support specific management processes.

Spreadsheet? Database? Or something new?

We might advise users that there are many ways of storing data, saying (as we do to our students):

"

Each program has specific strengths and weaknesses. You are advised to consider very carefully who it is that requires what information, who collects the data, what kind of transformation is required between the input data and the output information, and then to choose the right program – or programs. A good workman knows his or her tools, and chooses the appropriate tool for the job.

When you are manipulating data for yourself alone, or as part of a small team, or in a very small business, spreadsheets are likely to be more intuitive, initially more productive and easier to get started with. However, as the volumes of data, or the number of users, increase, databases become much the preferable option. It is often a sensible and viable option to prototype the requirements for a small business information system using a spreadsheet, and then, when the requirements are quite clear, to transfer the data storage element to a database.

"

3 MICROSOFT ACCESS AND SOME OF ITS LIMITATIONS

Relational databases remain the main way in which enterprises store their data. Although other mechanisms exist for storing business data (notably, content management systems linked with company websites and intranets), relational databases are critical to the integrity of corporate data and also permit very general questions to be asked and answered in a focussed and accurate way.

Microsoft Office Access is probably the most widely-used way to design, create and maintain a small-scale relational database. However, Microsoft does not recommend its deployment on the Web, even on an intranet.

Problems that can arise when using Access

1. A database exists to meet the needs of a community of users. A Microsoft Access database normally exists on a single client computer. If the computer is on a local area network (LAN), it is accessible to concurrent users on the local network, but not across the Web.

2. In order to deploy a simple electronic business application, it is necessary to integrate a database server with a Web server. This is technically complex.

4 DATABASES SUITABLE FOR END-USERS

We have compiled this list using as a cut-off criterion that it should be possible and straightforward for business students to learn the software mentioned. In some cases we have proof that this is the case from our own evaluation; in others, we are accepting the assertion of publishers.

Product	Website
Caspio	http://www.caspio.com/
cebase	http://www.cebase.com/
DabbleDB	http://dabbledb.com/
eUnify	http://www.eunify.net/
FileMaker Pro	http://www.filemaker.com/
Firebird	http://www.firebirdsql.org/
Force	http://www.salesforce.com/uk/platform/
Iron Speed Designer	http://www.ironspeed.com/
Kexi	http://www.kexi-project.org/
Longjump	http://longjump.com/
Microsoft Office Access	http://office.microsoft.com/en-gb/access/default.aspx
Microsoft SQL Server Express	http://msdn.microsoft.com/en-gb/express/aa718378.aspx
MyOwnDB	http://www.myowndb.com/
MySQL	http://www.mysql.com/
OpenOffice.org Base	http://www.openoffice.org/product/base.html
PerfectForms	http://www.perfectforms.com/
Qrimp	http://www.qrimp.com/index.html
QuickBase	http://quickbase.intuit.com/
Rollbase	http://www.rollbase.com/home/index.shtml
TeamDesk	http://www.teamdesk.net/
TrackVia	http://www.trackvia.com/
Webfuser (Inuvia Technologies)	http://www.webfuser.com/PortalSite.aspx
WebOffice	http://www.weboffice.com/web-database/index.html
Wolf Frameworks	http://www.wolfframeworks.com/
Zoho Creator	http://db.zoho.com/

5 PC-BASED DATABASES SUITABLE FOR END-USERS

Database	Main Characteristics	Needs for improvement
Microsoft Office Access	Mature technology easy to learn	Poor in multi-user applications, Not web-accessible
Firebird	Open source relational database	
MySQL	Powerful database	Need separation from front end tools
Microsoft SQL Server Express	Target for smaller-scale applications	
Kexi	Recent open source development	Not web-accessible
OpenOffice.org Base	New development based on HSQLDB	Not web-accessible
FileMaker Pro	Multi-platform, web-capable	Proprietary and expensive

6 TOOLS WHICH MAKE AN EXISTING DATABASE WEB-ACCESSIBLE

Tools that take data from a Microsoft Access database and make it available on the web include:

- Iron Speed Designer
- Webfuser

7 APPROACHES WHICH CREATE NEW WEB-ACCESSIBLE DATABASES

This approach is distinguished from the previous by the fact that a new database structure is created.
IBM has suggested a new name for this category of simple applications builders: "situational applications builders". [4] describes situational applications (SAs) in these terms:
"
The recent rise of grassroots computing among both professional programmers and knowledge workers has highlighted a development approach in which those who best understand the business problem at hand develop rapid solutions without the overhead and formality of traditional IT methods. The new breed of situational applications (SAs), often developed by amateur programmers in an iterative and collaborative way, shortens the traditional edit-compile-test-run development life cycle. SAs have the potential to solve immediate business challenges in a cost-effective way, capturing the part of IT that directly impacts end users and addressing the areas that were previously unaffordable or of lower priority.
"

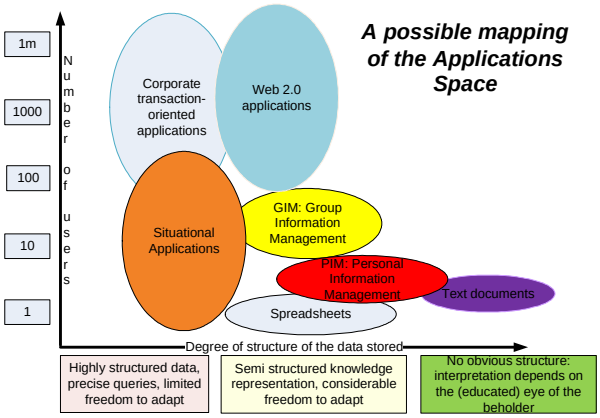
They go on to suggest that SAs have as characteristics that they are:

- Developed to address the situation at hand
- Often built by non-traditional, casual programmers with little up-front emphasis on reliability, scalability, maintainability, and availability
- Habitually use pre-existing software, often created and sourced from a third party via a public interface
- Are developed in short, iterative cycles measured in days or weeks rather than months or years, focusing on time-to-value
- Are usually information-centric
- Situational applications can be built as so-called ***"mashups"*** [5], [6]or as ***Web-based database applications***

Product	**Claims**
Caspio Bridge	True relational database on the web
Force	Fastest platform for building applications
Longjump	Robust Platform as a Service
PerfectForms	Enables anyone to create web forms. Easily integrated
Intuit QuickBase	Powerful
Qrimp	Easy and affordable
TeamDesk	Software as a service
Wolf	
Zoho	Tools in the web that can be used to build small applications quickly

8 EVALUATION AND CHOICE

We can summarise the positioning of situational applications in a mapping we suggest here for the "applications space":



Faced with such a richness of choice, how can business professionals or those who train and educate such professionals and/or students, choose the right software? No one tool of the many we have identified (and the many more that we have no doubt failed to identify) has achieved ubiquity, whether measured in terms of the extent of its use or the generality of its application. Further, there is little consensus about the evaluation of information technology or information systems [7] and comparatively few efforts have been made in this direction. Models have been developed for Information Systems evaluation (see for example [8]) in an attempt to specify necessary organizational process that reinforce the importance of evaluation and minimize Information Systems failure; but so far Information Systems evaluation remains underdeveloped and under managed: [9].

Two suggested dimensions of evaluation

One is to make a point-by-point comparison of the functionality offered by potentially-appropriate solutions, weighting the significance of the various functionalities in accordance with their importance to the proposed application. We identify this dimension as the ***"Weighted Functionality Matrix***". This gives at least an air of greater objectivity, subject to the obvious provisos that the weightings are themselves inevitably somewhat subjective, and the immense practical difficulty of determining whether in fact a particular package offers a given functionality and to what extent. In practice, such evaluation will often require extensive trialling, which may be ruled out on the grounds of software availability, time and cost.

A second dimension consists in attempting to identify broader and perhaps also subjective characteristics of potential solutions. Some at least of these characteristics may in fact have a significance which endures over time and which transfers between evaluation domains. We identify this dimension as the ***"Overall Characteristics".***

The table which follows suggests such potential characteristics. Clearly, the choice of these characteristics is subjective and interpretational.

Table 1: Overall characteristics of software solutions (a partial list)			
Characteristic	Meaning	Suggested significance for evaluation	References
Separation of concerns	The cardinal necessity of separating different dimensions of a problem and its solution. Examples include: ☐ Separating why from what from how ☐ Dijkstra's view that you should never do more than one thing at a time	Appropriate separation of concerns during the design and conception of a software product remains visible in the completed artefact and its documentation. The necessity for complex and confused documentation will suggest that the principle has not been observed.	[10], [11]
Orthogonality	Originally, orthogonality is a system design property facilitating feasibility and compactness of complex designs. Orthogonality guarantees that modifying the technical effect produced by a component of a system neither creates nor propagates side effects to other components of the system. The emergent behaviour of a system consisting of components should be controlled strictly by formal definitions of its logic and not by side effects resulting from poor integration, i.e. non-orthogonal design of modules and interfaces.	The absence of Orthogonality in the design and conception of a software product remains visible in the completed implementation. When the designer of the product has a clear idea of what he is attempting to achieve, with a clear and coherent object model, the eventual software will be easier to explain to potential users, quicker for them to learn, and as a result the software they produce will be more rapidly and more effectively applicable.	See for example [12]
Methodological coherence	Identifying a tool does not guarantee that its use will be effective. The application of a hammer to a screw gives poor results! It is important to choose appropriate tools, method and methodology.	The effectiveness of tools cannot be abstracted from a clear understanding by their designers of their context of use. Therefore an explicit evaluation criterion should be the methodological clarity and explicit recognition of the importance of the process of use which is demonstrated by the originators of a software product.	

Weighted Functionality Matrix and its application

The approach we have identified above as the weighted functionality matrix is hardly novel. Some such approach is frequently employed in the evaluation of supplier offerings by would-be purchasers.

A practical difficulty which arises is to identify a list of potentially significant functionalities, and then to obtain the necessary data from the suppliers.

Table 2 shows a deliberately greatly-abridged and slightly-modified extract from a table published by a consultancy company experienced in offering PaaS services. The full original was found via http://www.powerinthecloud.com/. The authors offer to make available on request their developed matrix.

Table 2 Situational Applications functionalities (***extract***)	Caspio	Force	Long-jump	Perfect Forms	Quick-Base	Qrimp	Wolf
Target Users							
Business Users/Analysts/Consultants	Y	Y	Y	Y	Y	Y	Y
Programmers	Y	Y	Y	Y	Y	Y	Y
Database							
Database Definition							
Relationships are handled without having to specify keys		Y	Y	Y	Y	Y	Y
Simple validation rules easily specified	Y	Y	Y	Y	Y	Y	Y
Custom validation rules without coding or scripting	Y		Y	Y			Y

Local master-detail (i.e. detail table embedded in master)	Y	Y	Y	Y	Y	?	Y
Master-detail relationship - 1 level (automatic support)		Y	Y	Y	Y	?	Y
Master-detail relationships - multi-level (automatic support)	Y		Y	Y	Y	?	Y
Automatic master-detail CRUD operations (e.g. cascading deletes)		P	Y			?	Y
Roll-up fields (automatic totalling of data from related detail records)	P	P	Y	Y	Y	P	Y
Default values can be specified conditionally			Y	Y			Y
Relationships are inferred automatically from imported data						Y	
Many-to-many relationships automatically resolved with link table						Y	
Supports hierarchy within a single table						Y	

Example application: an evaluation of the suitability of database software for Information Systems teaching

The authors have applied the suggested Weighted Functionality Matrix to some of the products mentioned earlier. We emphasise that the weightings are our own and are entirely specific to a particular evaluation. In no way do they reflect any overall evaluation of the power or usefulness of the products mentioned.

We carried out a quasi quantitative evaluation of the various possible software offerings, the findings of which appear in the subsequent tables. The evaluation was based on a qualitative assessment of the strength of a given package against the criterion, which was converted into a numerical form by using the first table, Table 3. The second table, Table 4, gives the result of the evaluation, which is certainly subjective, but perhaps slightly less so than a purely textual description.

Table 3: Qualitative and quantitative weightings		
F	0	Does not meet the criterion at all
E	1	Meets the criterion minimally
D	2	Meets the criterion adequately
C	4	Meets the criterion well
B	8	Meets the criterion very well
A	10	Meets the criterion excellently
G	0	Has not yet been assessed against this criterion

Table 4 Competing database management software	Licence	Free for student use	Ease of learning	Multi-platform: Windows, Linux, Mac	Ease of creating web-accessible data pages	Well-known - adds to a CV	Program-mability	Stable and mature	TOTAL
Weighting of this criterion		100%	200%	120%	100%	50%	50%	50%	
Firebird	Open source	A	C	B	E	D	C	D	32,6
MySQL	Open source	B	D	B	D	B	B	B	35,6
Microsoft SQL Server Express	Proprietary	A	C	E	B	B	C	B	37,2
Kexi	Open source	A	B	B	E	E	E	E	38,1
Microsoft Access	Proprietary	B	A	E	C	A	C	B	44,2
OpenOffice.org Base	Open source	A	B	A	D	B	B	D	49,0
FileMaker Pro	Proprietary	E	A	B	A	C	C	A	49,6

Overall characteristics of software solutions

We have suggested above that there may well be very significant evaluation criteria which do not correspond to specific functionalities, such as ***Architectural or Design coherence*** and ***Methodological coherence***. These are very difficult to identify and describe, let alone use as evaluation criteria. We would also caution that an excessive concern for design purity is often associated with tools and approaches which appeal to academics and which fail totally in the much more pragmatic marketplace. Nevertheless, we feel that they may be very significant in the levels of productivity achieved with tools which meet those criteria either very well or not so well. We intend further investigating this aspect empirically and experimentally.

9 CONCLUSIONS

We have reviewed PC-based databases and the emergence of Web-accessible situational applications. We have suggested the need for systematic evaluation of these new approaches and concluded that just as the applications created are situational, so too is the evaluation – in part. These interim conclusions also set an agenda for further research, both into the effectiveness of the applications discussed and into new methods for evaluating them.

Concerning our research questions, we can draw interim conclusions:

- We have presented a non-exhaustive list of Web-accessible applications builders (or situational applications builders) which has concentrated on commercially-available tools. We have here presented a small subset of the functionalities we have identified and against which our ongoing research is continuing to evaluate these tools.
- We have suggested how business users can evaluate the most appropriate Web-accessible applications builders (or situational applications builders) to employ in a given application context.
- We have shown how we evaluate functionalities using a Weighted Functionality Matrix, and we have hinted at the difficulty involved in obtaining the necessary data from tool suppliers.
- We have suggested significant Overall Characteristics of applications.
- In subsequent research, we shall refine the list of such characteristics as we experiment in the use of specific applications builders in specific situations.

10 REFERENCES

1	Cherbakov, L. & A. Bravery & B. D. Goodman & A. Pandya & J. Baggett (2007) 'Changing the corporate IT development model: Tapping the power of grassroots computing' IBM Systems Journal Volume 46, Number 4, 2007
2	Regan, E.A., O'Connor, B.N. (2002) Chapter 5 Knowledge Management, End-User Information Systems: Implementing Individual and Work Group Technologies. Upper Saddle River, NJ: Prentice Hall, 159-203
3	Gregory, Mark & Mario Norbis (2009a) 'Auditing Personal Information Management'. Paper presented at the UKAIS 2009 conference, St. Anne's College, Oxford in April 2009.
4	Cherbakov, L. & A. Bravery & A. Pandya (2007) 'SOA meets situational applications, Part 1: Changing computing in the enterprise' Found at http://www.ibm.com/developerworks/webservices/library/ws-soa-situational1/
5	Garrett, Jesse James (2005). "Ajax: A New Approach to Web Applications". Found at http://www.AdaptivePath.com accessed 25/06/2009
6	Gruber, Tom (2007) 'Ontology of Folksonomy: A Mash-Up of Apples and Oranges.' International Journal on Semantic Web and Information, 2007.
7	Beynon-Davies, P. & Owens, I. (2004) 'Information Systems Evaluation and the Information Systems Development Process.', The Journal of Enterprise Information Management, Vol. 17, No. 4. 2004, pp. 276-282
8	Teevan, Jaime & William Jones & Benjamin B. Bederson (2006) Personal information management: Introduction. Communications of the ACM Volume 49, Number 1 (2006), Pages 40-43
9	Love, P. & Ghoneim, A. (2004) 'Information technology evaluation: classifying indirect costs using the structured case method.', The Journal of Enterprise Information Management. Vol. 17, No. 4. 2004, pp. 312-325.
10	Rzevski, G. (1981) 'On the design of a design methodology,' in Design Science Method, R. Jacques and. I. A. Powell, Eds. Guildford, UK: Westbury House, 1981.
11	Dijkstra, Edsger W. (1982) 'On the role of scientific thought.' In: Selected writings on computing: a personal perspective. Springer Verlag, 1982.
12	Warboys, Brian (1980) 'VME/B: A model for the realisation of a total system concept'. ICL Technical Journal November 1980.
13	Date, Chris J. (2003) An introduction to database systems 8ed. Addison-Wesley
14	Ventana Research (2007) 'Requirements for 21st Century Spreadsheets: Uses and misuses of a critical business technology: Executive Summary.' San Mateo,CA: Ventana Research, 2007. Found at http://www.ventanaresearch.com/uploadedFiles/Ventana_Research_Requirements_for_21st_Century_Spreadsheets_Executive_Summary_FINAL.pdf accessed 11-07-2008.