AGILE SOFTWARE DEVELOPMENT APPLICATIONS TO ERP: DISCUSSION AND ILLUSTRATION USING DYNAMICS AX

Bojan Jovicic
IT Manager, DELTA SPORT
Milentija Popovica 7v, 11070 Novi Beograd, Srbija
+381 11 201 2840 or bojan.jovicic@deltasport.com

Todd Schultz
Hull College of Business, Augusta State University
Augusta, GA 30904
+1 706 667 4534 or tschultz@aug.edu

Dragan Djuric
School of Computing and Information Systems, Athabasca University
Athabasca, AB T9S 3A3 Canada
+1 604 569 8515 or
dragang@athabascau.ca

Abstract. This paper analyzes the technical aspects of ERP systems and their connection with agile methodologies, a crucial combination especially in times of economic crisis. ERP system deployments demand resolution to thousands of business process questions and the supporting databases and code base usually have huge numbers of tables and related classes. The complexity is clearly a challenge to manage especially when applying customizations and configurations. We describe how an ERP system's technical components can be classified from several viewpoints to identify the ones most worthy of investing our time in improving. An upgrade can be worth as much as one-thirt of the initial investment and this agile approach has helped us achieve a productive ERP system development environment, and it has enabled us to perform an ERP system upgrades effectively sometimes skipping an entire version. This provides that an effective connection between ERP deployemnt and agile software can be made to work.

INTRODUCTION

Agile methodologies are firmly established in software development but there is very little work on applying agile Enterprise Resource Planning (ERP) systems deployments and upgrades. ERP systems connect most of the departments of an enterprise, supporting the information flow inside the organization, and also connecting it with external partners. As such, these systems can be extremely large and complex.

In its Sure Step methodology and supporting assets, Microsoft provides for an agile project type. It includes tools to help manage the agile collection and articulation of business processes and "promotes a collaborative process between the resources that own and specify the requirements for the solution and the resources responsible for the development and rollout of the solution." (Shankar, 2011) The agile project type was introduced in the Sure Step 2010 release to facilitate development and rollout of of solutions to customers expecting to use Microsoft Dynamics as a platform and "customize the solution to their specific needs". This is an excellent tool for high-level managemnt of the processes but our contribution supplements such a methodology with identification of specific modules for customization.

Our main results are illustration of opportunities for ERP systems upgrades to take advantage of agile development technique to provide a more methodological approach to ERP systems deployment and maintenance. This paper will illustrate a technical perspective for analysis of ERP systems and connect the data from the analysis with future requirements for development and maintenance from an agile perspective.

Before the current economic crisis companies were – perhaps – more ready to invest in technology to enable growth. Now there is more tendency to delay or very carefully enter new ERP projects. Even with identified need for an ERP system upgrade or change, companies are focused on maintaining current implementation. Using agile with ERP helps a company to easily maintain the current ERP implementation but support new requests and remain ready for an upgrade. It has even enabled doing an upgrade to second next major version of ERP (from version 3 to version 5) skipping one whole version, which is considered a very worthy move in ERP world. Section 2 introduces us to ERP systems and outlines the current state of the art in development, maintenance and upgrade of ERP systems. It also explains basic agile concepts. In section 3 quantitative analysis of the software engineering state of an ERP system is performed from different aspects and each of

them is discussed. Section 4 connects this analysis with agile approach and explains the results that have been achieved using this approach. In the final section a summary of paper is presented.

PREVIOUS WORK

Agile software development has been growing steadily as methodology of choice for many organizations (Cockburn, Agile Software Development: The Cooperative Game (Agile Software Development Series), 2006). It comes in various forms including Extreme Programming, Scrum, Crystal and among others.

Agile is considered as very effective approach in comparison to other more rigorous methodologies in terms of business performance, customer satisfaction and quality (Cockburn & Highsmith, Agile software development, the people factor, 2001). In same paper (Cockburn & Highsmith, Agile software development, the people factor, 2001) the authors state that agile excels in exploratory problem domains (extreme, complex, high-change projects), and operates best in a people-centered, collaborative, organizational culture. However, it is not for all domains: imposing agile principles on process-centric, non-collaborative, optimizing organizations is likely to fail.

Enterprise Resource Planning (ERP) systems are information systems which out-of-the-box cover most of the business needs of a typical enterprise. Since these systems originated from the manufacturing field (Wallace & Kremzar, 2001), this is usually this business area that is supported most. For some other areas there might be a need to customize the ERP system by modifying the existing functionalities or creating completely new ones (Markus & Tanis, 2000).

There are only a few publications that tackle the issue of using agile with ERP systems. Reasons for this might be found in issues of agile application domain specification from (Cockburn & Highsmith, Agile software development, the people factor, 2001). There is also a confusion in research papers which try to connect agile and ERP. Most of them which address agile and ERP are referring to the agile property of ERP system in some context (e.g. using business rules to make the ERP system itself more agile in adapting to changes within the business) rather than trying to connect agile software development with ERP and show the challenges and possiblities. There are a significant number of papers highlighting ERP systems application in agile manufacturing as well.

Some of the ERP systems that have the largest market share are shown and compared in Figure Figure 1 which diagrams the systems on the two dimensions of completeness of vision and ability to execute. Another stratifications are to compare them by how they support the key functional areas or an alternate comparison by the technical aspects of the ease of maintaining ERP systems (Jovicic & Vlajic, 2009).

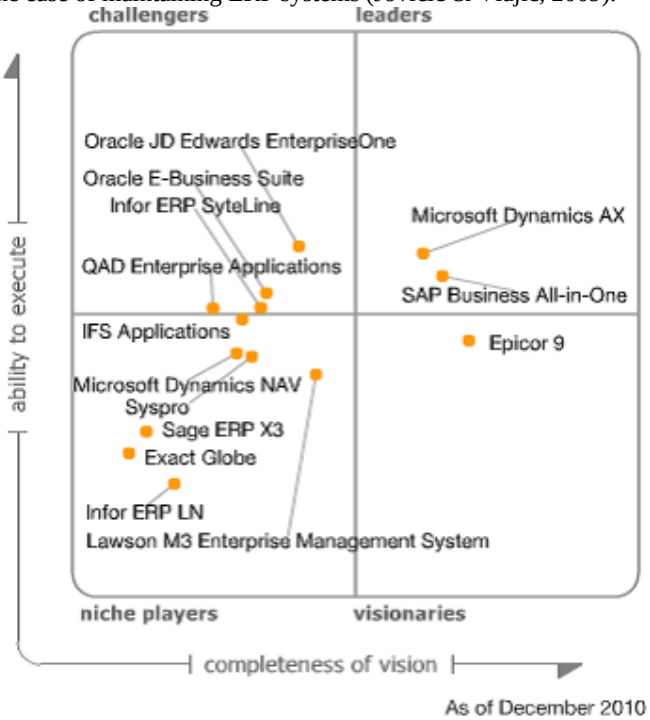

Figure 1: Gartner's Magic Quadrant for ERP for Product-Centric Midmarket Companies as of December 2010 (Magic Quadrant for ERP for Product-Centric Midmarket Companies, 2010)

As a growing number of companies adopt ERP systems, ERP systems implementation and upgrades are identified as one of the top five IT priorities among global CIOs according to independent surveys conducted by Morgan Stanley (Togut & Bloomberg, 2003) and Deloitte & Touche/IDG (Achieving, Measuring, and Communicating IT Value, 2002).

Much of the pressure to add new modules to their ERP systems seems to arise about six to 12 months after an organization has gone live with its initial ERP implementation (Beatty & Williams, 2006). In that analysis, the authors listed some of best practices for ERP upgrades. The finding that was listed as a big influence is the focus to un-customize the customizations and to focus on the usage of the out-of-the-box functionalities. As it has been indicated (Dittrich, Vaucouleur, & Giff, 2009) this is not always possible (e.g., a heavily customized ERP systems resulting from a company working in country or business area that has not been localized yet from process side).

One of the principles of agile software development (Fowler & Highsmith, 2001) is "Responding to change over following a plan". In order to adapt this principles we must plan to change. All agile methodologies have built-in processes to change their plans at regular intervals based on feedback from the customer or customer proxy. Their plans are designed to always deliver the highest business value first. One concrete agile artifact that is relevant for this paper is product backlog from Scrum (Schwaber K. , 2004). It basically shows the ordered list of requirements for the system or product being developed by business value for the company (Schwaber K. , 2004). It is also adapted for change and evolves as the product and the environment in which it will be used evolves (management constantly changes it to identify what the product needs to be appropriate, competitive, and useful).

METHODOLOGY

The ERP system that we have analyzed in this paper (Dynamics AX, shown in Figure Figure 1) has solid technical support for some of agile principles. We will first show some functionalities which are easily accessibility and configurable.

For Continuous Integration (CI) it supports Team Foundation Server, but also out-of-the-box Version Control System (VCS) which we will analyze in agile context. This VCS has some form of code check-in policies. It can reject code which has compiler errors or warnings, the code which contains TODO items and the code which contains best practice violations. In SAP ERP, by contrast, VCS issues proved to be very bad for agile (Meszaros & Aston, 2007).

Best practices are a form of static code analysis, which is performed without executing the programs (in most cases the analysis is performed on source code). Simply by focusing on correcting defects earlier (during the creation of code) rather than later in a project, one can cut development costs and schedules by factors of two or more (McConnell, 2004). There are about 350 rules on the cumulative development experience of this ERP system, which in turn are based on general good software development guidelines.

It also supports test cases which are written in its proprietary language, but are very ERP specific (support for different isolation levels like running tests in newly created business entity only for duration of the test). This further enhances CI support because VCS system can be configured to run a test project after every check-in and reject the code if it fails the tests.

A real wealth of useful information can be found in ERP's meta-model and in its runtime. One of the reasons this ERP uses meta-model is so that it can work with both Oracle and Sql Server . Besides this it contains model element definition of the whole ERP system. We have developed tools that harness and combine this information with other sources and have identified following aspects that help support agile software development for ERP systems:

1. Number of model elements by layer and type
2. Number of model elements by modules/business areas and layer
3. Usage frequency of elements by layer

In the following sections we give a more detailed view of these aspects through a detailed analysis of each of this aspects with the sample data shown for a retail/wholesales company operating in several countries.

Number of model elements by layer and type

Most of the existing research (e.g. (Beatty & Williams, 2006) (Nah & Delgado, 2006)) suggests the importance of avoiding customizations as much as possible. Our goal is to identify where the customizations are and to see how we can plan for change.

In order to do so, an aspect which focuses on identifying model elements that have the most customizations is introduced. Each element of Dynamics AX meta-model has a property called layer. This property indicates if the element is created by the vendor (Microsoft), by a partner or by then end-user company. If an end-user company's developer makes a modification to an element that is made by vendor (SYS layer) or by partner (e.g.

BUS and VAR layers) its definition is copied to end-user layer (e.g. USR and USP layer). This end-user layer is the top layer. The partner modifications are in the middle layers, and vendor layers are at the bottom. Based of layer location of specific model element, we can see if the element is vendor original or partner or end-user.

The meta-model contains several different model element types (tables, classes, class methods, menus, etc.). Their distribution over different application layers can be analyzed in order to see layer location of customizations.

Figure **Error! Reference source not found.** shows different model element types and their distribution over layers. The number of elements for each type is shown as vertical axis. Types with most elements in total are shown as the ones on top of Figure **Error! Reference source not found.** (i.e. Class methods). The layers are shown as horizontal axis and have been grouped and sorted in hierarchical order, as used by the application layer system mentioned before. For brevity, some element types which do not have customizations are not shown.

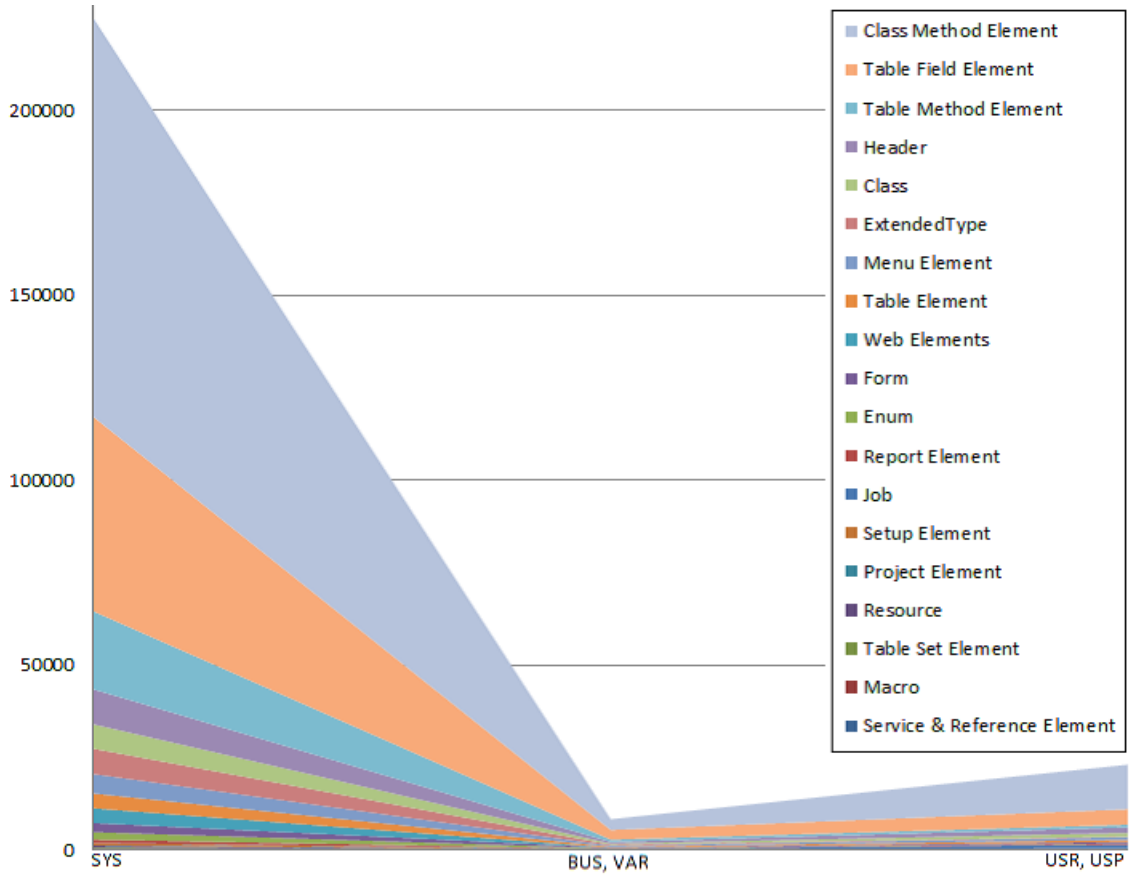

Figure 2: Element Type over Layer distribution

From Figure **Error! Reference source not found.** one can see that in sample data shown most of the customizations are located in the end-user layers (USR, USP). There are a number of customizations done by partners located at BUS and VAR layers. The most common customization method at end-user layers (USR, USP) was the modification of class methods, which is a way to customize the processes. This kind of customization is more widely present at end-user layers than at partner layers in respect to element types. Vendor (SYS) layer shows wide usage class methods, which is almost equally common as in end-user layers. The second most important customization was the creation of modification of table fields and it was equally done at end-user and partner layers. From this two element types and layer location aspects it can be concluded that in this sample the need for customizations was mostly process oriented, while existing data model didn't need as many modifications in the code.

Number of model elements by modules/business areas and layer

A valuable insight is provided in form of the aspect that identifies business area that a group of model elements belongs to and makes a larger functional group. This is also called a module. This aspect is important as a general guideline about the most developed areas of ERP systems.

Table 1 shows the distribution of tables by business areas (rows) and application layers (columns).

Table 1: Business areas and layers

Business area	SYS	BUS	VAR	USR	USP	Total
	420	30	46	67	258	821
Basic	357	4	1	3	10	375
Ledger	301	2	3	6	7	319
Inventory	217		7	9	13	246
HRM	208	15			1	224
Admin	153				4	157
Customer	121		10	7	13	151
Vendor	103		8	9	7	127

In the first row, there are the tables that don't have relevant attributes that determine business area properly set up. A company can try to remedy this on appropriate application layers where it has access (e.g. end-users at USR/USP layers). Most of these tables are in vendor (SYS) layer. Second row contains tables which are in basic area which supports all other areas.

The third row contains tables related to ledger (general ledger) area, which form a core of financial transactions module. Fourth row contains inventory related tables which form core of inventory management or Supply Chain Management (SCM) module. Next row contains tables related to Human Resource Management (HRM).

One very valuable insight comes from noting the ratio of number of tables in certain business area across different application layers. For instance it can be seen that the total number of tables in end-user layers (USR/USP) and partner layers (BUS/BAR) in customer area (or Accounts Receivable) and vendor area (or Accounts Payable) is highest compared to the number of tables for this area in vendor (SYS) layer. This indicates that the end-user company in this sample had most modifications in these modules. Other notable business area that was heavily customized is inventory management.

Usage frequency of elements by layer

This aspect identifies whether a part of the system has been used and how often it has been used. For this purpose the original tool was modified to perform the analysis of the whole ERP system including all layers.

Table 2 shows an analysis of a number of forms and their run count since ERP was installed and the layer where this form was last modified. Only a few sample rows have been listed, alphabetically ordered, save for last few rows which show forms without usage.

Table 2: Element usage frequency

Name	Usage Count	Layer
Address	2275	USP
AddressCountryRegion	74	SYS
AddressCountryRegionGroupBLWI	0	SYS
AddressSelect	2	BUS
CustTripJour	0	USP
CustTripJourTable	0	USP
CustVendCreditInvoicingLookup	0	SYS

Elements that have no usage (i.e. where Usage Count is zero) and which are created at appropriate application layers (e.g. USP for end-users) can probably be safely deleted, and will remove the amount of customizations and make upgrade process and maintenance easier. In Table 2 this would be the forms CustTripJour andCustTripJourTable.

There are situations where there are elements that are from the lower layers (SYS) and which are not used (in Table 2 this would be the forms from last two rows). This indicates that there are some out-of-the-box functionalities of ERP system that are not needed. By identifying the parts of ERP system which are not used better view for future development and maintenance of ERP system is achieved. Big saving on upgrade effort is created by not analyzing these parts of the ERP system during the upgrade process. Other way for the company to leverage this information is in vendor/partner negotiations; dropping modules which are not used and appropriate licenses reduces upgrade costs. Module licenses which are not used can even be traded for other modules with some vendors.

RESULTS

The previous analysis can present a clear picture with a quantitative backing about the weak points of the ERP system. For most aspects, investing in correcting all this points could take a lot of time. The presented approach is an agile one. The request priority in the product backlog indicates the priority of improving the previously discovered weak points.

For example, if the next iteration in software development includes working in business areas like HR, we would try to improve some weak points in this area. We would use the modified tools we have created to analyze the whole module for any weak points in form of the best practices violations, usage frequency, distribution of elements by layers, etc. This approach can provide a good overview of the current state of a part of an ERP system rather fast. The efforts needed to fix the identified weak points can be managed in a better way, and can even influence top requests in backlog in regard to their priority. Making improvements to class hierarchy to remedy this some of problematic situations could take a long time, so it might make sense to leave them 'as is', until we have a request to make modifications to the part of the ERP system that these classes are related to.

Continuous improvement by using the above approach would create a long term advantage for an upgrade of an ERP system. It would also influence the team to raise their awareness of good software development practices which are directly connected with quantified results from tools that they can use easily, so they would strive to create high quality software solutions that would reduce their work in maintenance.

Another benefit is that the analysis of different aspects can be run on an ERP system or on parts of the ERP system maintained or developed by partner/vendor companies. We could define minimal software engineering acceptance criteria that a partner would need to reach in order for us to accept their solution. In long term this transfer of good practices downwards (in application layer sense) to partners and to vendors would help the company get future solutions which are more just-in-time like (with less time, less money paid and with higher quality) (Imai, 1986).

Doing an upgrade is usually considered to cost 25-33% of the initial ERP implementation investment (Songini, 2000). Skipping one whole version can be considered an even worthier move that required greater analysis and planning. Using this approach has helped us perform upgrade from version 3 directly to version 5 (Dynamics AX 2009) of the ERP system that we are using.

As the software evolves, the analysis that was presented in this paper helps get benefits and improve the quality of an ERP system even after the initial analysis which was performed at previous periods. It also greatly helped in motivation of developers, by giving them a very productive environment where they can easily implement most of requests presented by management.

CONCLUSIONS

In the recent years there have been very few research results of applying agile methodologies to development and maintenance of ERP systems. The significance of this analysis derives from its different point of view - from technical perspective - and extends to connecting valuable and precisely quantifiable information with an agile development approach of ERP systems, mostly in response to the increased focus to development and maintenance of current ERP system determined by the economical breakdown. In difficult times of a global financial crisis this approach has proven as the most productive and extremely motivating for developers.

Technical perspective was introduced trough different technical aspects. Using data from this analysis and benefiting from an agile approach has resulted in an exceedingly productive way of maintenance and implementation of new requests. It also enabled easier ERP system upgrade resulting in a worthy upgrade to second next version (from version 3 to version 5).

The challenge is to continue to develop and maintain ERP system using technical perspective for agile and productive approach in the years yet to come.

WORKS CITED

(2002). *Achieving, Measuring, and Communicating IT Value.* techreport.

(2010). *Magic Quadrant for ERP for Product-Centric Midmarket Companies.* techreport.

Beatty, R. C., & Williams, C. D. (2006). ERP II: best practices for successfully implementing an ERP upgrade. *Communications of the ACM, 49*, 105--109.

Beck, K. (2001). *Extreme programming explained: embrace change.* Addison-Wesley.

Chidamber, S. R., & Kemerer, C. F. (1991). Towards a metrics suite for object oriented design., *26*, pp. 197--211.

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on, 20*, 476--493.

Cockburn, A. (2004). *Crystal clear: a human-powered methodology for small teams.* Addison-Wesley Professional.

Cockburn, A. (2006). *Agile Software Development: The Cooperative Game (Agile Software Development Series).* Addison-Wesley Professional.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *IEEE Computer, 34*, 131-133.

Dittrich, Y., Vaucouleur, S., & Giff, S. (2009). ERP Customization as Software Engineering: Knowledge Sharing and Cooperation. *Software, IEEE, 26*, 41--47.

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development, 9*, 28--35.

Henderson-Sellers, B. (1996). *Object-oriented metrics: measures of complexity.* Prentice Hall.

Hitz, M., & Montazeri, B. (1995). Measuring coupling and cohesion in object-oriented systems., *50*, pp. 75--76.

Imai, M. (1986). *Kaizen: The key to Japans competitive success* (Vol. 4). McGraw-Hill New York.

Jovicic, B., & Vlajic, S. (2009). Design Patterns Application in the ERP Systems Improvements. *Information Systems Development*, 451-459.

Markus, M., & Tanis, C. (2000). The enterprise systems experience - from adoption to success. In *Framing the domains of IT research: Glimpsing the future through the past.*

McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on software Engineering*, 308--320.

McConnell, S. (2004). Professional software development.

Meszaros, G., & Aston, J. (2007). Agile ERP: You don't know what you've got'till it's gone! *AGILE 2007* (pp. 143-149). IEEE Computer Society.

(n.d.). *Microsoft Dynamics AX 4.0.* techreport.

Nah, F. F., & Delgado, S. (2006). Critical success factors for enterprise resource planning implementation and upgrade. *Journal of Computer Information Systems, 46*, 99.

Olsen, L. D., Pontoppidan, M. F., Skovgaard, H. J., Kaminski, T., Kumar, D., & Thomas, S. (2009). *Inside Microsoft Dynamics AX 2009.* Microsoft Press.

Rosenberg, L. H., & Hyatt, L. E. (1997). Software quality metrics for object-oriented environments. *Crosstalk Journal, April*.

Schwaber, K. (2004). *Agile project management with Scrum* (Vol. 7). Microsoft Press Redmond (Washington).

Schwaber, K., & Beedle, M. (2002). *Agile software development with scrum.* Pearson Prentice-Hall.

Shankar, C. & Bellefroid, V (2011) Microsoft Dynamics Sure Step 2010. Packt Publishing Birmingham UK.

Songini, M. (2000). Users vent frustration over Oracle CRM/ERP upgrades. *Computerworld, 34*, 105.

Togut, D. M., & Bloomberg, E. (2003). *Morgan Stanley Research Report.* techreport.

Wallace, T. F., & Kremzar, M. H. (2001). *ERP: making it happen: the implementers guide to success with enterprise resource planning.* John Wiley & Sons Inc.

Watson, A. H., McCabe, T. J., & Wallace, D. R. (1996). Structured testing: A testing methodology using the cyclomatic complexity metric. *NIST special Publication, 500*, 235.