Application of System Dynamics Modelling in support of Microsoft's Automation Strategy

Siôn Cave [*], Mirek Gliniecki [*], Skip Johnson [**] and Géza Nemesszeghy [**]

*
SRCave@QinetiQ.com /
MJGliniecki@QinetiQ.com

QinetiQ Ltd
Cody Technology Park
Ively Road
Farnborough
Hampshire
GU14 OLX
United Kingdom

**
Microsoft Corp.
One Microsoft Way
Redmond
WA
98052
USA

Abstract

Development of software is a dynamic and complex problem. A number of software development methodologies exist to enable software to be produced effectively. Software development methodologies, such as Waterfall and Agile consist of a set of activities that are carried out in the production of software. Activities include Requirement Capture, Design, Development and Testing. Elements of key software development tasks can be automated to improve quality and free up resource capacity. For example, performing software tests can be a laborious activity which if automated can be carried out quickly and repeatedly without error. However, developing automation takes time and is more cost effective for applications with a long shelf life.

This paper describes an innovative System Dynamics based strategy tool, called the Automated Decision Support Tool (ADS Tool), developed by QinetiQ Ltd and Microsoft. The purpose of the ADS Tool was to assess the optimum level of automation to be used in the development of a software application.

The model is a key element of Microsoft IT (MSIT) Engineering's three year automation roadmap for increasing delivered scope and quality across MSIT Engineering, and is being successfully used by MSIT engineers in Seattle, India and China to develop automation strategies for a number of their internal IT applications.

Key Words: System Dynamics, Software Development, Automation, Project Management

1 Introduction

This paper describes an innovative System Dynamics based strategy tool, called the ADS Tool, developed by QinetiQ Ltd and Microsoft. The purpose of the ADS Tool was to assess the optimum level of automation to be used in the development of a software application.

The development of software is a dynamic and complex problem. A number of software development methodologies exist to enable software to be produced effectively. Software development methodologies, such as Waterfall and Agile, consist of a set of activities that are carried out during the development of a software product. Activities include Requirement Capture, Design, Development and Testing. Elements of the key software development activities can be automated to improve quality and free up resource capacity. For example, performing software tests can be a laborious activity which if automated can be carried out quickly and repeatedly without error. However, developing automation takes time and is more cost effective for applications with a long shelf life. The ADS Tool was developed to enable Microsoft to make strategic assessment on the appropriate automation strategy for different applications.

System Dynamics (SD) is an analytical methodology that can be used to better understand the dynamics and nonlinear interactions within a system. The approach can be used to make robust strategic decisions based on understanding the interactions within the system that drive performance. The System Dynamics approach has been found to be an appropriate method for modelling complex project management issues since it can readily represent [6]:

- The typical 'work' processes associated with project tasks
- The development and depletion of the resources required to carry out the 'work' processes (for example staff)
- Rework cycles
- Project control mechanisms
- Managerial mental models and decision making processes
- Ripple and knock on effects of policy changes

As such, a number of authors have published papers describing the use of System Dynamics for representing software development processes [1], [2], [3], [4], [5], [7], [8], [9], [10], [11], [12]. However, the System Dynamics models described in the literature do not:

- Consider the dynamic resource allocation of a single project team across software releases within and between applications
- Represent the simultaneous use of automated and non-automated development
- Represent the inheritance of attributes, such as Test Cases, Defects and Scope across multiple releases within an application

- Represent the specific training requirements for automation techniques

Further, some of the processes carried out by Microsoft IT in the development of software are unique to Microsoft.

This paper describes an innovative model developed by QinetiQ and Microsoft called the ADS Tool that addresses these issues. Section 2 provides a more detailed description of the specific strategic challenge faced by Microsoft that led to the development of the ADS Tool. Section 3 describes the approach that QinetiQ and Microsoft adopted when developing the model. Section 4 provides an overview of the processes represented in the System Dynamics model contained within the ADS Tool. Section 0 describes the architecture of the ADS Tool. Section 6 explains how the model has been used by Microsoft to develop automation strategies for particular applications and Section 7 discusses the benefits the tool has realised for Microsoft and how it is planned to develop the tool further.

2 Strategic Challenge

As described in Section 1, automation techniques can be used to speed up software development and free up resource capacity but do require an initial investment. Therefore there is a trade-off between automation and non-automated development. Microsoft IT required a tool to enable the benefits of automation across the different activities in the development of an application to be ascertained in order to prioritize their introduction. Key questions that Microsoft needed to be able to answer in order to create a robust automation strategy for an application included what type, and to what extent, should automation be implemented, and what returns could be expected and when would they be realised. Further, the optimum level of automation would vary by technology, application type, current level of automation and staff competency.

A tool was required by Microsoft IT that would take these issues into account when supporting the creation of an automation strategy. This is especially important as automation strategies have a potentially high initial cost to implement as a result of training and infrastructure investment. Further, the benefits of the strategy may not be realised until a number of releases down the line.

The challenge is illustrated graphically in Figure 1:

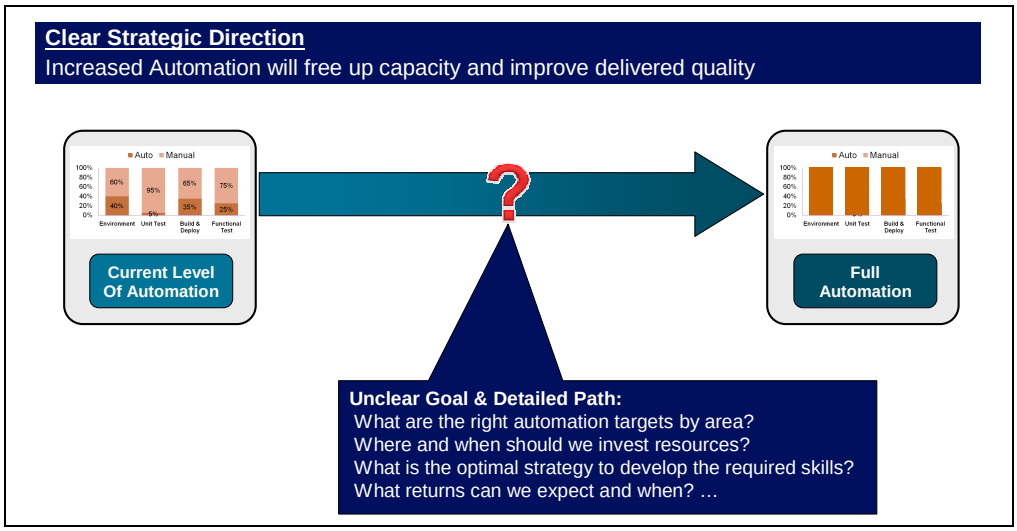

Figure 1: Strategic Challenge

QinetiQ worked with Microsoft to develop a System Dynamics based strategy tool, called the ADS Tool that enables these issues to be quantified and potential automation strategies explored within a 'safe' environment.

3 Development Process

QinetiQ and Microsoft developed the tool collaboratively between November 2009 and April 2010. Over the course of the development process a number of workshops were held at Microsoft offices in Seattle. During the initial workshops group model building exercises were conducted with the aim of:

- Developing generic Stock and Flow diagrams representing Microsoft's software development processes
- Determining how the different automations methods are implemented and their impact on the generic software development processes
- Determining the key strategic levers and outputs required in order to differentiate strategies
- Defining the data required in order to specify an application and the associated release schedule

The workshops were attended by key stakeholders involved in the development of internal software at Microsoft. These included highly experienced Testers, Developers and Program Managers who were able, within the workshop environment, to articulate not only the formal development processes but were also able to describe how these processes were actually executed over the course of a software development process.

Following the development of qualitative models defining the software development processes quantitative model development was carried out. The model was developed using the commercial off the shelf System Dynamics simulation package Powersim. While quantitative model development was being carried out by QinetiQ, Microsoft sourced data for a number of pilot projects from

across MSIT. These pilot projects were used to validate model behaviour during subsequent workshops.

In addition to the workshops held at Microsoft's offices in Seattle, QinetiQ and Microsoft held weekly 'live meetings' over the internet. This ensured that development was progressing to schedule and that model behaviour was representative. These 'live meetings' ensured that the geographic separation between QinetiQ and Microsoft did not have an adverse impact on the model development process.

Development was monitored by Microsoft's Quality and Business Excellence (QBE) department to ensure the final model was a standardised method for representing software development projects.

The final model allows Microsoft to define applications based on their current state and rapidly assess the impact of potential automation strategies. The tool has been used for strategy development from April 2010.

4 Structure of the System Dynamics Model

This Section describes the key structures represented within the System Dynamics model contained in the ADS Tool. The key System Dynamics structures represent:

- Generic software development processes
- Staffing levels and skill
- Infrastructure requirements
- Automation methods

These key structures are described briefly below:

4.1 Generic Software Development Process

An application is composed of a series of software releases that build on previously developed application functionality. Each software release requires the development of code that meets a set of predefined requirements. Each requirement in the release requires effort to carry out the following activities prior to production:

- **Requirements Capture –** Identification and acceptance of the requirements for a particular release.
- **Design –** Development of design documentation based on the defined requirements.
- **Development –** Initial development of software code to meet the requirements as defined in the design documentation.
- **Testing –** Testing of the developed code against a series of test cases which have been developed based on the software requirements and design documentation. A test case is a set of conditions or variables under which a tester will determine whether the developed code functions correctly. If a

defect is identified during testing then it is 'triaged' to assess whether rework is required.

- **Rework –** During rework, the cause of the identified defect is rectified. The rework could require correcting developed code, test cases and/or design documentation. Once rework is carried out the code can be released for further testing.

The extent to which the activities are carried out in parallel will be dependent upon the particular software development methodology being followed. For example, the Waterfall method would have very little parallel work, whereas Agile development would have a much greater extent of parallel work. Further, planned software releases within a single application may have overlapping tasks between releases. This is illustrated in Figure 2 which shows the overlapping tasks for three releases for a single application.

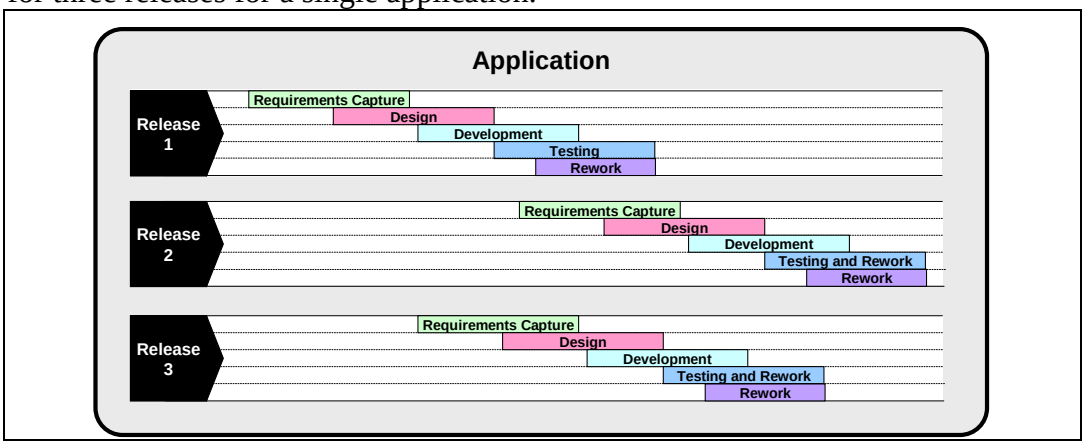

Figure 2: Generic Software Task Schedule

The generic software methodology described above was converted into a Stock Flow diagram that captured the high level Requirements Capture, Design, Development Testing and Rework activities.

Following the development of the high level stock flow diagram, the Requirements Capture, Design, Development, Testing and Rework activities were decomposed into their sub tasks. For example, the Design activity is composed of the development of three particular types of documentation, the Technical Specification, the Functional Specification and the Test Plan. Detailed Stock Flow diagrams were developed that captured each of the sub Tasks.

Each application has a set of attributes that can be inherited between releases. For example test cases developed for the first release of an application can be reused for testing the same functionality in subsequent releases. Attributes that need to be tracked between releases for an application include:

- Test Cases
- Requirements
- Automation levels
- Defects

The ADS tool was required to represent up to 3 applications and up to 16 releases. This was viewed as an optimal number of applications and releases to model, but could be extended if required.

4.2 Staff

The activities described in Section 4.1 are carried out by Microsoft IT staff and/or Vendors. Staff members are formed into project teams that are responsible for different software releases. Different types of staff are required to carry out the different sub tasks across the software development process. For example, Testers are required to carry out sub tasks within the Design and Test tasks. Further, a team may be responsible for different releases within applications across multiple applications. As such, the development of software is a dynamic resource allocation problem.

The different types of staff required for each of the sub tasks were defined during the workshops. This resulted in the requirement for the model to represent 5 key staff types required for the successful delivery of a software release, namely:

- Solution Delivery
- Project Manager
- Systems Analyst
- Developer
- Tester

The allocation priority of staff to a particular release and sub task was ascertained through workshop discussion.

Finally, staff productivity and competency can vary by project team, and staff may require training in the particular automation techniques described in Section 4.4.

The ADS tool was required to represent up to 3 project teams.

4.3 Infrastructure

In addition to the staff resources required for a particular application there are also a number of infrastructure requirements that need to be available. For example the environments used by the developers to develop code need to be created and maintained throughout the development and rework phases.

4.4 Potential Automation

The purpose of project was to provide Microsoft with a tool that enabled the assessment of alternative automation strategies for a particular application. As such it was necessary to identify each of the potential types of automation and the potential impact that they have on the software development process. These are summarised in the Table below:

Type of Automation	Description	Expected Benefits	Required Investments
Environment Build (Development and/or Test)	An environment is the set-up of software and hardware used during development or testing. The software environment must be regularly rebuilt. This process can be carried out programmatically.	• Reduced time spent by Developers and Testers building the Development and Test Environments respectively • Reduced defect rates	• One time big investment per application • Maintenance
Build & Deploy	Build and Deploy is the process of packaging up the application and deploying the software to the Test environment for Testing. The process can be automated programmatically.	• Improved Developer productivity through less time spent doing Build and Deploy activities • More frequent deployments to Test	• One time big investment per application • Ongoing Maintenance
Unit Testing	Unit testing is a method by which developers test individual units of source code as they are being developed. Unit tests can be automated so that they are carried out rapidly and consistently.	• Reduced defect density prior to release to Testing • Less rework	• Developer training • Culture change investments • Longer to develop code as automation must be written
Functional Testing	Functional testing refers to activities that verify a specific action or function of the code and is carried out by the Testers. The test can be automated so that it is carried out rapidly and consistently.	• Reduced time to carry out a test cycle • More frequent test runs • Faster defect detection	• Staff training • More time required to develop an automated test case than a manual test cast • Component library to store automate Test Cases • Ongoing Maintenance

Table 1 – Types of automation considered in the ADS Tool

Each of the potential types of automation was considered in the context of their impact on the generic software development stock flow diagram.

4.5 Defining a 'Successful' Strategy

The purpose of the System Dynamics model was to provide Microsoft with a tool which enabled the rapid development and assessment of automation strategies. As such it was necessary to define the metrics by which the strategy would be considered successful. These were defined to be:

- **Scope** – The proportion of the desired software requirements that could be delivered by the required production date
- **Capacity** – The man hours that have been freed up as a result of the automation
- **Quality** – The number of undiscovered defects remaining in the release

4.6 High Level Functional Model

The System Dynamics model was broken down into functional areas that represented different processes in the software development processes and the

staff and automation functions. This high level representation is shown in Figure 3:

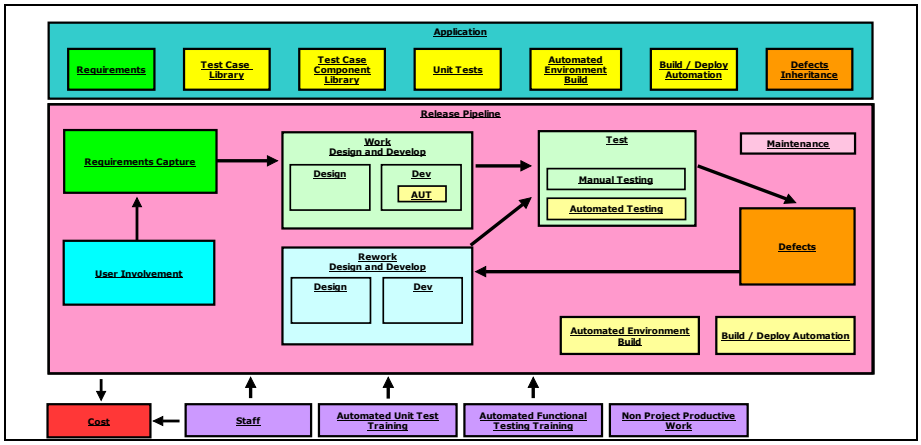

Figure 3: High Level Functional Areas

Each of the model functional areas had fully developed Stock Flow diagrams, as illustrated in Figure 4

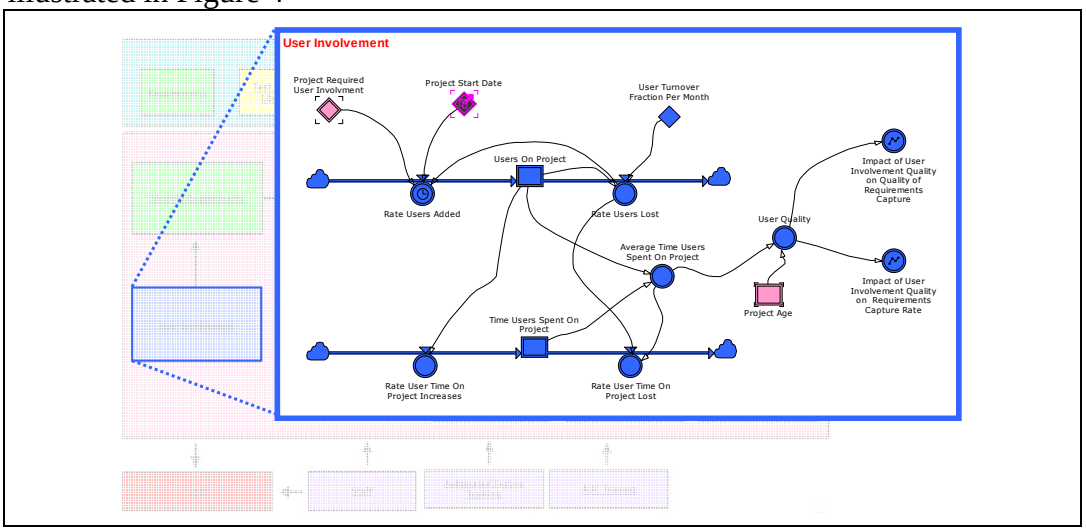

Figure 4: High Level Functional Areas

5 Model Architecture

This Section describes the architecture of the ADS Tool. The model architecture is illustrated in Figure 5 below:

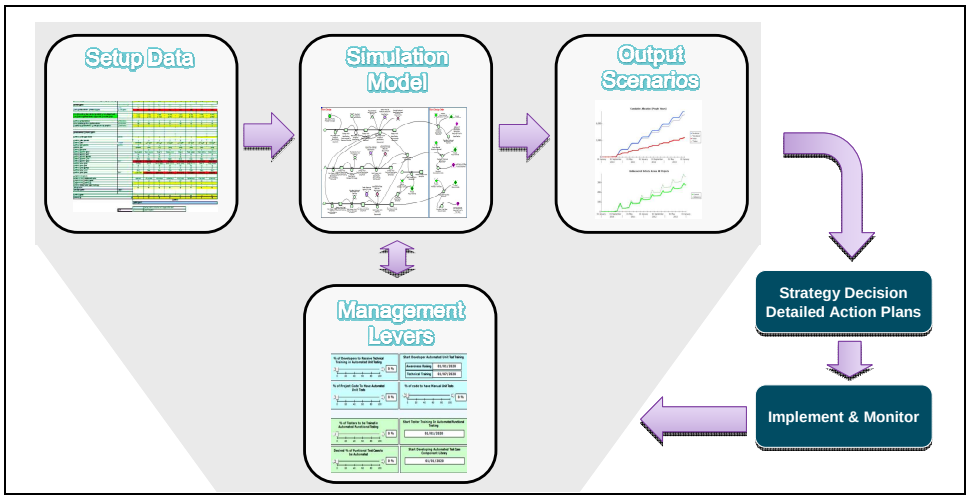

Figure 5: ADS Tool Architecture

The ADS tool was developed using Powersim and used an Excel spreadsheet to store all the model input data that defined the current state. Automation strategies were entered using a management interface developed within the Powersim environment. The management interface allows the user to:

- Define the automation strategy to be tested
- View the results of the scenario in graphical and tabular format. The graphs and tables were set up to show key outputs such as:
    - Application attributes, such as the size of the Test Case library
    - Release Gantt chart displaying when each software development activity is completed for each release within the application
    - Completed scope for the release (i.e. the number of the planned requirements that were completed prior to the production date)
    - Software defects (undiscovered, discovered and fixed)
    - Staff effort applied to each sub task and staff type
    - Staff utilisation and experience
    - Cost

The outputs can be viewed at a release, application or multiple application level.

The management interface also allowed the user to view detailed model structure. A sample views from the model interface is given in Figure 6:

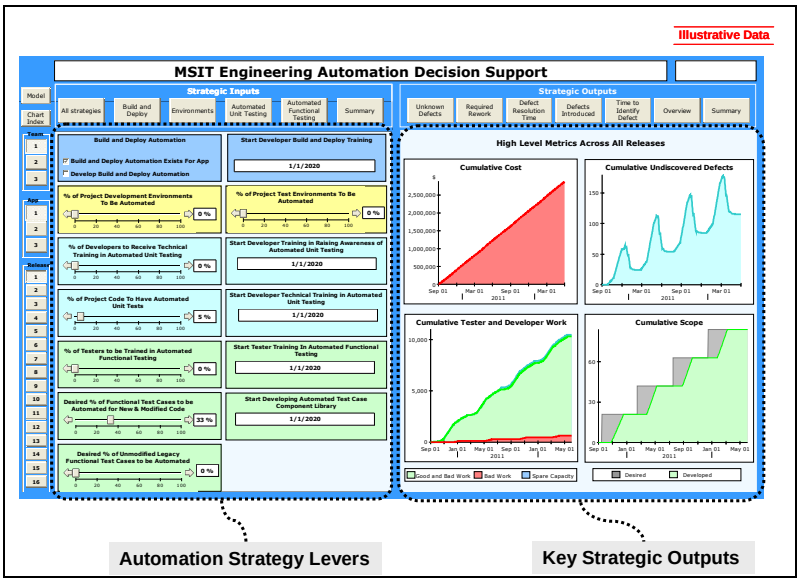

Figure 6: Sample Model Screen Shots

Finally, the results of each run were exported to an Excel Spreadsheet to enable detailed post-processing of the result to be carried out

6 How the Model Is Used To Support Strategy Development

A streamlined and consistent methodology has been developed for the assessment of potential automation strategies for a particular application release. The methodology has been developed to ensure that the maximum benefit can be realised with the minimum of stakeholder effort. The methodology is facilitated by an ADS Tool Specialist. The ADS Tool Specialists are members of the MSIT Engineering community who have a passion for automation and helping teams invest in automation in an optimized way. There are currently twenty ADS Tool Specialists based in Seattle, India and China supporting the 1500 Microsoft IT Engineers.

The strategy development methodology is illustrated in Figure 5:

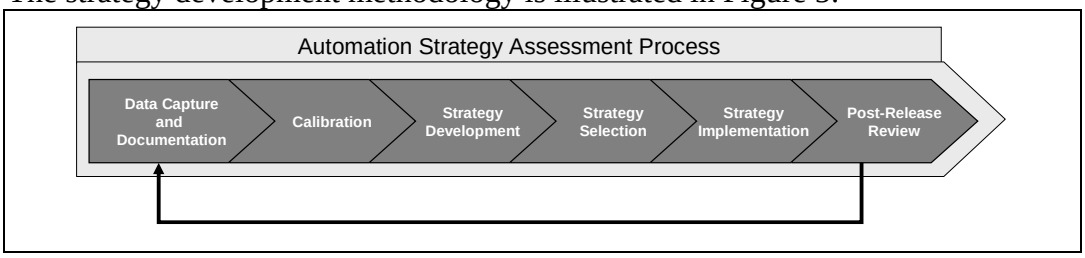

Figure 7: Strategy Assessment Approach

The methodology is composed of the following stages:

1. **Data Capture and Documentation** – A short meeting of the Project Manager, Developer Lead and Test Lead with the Microsoft ADS Tool Specialist to capture the data required by the tool to represent the current state of the application.
2. **Calibration** – Offline refining of the initial model inputs by the ADS Tool Specialist in order to validate model behaviour against historic data. The

calibration may require a short meeting or email exchange between the ADS specialist and the project team to clarify and/or validate the calibration.

3. **Strategy Development** – An initial assessment of the potential automation strategies carried out by the ADS Tool Specialist.
4. **Strategy Selection** - The results of the different strategies are presented to the project team and reviewed to determine the best strategy based on the teams goals. This is carried out over the course of a short meeting.
5. **Strategy Implementation** - The project team plans and executes the selected strategy. The ADS Tool Specialist and the team work together to determine what data will be collected during the first release cycle to help validate the tools results.
6. **Post-Release Review** - The team will meet with the ADS Tool Specialist to review the data collected during the release and validate the actual with predicted results.

The process is repeated for each future release.

The process has so far been used to select the optimum automation strategy for three projects.

7 Conclusions

This project has resulted in the development of a customized tool that meets Microsoft's requirements for strategy assessment. The tool was developed in collaboration with Microsoft IT experts, thus ensuring the validity of the logical representation of the system process and data. The generic nature of the model, with regards to representing any software development methodology (e.g. waterfall, agile etc) provides Microsoft IT with a safe environment to test potential automation strategies prior to large investment.

Qualitative and quantitative System Dynamics techniques were applied over the course of the development of the ADS Tool:

- **Qualitatively** to ensure an agreed, stakeholder owned understanding of the processes involved in developing software at Microsoft
- **Quantitatively** to allow Microsoft to explore potential automation strategies for any potential application in terms of capacity, defects, cost and delivered scope.

The generic nature of the tool also allows Microsoft IT to explore strategic questions beyond the initial scope of the model, as illustrated in Figure 8:

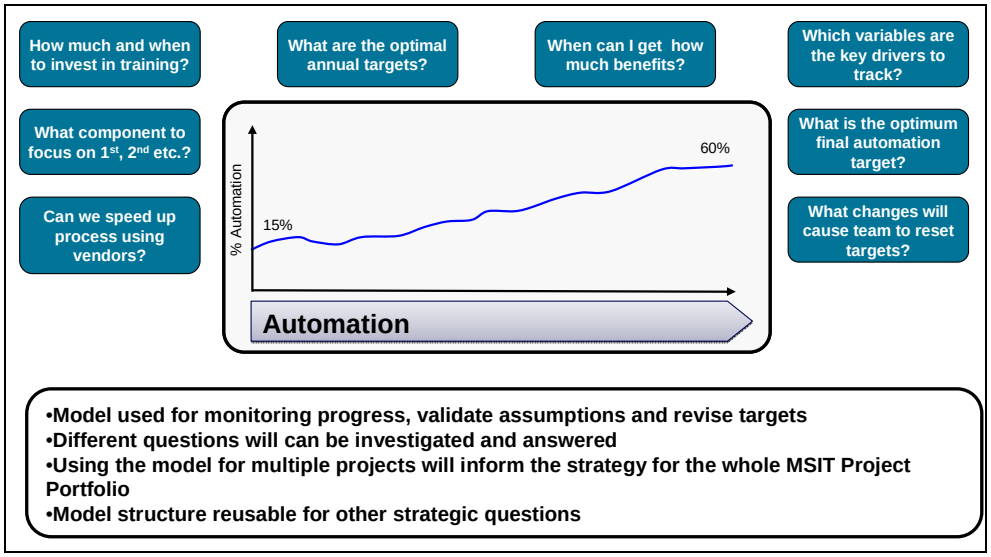

Figure 8: Model Supporting Strategy in a Changing Landscape

The model is a key element of Microsoft IT Engineering's Three year automation roadmap for increasing delivered scope and quality across MSIT Engineering, and is being successfully used by Microsoft engineers in Seattle, India and China to develop automation strategies for a number of their applications.

8 References

[1] Abdel-Hamid and Madnick, S. E (1989). Lessons Learnt from Modeling the Dynamics of Software Development, Management of Computing, Vol 32 No 12

[2] Andersson, C and Karlsson, L (2001), A System Dynamics Simulation Study of a Software Development Process, Lundt Institute of Technology

[3] Calavaro, G. F; Basili, V. R and Iazeolla, G. (1995) Simulation Modeling of Software Development Processes, European Simulation Symposium, 7th, 26-28 Oct. 1995, Nuremberg, Germany

[4] Khosrovian, K; Pfahl, D and Garousi, V (2008). Calibrating a Customisable System Dynamics Model of Generic Software Development Processes, Simula Reseach Laboratory, Technical Report, Simula 2008-02

[5] Lehman, M and Wernick (1998), System Dynamics Models of the Software Evolution Process, Proc. Int. Wrkshp. on the. Principles of Software Evolution, ICSE '98

[6] Lyneis, J. M. and Ford, D. N. (2007), System Dynamics Applied to Project Management: A Survey, Assessment and Directions for Future Research. System Dynamics Review Vol 23, No 2/3

[7] Makio J and Betz S (), A System Dynamics Perspective of Offshore Software Outsourcing – uncovering Correlations between Critical Success Factors

[8] McLucas, A (2008), How to Deliver Multi-Phase Software Development Projects: System Dynamics Simulation of Alternative Project Strategies, The 2008 International Conference of the System Dynamics Society

[9] Saurabh, K (2010), Software Development and Testing: A System Dynamics Simulation and Modeling Approach. Recent Advances in Software Engineering, Parallel and Distributed Systems

[10] SSemaluulu, P and Williams, D (2007), Complexity and Risk in IS Projects: A System Dynamics Approach, In Special Topics in Computing and ICT Research: Strengthening the role of ICT in Development, Kizza J.M, Muhirwe J, Aisbett J., Getao K., Mbarika V., Patel D, and Rodrigues A.J., Eds. Fountain Publishers, Vol 3 pp 243-250.

[11] Tawileh, A; McIntosh, S; Work, B and Ivins, W (2007), The Dynamics of Software Testing, Proceedings of the 25th System Dynamics Conference, July 29 - August 2, 2007, MIT, Boston, USA.

[12] Van Oorshot, K. E, Sengupta, K and Wassenove (2009) Dynamics of Agile Software Development, Proceedings of the 27th International Conference of the System Dynamics Society, July 26-30, 2009, Albuquerque, New Mexico, USA.

-