ATTEST: an Automated-Test-Tool Evaluation and Selection Technology

Mr Daniel Rowley; Monash University; Clayton, Victoria, Australia
Dr Sita Ramakrishnan; Monash University; Clayton, Victoria, Australia

Keywords: Forensic Software Engineering, Technology Evaluation and Selection

Abstract

A significant part of software testing process improvement effort pertains to defect prevention, software testing technology change management and software testing process change management. ATTEST is an automated-test-tool evaluation and selection technology developed by the School of Computer Science & Software Engineering (CSSE) at Monash University in Australia to help SMEs (small- to medium-sized enterprises) improve their management of software testing technology change. Although ATTEST has software-process-improvement-oriented application, it can also be used to help forensic software engineers more easily identify candidate equipment for software-intensive incident and accident investigations. The problem with traditional automated-test-tool (or more generally, computer-aided software engineering (CASE) tools) evaluation and selection techniques is that they provide limited visibility/measurement into the selection (acquisition and/or equipping) of automated-test-tools (or CASE tools). In forensic investigations of software-intensive accidents and incidents, it is important that forensic software engineers correctly identify, measure, and collect the data needed to draw valid conclusions regarding technology adoption. Without an automated-test-tool evaluation and selection process that supports completeness and consistency between evaluations and selections, it becomes difficult for forensic software engineers to justify and evidence their software testing technology change management decisions. While most applications of ATTEST are oriented toward the prevention of software failures (or software-intensive incidents and accidents), we aim to demonstrate that ATTEST also has response-orientated application.

Background

The ATTEST: an Automated-Test-Tool Evaluation and Selection Technology project has been partly funded by the School of CSSE at Monash University to assist with software testing technology adoption in organisations (especially focusing on small- to medium-sized enterprises (SMEs)). This project is an add-on project to the TestIT project funded by the Department of Communication, IT & Arts (DCITA) (URL: http://www.dcita.gov.au/Article/0,,0_1-2_1-4_16089,00.html). One of the aims of that project [12] has been to set up a facility for an independent validation and conformance process for existing commercial software testing tools to address industry's current concerns as articulated by small to medium enterprises (SMEs) and software accreditation bodies such as NATA (National Association of Testing Authorities), Australia. An overview of our work is available at http://honeyant.csse.monash.edu.au/index.html.

The Taxonomy of Computer-Aided Forensic Software Engineering

Forensic software engineering is the utilisation and application of software engineering principles, knowledge, expertise and experience for the purposes of the law (negotiation and mediation) or other dispute resolution processes. In particular, forensic software engineering focuses on research and investigation to determine the relevant data and facts following a software-intensive incident or accident. Rowley and Ramakrishnan [2] argue that forensic software engineering is much like independent verification and validation (IV&V). The IEEE

Standard Glossary of Software Engineering Terminology [8] defines independent verification and validation (IV&V) as "verification and validation performed by an organisation that is technically, managerially, and financially independent of the development organisation". The IEEE Standard Glossary of Software Engineering Terminology also defines verification and validation (V&V) as the "process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfil the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements". According to Rowley and Ramakrishnan [2], forensic software engineering is an outcome of not considering (or mitigating) the likelihood and consequences of software failure whereas IV&V is an upshot of considering (or mitigating) the likelihood and consequences of software failure. Moreover, forensic software engineering involves strict financial, managerial, and technical independence from both clients and suppliers of software-intensive systems. However, despite these differences, forensic software engineering and IV&V both require software testing technologies (techniques and tools) to investigate and report on the correctness of software-intensive systems.

According to Van Wyk and Forno [1], forensic evidence collection processes must be comprehensive, objective, and precise. Automated-test-tools have long been recognised as an effective way to not only improve software development variables such as productivity and product quality but also address the essential difficulty of forensic software analysis: gathering evidence of software failures and faults through clouds of complexity, conformity, changeability, and invisibility. According to Brooks [10], the most difficult work of software engineering is not coding or testing but the essential parts of software engineering. Brook argues that software development is difficult because of the essential complexity, conformity, changeability, and invisibility of software-intensive systems. Moreover, Bruckhaus et al. [3] argue that tools can help improve development processes by facilitating activities that were not practiced before or by supporting activities that are usually carried out with little or no tool support. Schach [6] argues that the simplest form of CASE (computer-aided (or -assisted) software engineering) is the software tool, a product that assists in just one aspect of software production. According to Schach, CASE tools that help the developer during the earlier phases of the process are sometimes termed upperCASE or front-end tools, whereas those that assist with implementation, integration, and maintenance are termed lowerCASE or back-end tools. A CASE workbench is a collection of tools that together support one or two activities, where an activity is a related collection of tasks. Unlike the workbench, which supports one or two activities, an environment supports, at the very least, a large portion of a software process [9].

Our approach to CASE tool evaluation and selection focuses on the V-model which demonstrates how testing activities are related to analysis and design. According to Moriguchi [7], the V-model of software development (see Figure 1.1) is the result of a re-examination of the life cycle model from the point of view of quality assurance. Moriguchi describes the design processes of the V-model as "conversion processes that define [a software solution] in more detail, finally reaching a level of detail that the computer can execute as computer program instructions". We argue that the V-model provides an appropriate process framework for software-intensive incident or accident investigation. Because the V-model details interrelationships between testing and design activities, it is practical for measuring whether or not a developer undertook all reasonable steps to assure software correctness (or more generally, software quality). Furthermore, the V-model can generally be applied to any software development lifecycle and fits into international standard requirements such as ISO 9000.

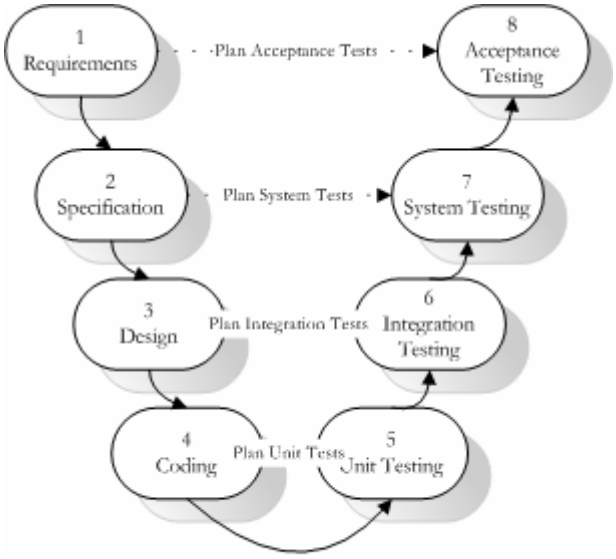

Figure 1.1 - V-Model

By traversing the V-model in a reverse direction (see Figure 1.2) a forensic software engineer is able to appraise the "all reasonable steps"-ness of software product and process documentation. Process documentation that demonstrates "all reasonable steps" to assure software quality is the best defence in software-intensive litigations [14, 15, 16]. Because the V-model mandates that test planning be a part of requirements, specification, design and coding effort, acceptance, system, integration and unit test plans are expected to be compliant with design. In many legal cases, the job of a forensic software engineer is to determine whether or not a software engineer (or team of software engineers) undertook all reasonable steps to ensure that the delivered software product complied with quality requirements. While it could be argued that measuring the compliance of test documentation at different levels of design is tedious work, it is the only way to be sure beyond a reasonable doubt that the developers were or were not negligent with quality assurance (or more specifically, test design and execution). Moreover, if it is not an issue of whether or not the developers were negligent but only whether or not (and how) the software fails, extensive testing still needs to be undertaken to determine the conditions which can cause and caused the software product to miscarry.

While it is obvious that it is difficult (or impossible) to test software fully, automated test tools can help ensure that much of the guesswork/uncertainty and human error is reduced. Furthermore, by viewing or appraising development documentation in the context of the V-model, forensic software engineers are able to better plan and execute their work so that relevant evidence is not excluded. Although relevant, evidence may be excluded if its probative value is substantially outweighed by the danger of unfair prejudice, confusion of the issues, or misleading the jury, or by considerations of undue delay, waste of time, or needless preparation of cumulative evidence. Although using the V-model as a guide for forensic software engineering appears to be a methodical (or breadth-first) heavy-weight analytical approach, it is easy to see that it also accommodates a light-weight inquisitive (or depth-first) style of investigation. That is to say, the tactic does not necessitate that all test results be derived during failure analysis. In some cases, it may be appropriate (or timely) to abandon comb-like search operations (or testing) to concentrate on the validation of a particular hypothesis or casual theory.

Figure 1.3(a) represents a CASE tool that assists with part of the requirements phase (acceptance test planning). Figure 1.3(b) represents a workbench of tools that assist with acceptance test, system test, and integration test planning whereas Figure 1.3(c) depicts an environment that supports all aspects of all phases of the V-model (test planning and test execution). We argue that the forensic software engineering process involves four distinct testing activities: acceptance testing, system testing, integration testing, and unit testing, and four distinct review activities: code review, design review, specification review, and requirements review. When forensic software verification process activities indicate discrepancies between design and test results (poor test coverage), the arrested test specifications need to be corrected and executed or new test specifications need to be designed, written, tested, and executed - in other words, the forensic software validation process begins. In general, the order in which testing activities and review

activities are performed is dependent upon the quantity and quantity of process and product documentation that is made available to the forensic investigator. Nevertheless, we argue that forensic software engineering investigation lifecycles are typified by eight distinct activities. Figure 1.4(a) represents a CASE tool that assists with part of the requirements review phase. Figure 1.4(b) represents a workbench of tools that assist with requirements review, specification, and design review whereas Figure 1.4(c) depicts an environment that supports all aspects of all phases of the forensic V-model (design review and test execution).

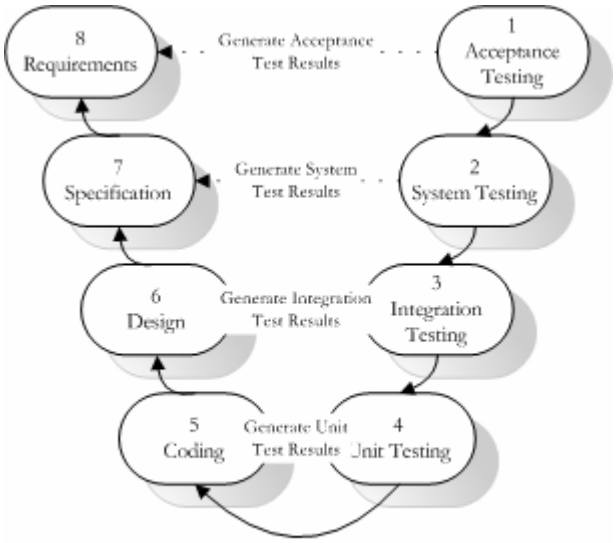

Figure 1.2 - Reverse V-Model

Traditional scorecard systems are traditionally and typically paper-form-based and rely heavily on human effort to construct, validate, maintain and analyse. Furthermore, use of a paper-based system makes it difficult to justify the evaluation and selection of software testing tools when the authenticity of forensic evidence (software failures and faults) is questioned or scrutinised. ATTEST facilitates the mapping of automated-test-tool requirements to automated-test-tool characteristics using a mixture of scorecard evaluation techniques: the evaluation-scorecard technique [13] and the preferred-scorecard technique [13]. The evaluation-scorecard technique and the preferred-scorecard technique have proven to be useful for evaluating and equipping (already-acquired) automated-test-tools however they provide comparatively-minimal insight into whether or not an automated-test-tool acquisition is optimal.

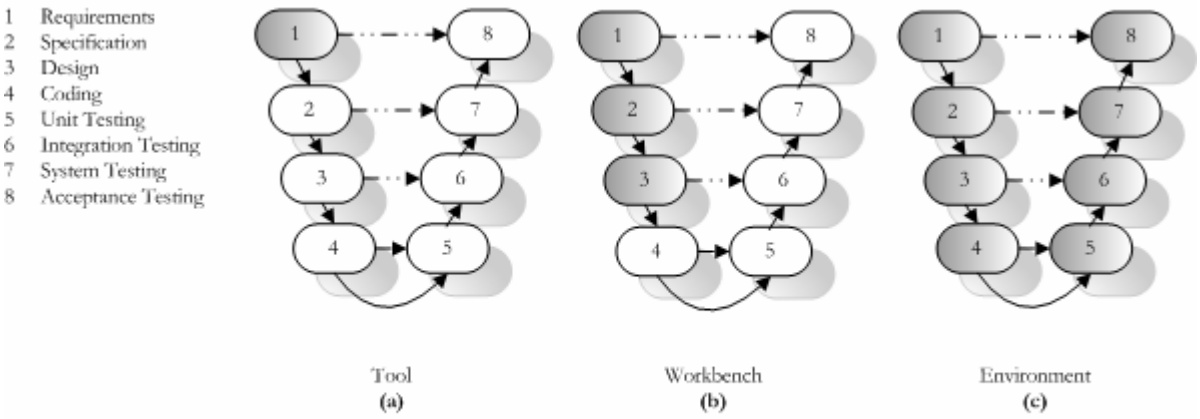

Figure 1.3 - The V-Model and The Taxonomy of CASE

The evaluation scorecard technique involves specifying weighted requirements for an automated-test-tool selection against all characteristics of a technology. On the other hand, the preferred scorecard differs from the evaluation scorecard technique by considering only high-weight requirements (or in other words, highly-preferred characteristics). Both techniques rank

scorecards by their sum of weight-score products however comparative results between both techniques indicate that, in some cases, a CASE tool can obtain two very different rankings using both techniques and the same evaluation scores. The problem with the evaluation-scorecard technique is that it considers all non-zero-weighted characteristics to be essential requirements. In some cases, a CASE tool covering a great number of low-weight requirements can attain a higher ranking than a CASE tool that covers a smaller number of high-weight requirements.

As mentioned earlier, the preferred-scorecard technique only considers high-weight requirements. Another disadvantage of both techniques is that they do not (visibly) separate the specification of requirements from the evaluation of a technology. Moreover, neither technique can distinguish between mandatory (or essential) and optional (or favourable but not essential) requirements. The reason why it is beneficial to distinguish between mandatory and optional requirements is that it allows a forensic engineer (or technology change management groups (TCMGs)) to distinguish between two or more automated-test-tools that cover mandatory requirements equally-well. In some cases (such as the acquisition of new forensic equipment), it may be appropriate to have insight into which automated-test-tools offer additional functionality above that required. In other words, in some situations, it may be appropriate to be cautious about which automated-test-tool characteristics could be required at later dates. On the other hand, in some cases (such as the identification of which already-acquired automated-test-tool to equip), it may not be necessary to separate automated-test-tools that offer additional (but unneeded) features from those that do not.

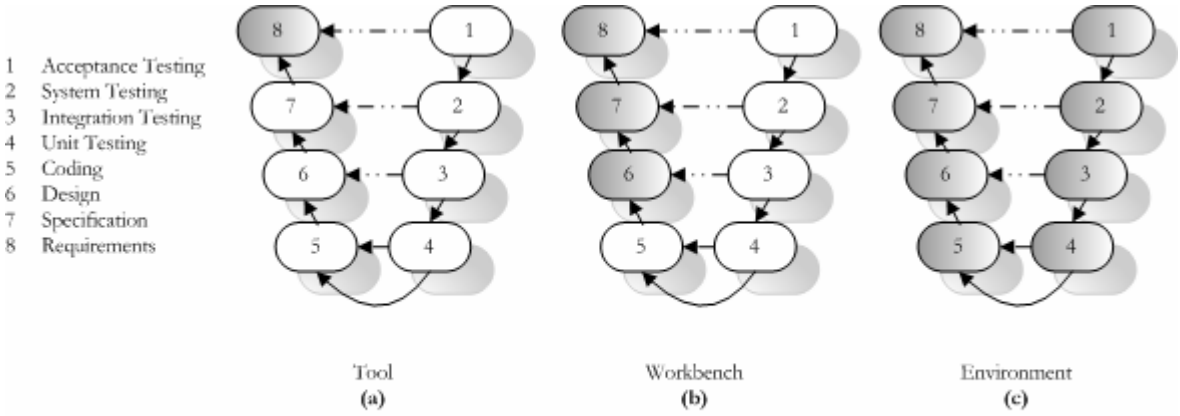

Figure 1.4 - The Reversed V-Model and The Taxonomy of CAFSE

In light of the advantages and disadvantages of the evaluation-scorecard and preferred-scorecard techniques, we propose a new scorecard-technique that allows forensic software engineers (or TCMGs) to specify whether or not a CASE tool requirement is mandatory or optional. Furthermore, our new technique enforces the separation of specification of requirements from the evaluation of a technology; doing so enables requirements specifications and technology evaluations to be reused. Although the evaluation-scorecard and preferred-scorecard techniques both offer a measure of requirement-coverage quality (sum of weight-score products), neither technique explicitly offer a measure of the quantity of requirement-coverage (the percentage of non-zero-weight requirements with non-zero scores). While a trivial computation, a requirements-coverage metric allows forensic software engineers to clearly identify automated-test-tool candidates that satisfy all requirements regardless of rating. In some situations, it may be necessary to select an automated-test-tool based on the quantity and not the quality of the

requirements coverage. Alternatively, it may be appropriate to select an automated-test-tool based not only on the quantity but also the quality of the requirements coverage.

ATTEST: an Automated-Test-Tool Evaluation and Selection Technology

ATTEST is an object-oriented software tool designed to facilitate the evaluation and selection of automated-test-tools that can assist in validating and verifying software products at different levels of design. Brown and Wallnau [4] argue that much of the informality in interpreting any evaluation's results is due to the absence of well-defined goals before starting the evaluation; controlled rigorous techniques for data gathering during the evaluation; and a conceptual framework for analysing the resultant data in the context of existing technologies. In consider of this, we identified two ways to improve the formality in interpreting CASE tool evaluations: by improving the specification of requirements (or definition of goals) and by improving the control and rigor of evaluation data collection. In regard to the controlled, rigorous collection of evaluation data, it is impossible for us (through ATTEST) to provide guidelines on measurement for every feature of all technologies. Instead, we are able to ensure (through design) that evaluators are presented with all the criteria to assess an automated-test-tool against. ATTEST supports the specification of requirements by presenting selectors with all the criteria (or characteristic or features) that can be expected from a particular type of automated-test-tool.

According to Freedman [5], an entity relationship model is a data model that describes attributes of database entities and the relationships among them. Figure 2 depicts an entity relationship model that describes the relational-database entities and entity relationships needed for product-oriented evaluation of CASE tools (products that assists in just one aspect of the production of software). As shown in Figure 1, ATTEST operates on three types of document (or data set): specifications, scorecards, and scoreboards. A specification describes a type of technology (using characteristics) or a set of requirements (using requirements) whereas a scorecard describes the quality of a technology implementation (using scored characteristics). A scoreboard provides sets of measurements for a set of technology implementations (technology scorecards). ATTEST uses folders to collate related specifications, scorecards, and scoreboards. ATTEST can be set up to contain a folder for each distinct software engineering process, activity or task in any software development lifecycle. A technology specification defines the characteristics of a software testing technology (or automated-test-tool) type. A technology specification is a collection of characteristics where each characteristic has a name and a description. On the other hand, a technology scorecard defines the quality of an automated-test-tool in terms of characteristics; in other words, a technology scorecard is an evaluation of an automated-test-tool. A technology scorecard is a collection of scored characteristics where each scored characteristic is a characteristic with a score (between 0 and 100); a characteristic with a high score is a high-quality characteristic whereas a characteristic with a low score is a low-quality characteristic. A technology specification is used to provide evaluators with criteria to assess an automated-test-tool against whereas a technology scorecard is used to enter the results of an automated-test-tool evaluation. A technology scoreboard is a table that lists characteristic coverage metrics pertaining to automated-test-tools: rating, percentage-of-characteristics-covered, and percentage-of-characteristics-not-covered. A rating is a metric that quantifies the quality of an automated-test-tool in terms of characteristic coverage. A percentage-of-characteristics-covered metric describes the quantity of characteristics covered by an automated-test-tool whereas a percentage-of-characteristics-not-covered metric describes the quantity of characteristics not covered by an automated-test-tool. A technology scoreboard is useful for providing evaluators with an overview of the quality of automated-test-tools in a set of automated-test-tools.

While technology scoreboards can provide some insight into which automated-test-tools offer high-quality functionality (characteristic coverage) and/or a high-quantity of functionality, a requirements scoreboard enables forensic engineers to identify those automated-test-tools that provide high-quality coverage of requisite functionality. In some situations, a forensic software engineer may only require a subset of automated-test-tool functionality. In light of this, ATTEST also operates on requirements specifications. A requirements specification describes the requirements of a particular automated-test-tool type. A requirements specification is a collection of requirements where each requirement is a characteristic with a weight (between 1 and 100) and a flag. The weight quantifies the relative importance of the requirement to other requirements whereas the flag indicates whether or not the requirement is mandatory or favourable (optional). A requirement with a weight of zero is not considered to be a requirement (regardless of the typing). The problem with technology scoreboards is that they provide no insight into which automated-test-tools best cover any subset of characteristics. By specifying whether or not a requirement is requisite (mandatory) or favourable (optional), two subsets of requirements can be identified: mandatory requirements and optional requirements. In general, for each subset of requirements, three metrics can be computed (on each automated-test-tool (technology scorecard)): rating, percentage-of-requirements-covered, and percentage-of-requirements-not-covered.

A rating is a metric that quantifies the quality of an automated-test-tool in terms of requirements coverage. A percentage-of-requirements-covered metric describes the quantity of requirements satisfied by an automated-test-tool whereas a percentage-of-requirements-not-covered metric describes the quantity of requirements not satisfied by an automated-test-tool. ATTEST offers two sets of metrics: one set for each subset of requirements (or type of requirement). The set of metrics for mandatory requirements provide insight into which automated-test-tools offer a high-quality and/or high-quantity coverage of requisite automated-test-tool characteristics. On the other hand, the set of metrics for favourable (optional) requirements provide insight into which automated-test-tools offer high-quality and/or high-quantity coverage of favourable (optional) automated-test-tool characteristics. Alike a technology scoreboard, a requirements scoreboard is a table that lists automated-test-tools according to a number of metrics: rating, mandatory rating, optional rating, percentage-of-requirements-covered, percentage-of-requirements-not-covered, percentage-of-mandatory-requirements-covered, percentage-of-mandatory-requirements-not-covered, percentage-of-optional-requirements-covered, and percentage-of-optional-requirements-not-covered.

The benefit of producing a complete specification of a technology (automated-test-tool type) is that it helps forensic software engineers ensure that their specification of requirements is complete and consistent. Although computerisation cannot validate the weighting or typing of automated-test-tool requirements, ATTEST can help ensure that forensic engineers only qualify true automated-test-tool characteristics. The benefit of producing a complete evaluation of an automated-test-tool is that prevents the need for re-evaluation at a later stage (where time may be limited). In regard to requirements specification, ATTEST ensures that selectors are presented with all the criteria (or characteristic or features) that can be expected from a particular type of automated-test-tool.

In terms of maintainability, ATTEST is able to accommodate changes to the specifications of technologies. In some cases, a technology specification may contain an erroneous characteristic (or characteristics) or may omit a characteristic (or characteristics). More significantly, a technology type may evolve over time. At present, ATTEST cascades all additions of characteristics to and updates and deletions of characteristics from technology specifications to technology scorecards. However, modifications to a technology specification often cause one or

more technology scorecards to become outdated (and needy of attention). As the design and development of ATTEST continues, we aim to remain focused on ensuring that the technology solves this and other problems faced by technology evolution.

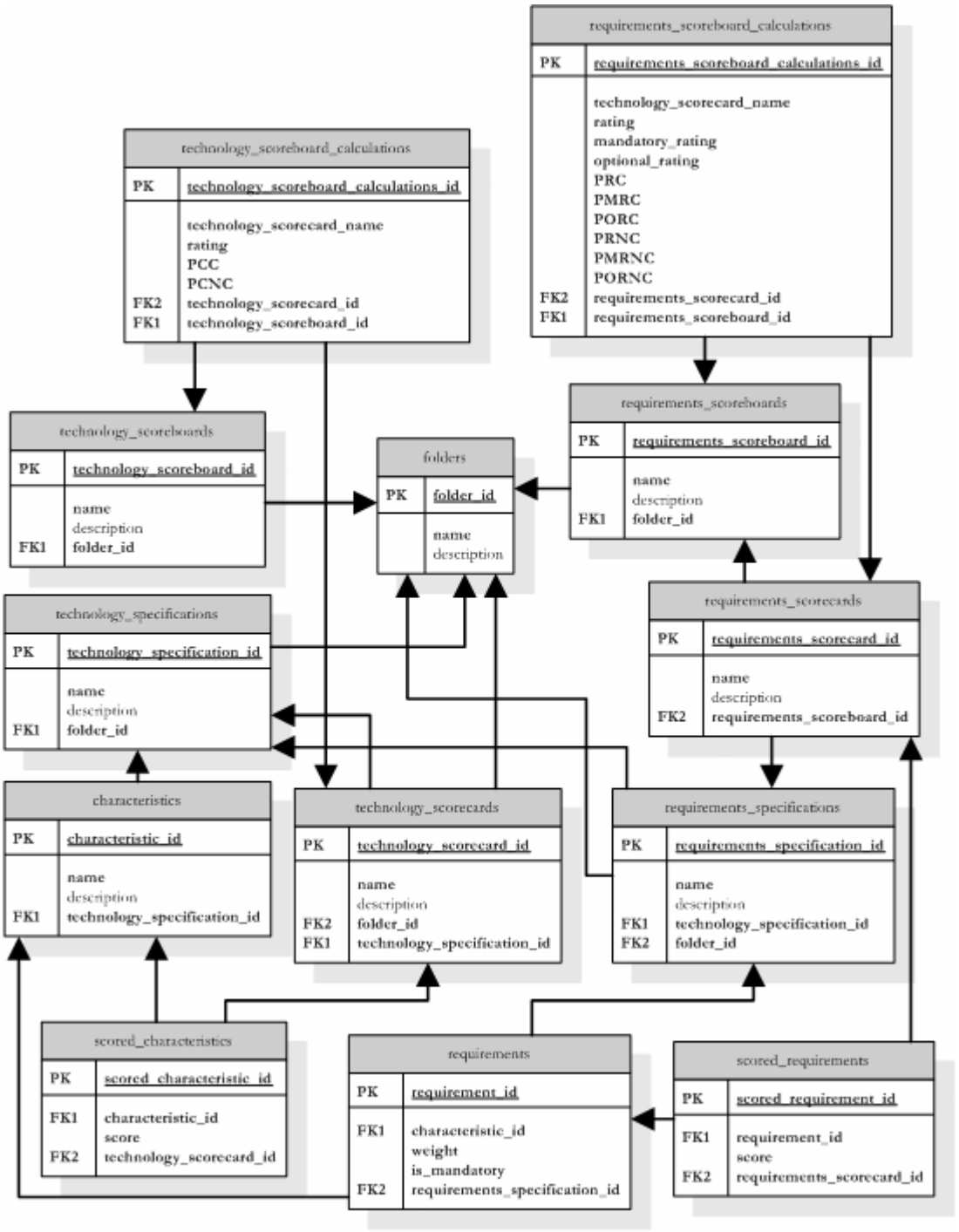

Figure 2 Entity Relationship Model for Computerising the Product-oriented Evaluation of CASE Tools.

Because ATTEST was designed to be general enough to accommodate any software development or software analysis lifecycle model, it is not appropriate to discuss how different CASE tools

support different software development (or forensic software analysis) activity. In fact, many text books describe how CASE tools fit into different parts of the software development lifecycle. Another project within the School of CSSE has elucidated how automated-test-tools support activity in the V-model of software development. Later this year we intend to release a beta version of ATTEST that includes a set of technology specifications that can be used to evaluate automated-test-tools in two contexts: software testing process improvement and software failure investigation and reporting. While it is possible to distribute evaluations of automated-test-tools (given the input/output architecture of ATTEST), we face two problems: not only does the nature of most evaluations tend to be subjective rather than objective (and therefore difficult to validate), some automated-test-tool vendors prohibit the evaluation of their products through stipulations in usage agreements. Nevertheless, forensic software engineering laboratories should perform their own evaluation of equipment as it would help ensure that their selection decisions reflect their understanding of the capabilities and performances of their automated-test-tools.

To realise the function of ATTEST, consider the evaluation of three automated-test-tools (Tool A, Tool B, and Tool C) of type Y that support an activity X (see Table 1.1).

TABLE 1.1 – TECHNOLOGY SCORECARD

Characteristic	Tool A	Tool B	Tool C
characteristic 1	98	0	100
characteristic 2	34	100	87
characteristic 3	56	50	12
characteristic 4	0	97	78

Each evaluation of a tool scores four characteristics that describe Y. While it is relatively trivial to calculate the rating of and the percentage of characteristics covered by each tool in this example (see Table 1.2 and Figure 3.1), it becomes much more difficult when there are tens or hundreds of characteristics to consider.

TABLE 1.2 – TECHNOLOGY SCOREBOARD

	Tool A	Tool B	Tool C
Rating	(98 + 34 + 56 + 0)/400 = 0.4700	(0 + 100 + 50 + 97)/400 = 0.6175	(100 + 87 + 12 + 78)/400 = 0.6925
Percentage of characteristics covered	¾ = 0.7500	¾ = 0.7500	4/4 = 1.000
Percentage of characteristics not covered	¼ = 0.2500	¼ = 0.2500	0/4 = 0.000

In fact, it takes O (n)-time to compute the rating and percentage of characteristics covered by a tool. Moreover, consider the requirements of a tool of type Y to support an activity X (see Table 1.3 and Figure 3.2). Table 1.3 specifies three requirements of a tool of type Y to support an activity X: two heavy-weight mandatory requirements and one mid-weight optional requirement.

TABLE 1.3 – REQUIREMENTS SPECIFICATION

Characteristic	Requirement Weight	Requirement Is Mandatory
characteristic 1	100	true
characteristic 2	60	false
characteristic 3	90	true
characteristic 4	0	false

Calculating the ratings of and the percentages of characteristics covered by each tool in this example (see Table 1.4) is somewhat laborious given the complexity of the computations. Although the time complexity to compute ratings of automated-test-tools in a requirements context is also proportional to the number of characteristics (mapped to requirements), the calculations are much more intricate. Again, it is easy to see that the measurement process becomes more laborious as the number of characteristics to consider increases. In other words, without computerisation, measuring the suitability of a CASE tool selection requires substantial routine (and error-prone) effort.

TABLE 1.4 – REQUIREMENTS SCOREBOARD

	Tool A	Tool B	Tool C
Rating	(100 * 98 + 60 * 34 + 90* 56) / (100 * 100 + 60 * 100 + 90 * 100) = (9800 + 2040 + 5040) / (10000 + 6000 + 9000) = 16880 / 25000 = 0.6752	(100 * 0 + 60 * 100 + 90 * 50) / (100 * 100 + 60 * 100 + 90 * 100) = (0 + 6000 + 4500) / (10000 + 6000 + 9000) = 10500 / 25000 = 0.4200	(100 * 100 + 60 * 87 + 90 * 12) / (100 * 100 + 60 * 100 + 90 * 100) = (10000 + 5220 + 1080) / (10000 + 6000 + 9000) = 16300 / 25000 = 0.6520
Mandatory rating	(100 * 98 + 90 * 56) / (100 * 100 + 90 * 100) = (9800 + 5040) / (10000 + 9000) = 14840 / 19000 = 0.7811	(100 * 0 + 90 * 50) / (100 * 100 + 90 * 100) = (4500) / (10000 + 9000) = 4500 / 19000 = 0.2368	(100 * 0 + 90 * 50) / (100 * 100 + 90 * 100) = (4500) / (10000 + 9000) = 4500 / 19000 = 0.2368
Optional rating	(60 * 34) / (60 * 100) = 2040 / 6000 = 0.3400	(60 * 100) / (60 * 100) = 6000 / 6000 = 1.0000	(60 * 87) / (60 * 100) = 5220 / 6000 = 0.8700
Percentage of requirements covered	3/3 = 1.0000	2/3 = 0.6667	3/3 = 1.0000
Percentage of mandatory requirements covered	2/2 = 1.0000	1/2 = 0.5000	2/2 = 1.0000
Percentage of optional requirements covered	1/1 = 1.0000	1/1 = 1.0000	1/1 = 1.0000
Percentage of requirements not covered	0/3 = 0.0000	1/3 = 0.3333	0/3 = 0.0000
Percentage of mandatory requirements not covered	0/2 = 0.0000	1/2 = 0.5000	0/2 = 0.0000
Percentage of optional requirements not covered	0/1 = 0.0000	0/1 = 0.0000	0/1 = 0.0000

Not only does ATTEST expedite the measurement process, it also facilitates the interpretation and presentation of measurement data. While it is not difficult work to replicate (or duplicate) and reorder data, it is a tedious process that is better managed by computer technology. ATTEST operates a SQL (Structured Query Language) interface (to an implementation of the entity relationship model (in Figure 2)) that not only allows a forensic engineer to enter evaluation and selection (requirements) data but also customise the presentation of report data.

TABLE 1.5 – REQUIREMENTS SCOREBOARD

Candidate	Rating	Mandatory rating	Optional rating	%RC	%MRC	%ORC	%RNC	%MRNC	%ORNC
Tool A	0.6752	0.7811	0.3400	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000
Tool B	0.4200	0.2368	1.0000	0.6667	0.5000	1.0000	0.3333	0.5000	0.0000
Tool C	0.6520	0.2368	0.8700	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000

Table 1.5 and Figure 3.3 display the measurement data from Table 1.4 as presented in ATTEST by default. While Table 1.5 contains much useful information, Miller's law [11], states that at any one time, a human being can concentrate on at most 7 ± 2 quanta of information. In light of this, ATTEST can be controlled to display and order any subset of data columns and records (rows). In continuance of our example, Table 1.6 and Figure 3.4 show a result of using stepwise-refinement and SQL to display the pertinent data needed to select a tool that best covers all (100% of) the requirements for a tool of type Y.

TABLE 1.6 – FILTERED REQUIREMENTS SCOREBOARD

Candidate	Rating	Mandatory rating	Optional rating	%RC	%MRC	%ORC
Tool A	0.6752	0.7811	0.3400	1.0000	1.0000	1.0000
Tool C	0.6520	0.2368	0.8700	1.0000	1.0000	1.0000

Conclusions

At present, ATTEST only supports the evaluation and selection of the simplest form of CAFSE (or more generally, CASE): the software tool. That is, ATTEST is known to be better suited to guiding the acquisition and/or equipping of automated-test-tools than to the acquisition or equipping of automated-test-workbenches or -environments. Furthermore, ATTEST only supports the product-oriented evaluation of CASE tools; future directions for ATTEST aim to not only accommodate the product-oriented evaluation of automated-test-workbenches and -environments but also the process-oriented evaluation of automated-test-tools, -environments and -workbenches. Product-oriented evaluation involves selecting among a set of products that provide similar functionality whereas process-oriented evaluation involves assessing the impact of a new technology on existing practices to understand how it will improve performance or increase quality.

In reality, it is difficult to orthogonally classify automated-test-tools because most modern automated-test-tools support more than one part of the software development (or forensic software analysis) lifecycle. Tools that support more than one software engineering process or task can only be accommodated in ATTEST by producing separate specifications of the tool for each distinct supported process or task. Once a forensic software engineer has identified what task (or type of test planning or test execution) needs to be performed, the engineer can use ATTEST to identify the most appropriate automated-test-tool for that particular task. Again, ATTEST (at this stage) cannot manage with the complexity of identifying optimal automated-test-tool sets (or workbenches or environments) for performing multiple distinct forensic software engineering tasks. Although ATTEST has many useful features (including SQL (structured query language) interfaces and data exportation), it is clear that further work is needed to extend ATTEST into a totally-effectual CASE technology evaluation and selection tool.

Brown and Wallnau [4] maintain that software technology selection, application, and introduction requires consideration of initial technology acquisition cost; long-term effect on quality, time to market, and cost of the organisation's products and services, when using the technology; training and support services' impact of introducing the technology; relationship of this technology to the organisation's future technology plans; and response of direct competitor organisations to this new technology. Although non-technical factors such as acquisition cost are important considerations (in general), ATTEST was designed with focus on ranking automated-test-tools according to their satisfaction of technical (or functional) requirements. Although it is easy to specify non-technical requirements in ATTEST by adding non-technical characteristics into tool specifications, care must be taken to ensure that the weights of non-technical requirements are in proportion to the weights of technical requirements. Alternatively, ATTEST is able to persist delimited-textual shortlists of candidate automated-test-tools into plain text files that can be manipulated by spreadsheet and work processing software. More significantly, we aim to extend ATTEST to allow engineers to pipe shortlists of candidate automated-test-tools back into the short listing process with new requirements specifications so that requirements (or sets of requirements) can not only carry weight but also precedence (or an ordering of importance). The scope of the ISO/IEC 14102:1995 (Information Technology - Guidelines for the Evaluation and Selection of CASE Tools) International Standard is to establish processes and activities to be performed when evaluating different CASE (computer-aided software engineering) tools and selecting the most appropriate for a given organisation and/or project. Although ATTEST was derived from an intention to improve the change management of software testing technologies, it is general enough to be adapted to help evaluate and select COTS (Commercial-Off-The-Shelf)

software components and other types of CASE tools; another direction of ATTEST aims to investigate the feasibility of attaining compliance with ISO/IEC 14102:1995 and other technology evaluation and selection standards.

Using a computerised automated-test-tool evaluation and selection system is an important consideration for forensic software engineering laboratories. Computerising the automated-test-tool evaluation and selection process can help improve the investigation and reporting of software-intensive incidents and accidents because it enables forensic software engineers to more completely and consistently specify their equipment requirements. Although the mapping of characteristics to requirements is a trivial concept, it is a central concept in the design of automated-test-tool evaluation and selection systems that must ensure completeness and consistency between evaluations and selections. This paper has presented and attested a database entity relationship model that describes the database entities and entity relationships needed for computerising the product-oriented evaluation of automated-test-tools (or more generally, CASE tools). Through the demonstration of ATTEST, we aim to prove that regardless of orientation (prevention or response), computerising (and making more formal) the product-oriented evaluation of CASE tools can more easily and more quickly provide confidence in automated-test-tool selections.

References

1. K Van Wyk and R Forno: *Incident Response*, First Edition, O'Reilly & Associates, July 2001
2. D Rowley and S Ramakrishnan: *Forensic applications of software analysis*, in S Lesavich (ed), Proceedings of the Third International Conference: Law and Technology, Cambridge, USA, 6-7 November 2002, ACTA Press, Anaheim, USA, ISBN: 0-88986-333-4, pp 129-134
3. T Bruckhaus. et al: *The Impact of Tools on Software Productivity*, IEEE Software, September 1996, pp 29-38
4. A Brown and K Wallnau: *A Framework for Evaluating Software Technology*, IEEE Software, September 1996, pp 39-49
5. A Freedman: *The computer glossary: the complete illustrated dictionary*, Eighth Edition, AMACOM, 1998
6. S Schach: *Object-oriented and Classical Software Engineering*, Fifth Edition, McGraw-Hill, 2002
7. S Moriguchi: *Software Excellence: A Total Quality Management Guide*, Productivity Press, 1997
8. The IEEE Standard Glossary of Software Engineering Terminology
9. A Fuggetta: *A Classification of CASE Technology*, IEEE Computer, Volume 26, December 1993, pp 25-38
10. F Brooks: *No silver bullet: Essence and accidents of software engineering* (Computer, 1987); in F. P. Brooks, Jr.: The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, Reading, Mass., 1995, pp 177-203
11. G Miller: *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*, The Psychological Review, March 1956, pp 81-97
12. S Ramakrishnan and C Mingins: *A Faciltity for Conformity and Compliance Testing*, Commonwealth Government Australia, Department of Communication IT and the Arts (DCITA) two-year funded Test-IT project, June 2001
13. E Dustin, J Rashka, and J Paul: *Automated Software Testing: Introduction, Management, and Performance*, Addison-Wesley, 1999
14. J Cosgrove: *Software Engineering and the Law*, IEEE Software, Volume 18, Number 3, 2001

15. B Lawson: *An Assessment Methodology for Safety Critical Systems*, bud@damek.kth.se
16. T DeMarco and T Lister: *Both Sides Always Lose: Litigation of Software-intensive Contracts*, Crosstalk, Volume 13, Number 2, 2000