Information and Software Technology xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Improving component selection and monitoring with controlled experimentation and automated measurements

Christoph Becker *, Andreas Rauber

Vienna University of Technology, Institute of Software Technology and Interactive Systems, Favoritenstrasse 9-11/188, 1040 Vienna, Austria

ARTICLE INFO

Article history:
Received 1 June 2009
Received in revised form 27 January 2010
Accepted 1 February 2010
Available online xxxx

Keywords:
Software evaluation
Component selection
COTS
Trust management
Software measurement
Multi-criteria decision making

ABSTRACT

Context: A number of approaches have been proposed for the general problem of software component evaluation and selection. Most approaches come from the field of Component-Based Software Development (CBSD), tackle the problem of Commercial-off-the-shelf component selection and use goal-oriented requirements modelling and multi-criteria decision making techniques. Evaluation of the suitability of components is carried out largely manually and partly relies on subjective judgement. However, in dynamic, distributed environments with high demands for transparent selection processes leading to trustworthy, auditable decisions, subjective judgements and vendor claims are not considered sufficient. Furthermore, continuous monitoring and re-evaluation of components after integration is sometimes needed.

Objective: This paper describes how an evidence-based approach to component evaluation can improve repeatability and reproducibility of component selection under the following conditions: (1) Functional homogeneity of candidate components and (2) High number of components and selection problem instances.

Method: Our evaluation and selection method and tool empirically evaluate candidate components in controlled experiments by applying automated measurements. By analysing the differences to system-development-oriented scenarios, the paper shows how the process of utility analysis can be tailored to fit the problem space, and describes a method geared towards automated evaluation in an empirical setting. We describe tool support and a framework for automated measurements.

We further present a taxonomy of decision criteria for the described scenario and discuss the data collection means needed for each category of criteria.

Results: To evaluate our approach, we discuss a series of case studies in the area of digital preservation. We analyse the criteria defined in these case studies, classify them according to the taxonomy, and discuss the quantitative coverage of automated measurements.

Conclusion: The results of the analysis show that an automated measurement, evaluation and selection framework is necessary and feasible to ensure trusted and repeatable decisions.

1. Introduction

The problem of evaluating and selecting the right software component out of a range of choices to fit in a larger system, often referred to as component selection or Commercial-off-the-shelf (COTS) selection, has received considerable attention in the field of software engineering during the last two decades [40]. The main problem appears in a wide range of different scenarios, from Component-Based Software Development (CBSD) to web service selection and composition.

Most approaches to component evaluation and selection apply a goal-driven approach [76] to support the selection of the most suitable software component. Product quality models and domain-specific criteria catalogues are being developed to structure and reuse knowledge and experience. Most selection methods conform to a general component selection process with the steps *Define criteria, Search for products, Create shortlist, Evaluate candidates, Analyze data and select product* [51]. In general, these approaches are geared towards flexibility and applicability in a wide range of domains, and assume that component selection is a one-off procedure carried out within a development effort to build a new system. Thus evaluation of candidate components against requirements can be done in a largely manual way, which usually implies, but also allows for, high levels of complexity.

However, some of the outlined assumptions are beginning to change. The last decade has seen significant shifts in a number of determining factors for software component evaluation, selection, and integration. Service orientation has become the primary paradigm for decoupling components to build and integrate complex systems; the sheer number of software components in any given system

* Corresponding author. Tel.: +43 1 58801 18818; fax: +43 1 58801 18899.
E-mail addresses: becker@ifs.tuwien.ac.at (C. Becker), rauber@ifs.tuwien.ac.at (A. Rauber).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.02.001

has soared; and the question of trust has become central to the assessment and selection of software, especially of services. Garlan et al. recently re-emphasise that the main challenges for software reuse today are trust, dynamism, architecture evolution, and architecture lock-in [25]. In complex environments with changing requirements, subjective human judgement of software quality and the reliance on declared capabilities of components cannot be considered sufficient evidence for trustworthy decision making, and cannot replace objective measurements obtained in a controlled environment as the basis of decision making. As Terzis recently stated,

> ...the modern view of trust is that trustworthiness is a measurable property that different entities have in various degrees. Trust management is about managing the risks of interactions between entities. Trust is determined on the basis of evidence ... and is situational – that is, an entity's trustworthiness differs depending on the context of the interaction [73].

If an entity's trustworthiness has to be validated in the context of an interaction, we need to do so in a controlled environment where the varying parameters are known and the outcomes repeatable, reproducible, and measurable.

In many cases, component evaluation is not done once when constructing a system, but needs to be managed as a recurring operation where components need to be monitored to detect mismatches and eventually reconfigured or replaced when they prove to be unsuitable in a continuously changing environment. This usually occurs in environments where the functionality offered by competing components or services is sufficiently standardised and well-defined, so that non-functional quality attributes are of primary importance. Examples are data transformation routines in large-scale information systems such as content migration in digital library systems for content preservation [7,23,31,71], or machine translation modules [41].

In digital preservation, which we will use as our main running example in this paper, the component is sought that best preserves a potentially large number of digital objects, such as images or documents, by converting them to more suitable formats in a way that meets requirements specific to each context. As objects and requirements evolve, the suitability of these conversion components needs to be solidly established and monitored. In this area, the level of responsibility of institutions such as national archives implies that a trusted evaluation and selection process is vital and needs to be auditable. This is emphasised by auditing initiatives for trusted repositories such as TRAC [1], which is currently in the process of ISO standardization. In machine translation, components have similarly focussed and well-defined functionality, and techniques for automated evaluation of translation quality are being developed [59,20,47]. Again, thorough evaluation and continuous monitoring of a translation component is required to cope with e.g. topic drift in the source documents. Similar requirements can be found in numerous application domains such as compression tools or sort and search in high-dimensional index structures. We will use machine translation as a second scenario, briefly pointing at the corresponding concepts alongside the main application scenario of digital preservation, to demonstrate the principles of our approach in two very different domains.

The concepts and the system presented in this paper support auditable selection and continuous monitoring of components via plugins that encapsulate general and domain-specific measurements. They can be applied beneficially in settings sharing the following characteristics:

1. *Homogeneous functionality*. The functionality of components is homogeneous and well-defined. Competing tools will provide the same functionality, allowing the creation of dedicated evaluation modules for these components.
2. *Continuous evaluation and monitoring*. The selection process has to be repeated regularly, potentially leading to a reconfiguration or replacement of components.
3. *Transparent and auditable decisions* are necessary to support the critical requirement of trust in software components and services. Specific quality requirements might be negotiable, but a thorough and objective documentation about the evidence that was available at the time of decision making is of vital importance. Thus, decision making and component selection procedures need to be fully transparent and reproducible to provide sufficient levels of accountability.

These peculiarities on the one hand have the effect that the existing approaches for evaluation and selection do not fully satisfy the needs of the scenario, as will be discussed in the next section. On the other hand, it provides opportunities for leveraging the scale of the problem space to employ truly automated evaluation techniques for the selection process.

In this paper we present a framework and system for evaluating and selecting components in an empirical process by assessing them in a controlled setting using automated measurements to ensure repeatability and reproducibility. The method conforms to generic selection processes as analysed in the literature [40,51]. Existing multi-criteria decision making approaches were tailored to fit the needs of auditable, trustworthy decision processes. The outcomes of the controlled experiments are analysed in a quantitative and repeatable manner according to hierarchically defined criteria. An extensible framework supports automated measurements for a variety of generic and domain-specific quality attributes. We have developed extensive tool support for our method, which is increasingly being used by large institutions for decision making. The tool *Plato* (Planning Tool) is available as Open Source, with a deployed server freely available for use at the Plato Homepage.[1]

We have applied this selection method and our tool in the area of digital preservation in a series of case studies which are discussed. Criteria to be measured can be organised along a taxonomy extracted from a number of case studies. While stemming from a specific domain, a first evaluation reveals that it is more widely applicable. Based on a quantitative analysis, we show that an automated measurement, evaluation and selection framework is necessary and feasible to ensure trusted and repeatable decisions and enable scalable monitoring.

The remainder of this paper is structured as follows. The next section outlines related work in the area of software component evaluation and selection and web service quality, and presents our primary application area of digital preservation that we use as a running example and as the context of later evaluation. Section 3 describes the method for requirements definition and component selection. It first explains the main procedure on a high level, comparing it to other approaches and providing a simplified example for its usage. It further describes the high-level architecture of the corresponding tool support. Section 4 describes how the framework for automated measurements of criteria is implemented in a digital preservation environment. It explains the requirements specific to the scenario and the tools needed to provide an appropriately controlled environment with automated measurements. It further defines a taxonomy of criteria categories and explains how data are collected for each category. Section 5 discusses the claim that automated measurements are a key to trustworthy component evaluation by analysing the distribution of criteria defined in nine case studies according to the taxonomy, and discusses threats to the validity of the claim. Section 6 draws conclusions and gives directions for future work.

[1] http://www.ifs.tuwien.ac.at/dp/plato.

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

2. Related work

2.1. Component evaluation and selection

Software component evaluation and selection has received considerable attention during the last two decades. Thorough reviews of literature have been presented in [40,51]. One of the first selection methods presented was the Off-the-Shelf-Option (OTSO) [44,45]. It provides a repeatable process for evaluating, selecting and implementing reusable software components. OTSO relies on the Analytic Hierarchy Process (AHP) [67] to facilitate evaluation against hierarchically defined criteria. Using AHP, the relative importance factor of each criterion is obtained through series of pairwise comparisons which result in a ranking matrix. The resulting eigenvalues are used for calculating relative weights of all factors on each level of the hierarchy.

Ncube discusses limitations of multi-criteria decision making techniques such as the Weighted Scoring Method (WSM) or Analytic Hierarchy Process (AHP), which are often used in component selection [53]. While WSM has earned criticism for the necessity to determine criteria weights in advance without having seen alternative solutions, AHP is problematic because of the sheer complexity and effort that is introduced by the pairwise comparison of criteria [55,48,53]. For n criteria, the number of comparisons is $n(n-1)/2$. Perini describes results of an empirical study comparing AHP with a Case-based ranking approach called *CBRank* that reduces the number of pairwise comparisons. They analyse time consumption, ease of use, and accuracy. AHP outperformed CBRank in terms of accuracy, but was far more time-consuming to use [60]. Another drawback of AHP is that adding or deleting candidate components can lead to changes in the ranking [40].

Alves discusses several peculiarities of COTS requirements engineering in [3], introducing the notion of conflict management. Goals and features need to be acquired and matched, and arising conflicts resolved. They propose a selection process called CRE which identifies four dimensions of COTS selection: domain coverage, time restriction, costs rating, and vendor guarantees [4]. While CRE emphasises non-functional requirements, it "does not address the issues of quality testing" [3]. The PORE method [54] progressively filters candidates by iteratively refined evaluation criteria.

Considerable effort has been spent in standardising software product quality models. The ISO/IEC 9126 standards [35] provide guidance for quality models and define a hierarchy of high-level quality attributes. Quality measures are based on measurement procedures recommended in ISO 15939 [32]. The model distinguishes between three types of quality:

1. *Internal software quality* refers to static attributes such as complexity measures that can be obtained by analysing the source code. Measures are primarily made during the development stages.
2. *External software quality* refers to the behaviour of a system that includes the component such as the number of failures found during testing. Measurements are generally taken during testing and operation.
3. *Quality in use* describes in how far the usage of a component satisfies user needs. Measurements of these attributes have to be taken in a realistic environment [39].

The successor ISO 25000 standards for Software Product Quality Requirements and Evaluation (SQuaRE) combine the ISO 9126 models with evaluation procedures based on ISO 14598 [33]. They also define requirements on the specification of software product quality criteria [39].

Franch describes hierarchical quality models for component selection based on the ISO 9126 quality model in [24]. They propose a six-step method for defining a hierarchy of quality attributes for a specific domain in a top-down fashion:

1. determining quality subcharacteristics,
2. defining a hierarchy of subcharacteristics,
3. decomposing subcharacteristics into attributes,
4. decomposing derived attributes into basic ones,
5. stating relationships between quality entities, and
6. determining metrics for attributes.

This procedure of hierarchical structuring is applicable in the method described in this paper which relies on a similar tree structure.

Carvallo discusses experiences with quality criteria [16] and proposes a method called RECSS to support structuring of the system environment and requirements elicitation [15]. The approach results in thoroughly defined hierarchies of quality factors and the relationships between them. However, requirements are not discussed in detail. They focus on creating a match between features and user requirements, emphasising requirements flexibility.

Techniques from Search-Based Software Engineering [30] have increasingly been applied to the problem of component selection, primarily to rank and select components and find near-optimal solutions for the selection problem in large search spaces. Baker describes ranking and selection as a feature set selection problem and shows that expert judgement in selecting sets of components from a database is outperformed by automated search algorithms [6]. Similarly, Vijayalakshmi et al. apply a genetic algorithm with a WSM fitness function to select components from a large set of candidates [77]. DEER [18] focuses on the tasks of ranking and selection of components in the requirements stage of component-based development. It assumes that evaluation values are known and provides a heuristic to rank and select components or assemblies of multiple components.

Ochs et al. describe a method for measurement-based assessment and selection called CAP [57,58]. The method comprises a four-level taxonomy of evaluation criteria, several specification and assessment activities, and a control workflow. Like other methods, CAP is based on the view that measurement of all applicable criteria is too difficult and expensive, which is true for many classic CBSD scenarios. Hence, the CAP heuristic was designed to increase efficiency by reducing the number of actual measurement operations. CAP goes considerably further than other methods in defining accurate metrics. Yet, it does not appear to use controlled experimentation and automated measurements to collect data on the components, even though this may be both feasible and desirable in other settings. The primary goal is to avoid taking measurements in order to increase the efficiency of the decision making procedure.

Work on component selection arose from the component-based system development paradigm [17], where components can be selected for inclusion into a system that is being built during any phase from requirements analysis to system integration. The level of requirements and available information throughout these phases generally change, and different methods have been suggested for different stages. One of the primary requirements for assessment methods in that context is efficiency. Once the components are selected and the system is built, there is often no reassessment. However, Yang et al. discuss the issues of changing COTS components in large component-based systems and emphasise the need for monitoring and reassessing components, pointing out that typical release cycles average about 10 months [79].

Towards the other end of the dynamics spectrum we find web service composition and web service quality (QoS) [21,22,50,63,75]. Web services are often selected and composed dynamically at runtime, and QoS information has to be collected on-the-fly with little overhead. Quality attributes need to be modelled,

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

Table 1
Range of approaches to component evaluation and selection.

	Component selection for CBSD	Web service selection and composition
Scenario	Static selection sometimes followed by monitoring	Dynamic selection and continuous monitoring of QoS
Granularity of components	Different levels of granularity and features	Fine-grained services with very homogeneous features
Quality attributes	Focus on feature sets and quality models	Focus on low-overhead measurements of generic attributes in production
Data collection	Expert knowledge and standardised quality models	Automated measurements with minimum overhead in service delivery
Target values	Structured definition of requirements and desired target ranges	Automated optimisation of basic QoS attributes

measured, evaluated, and monitored constantly. Trust management is considered a central aspect of current web service research [14,49,70].

Table 1 highlights some key aspects that differ across the range of component selection scenarios. The approach we are presenting here draws from both ends of the spectrum; we will discuss the relation to existing methods in Section 6.

2.2. Digital preservation

The application area where we have developed our approach is digital preservation. This denotes the efforts to preserve digital content for a given purpose over long periods of time. While analog objects such as photographs or books directly represent the content, digital objects are often useless without the technical environment they have been designed for. In contrast to a book, a Word document cannot be opened and read without a suitable hardware and software environment. The constant changes in IT environments render objects incompatible within years and thus challenge the longevity of digital information. A variety of reasons cause obsolescence, e.g. media failure, file formats and tools becoming obsolete, or the loss of necessary metadata. Especially for born-digital material this often means that the contained information is lost completely. Digital preservation has thus become an active area of research in the last decade, as many memory institutions realised that the information they are responsible for will cease to exist within years [52,72].

The two dominant types of *preservation actions* taken to keep digital content alive today are migration and emulation. While migration transforms the objects to more widely accessible representations, emulation creates a technical environment where the objects can be rendered. Consider a collection of electronic documents created years ago on an old operating system running a now obsolete version of Microsoft Word. One could convert these documents to standardised formats such as the Open Document Format ODF [38] or the archival PDF standard PDF/A [37]. The latter would limit the possibility to further edit the documents and instead rely on a file format that is widely supported and considered stable. Emulating the original environment is another option, which would retain the look-and-feel of the original environment.

A growing number of components performing migration and emulation are available today; each tool has particular strengths and weaknesses, and most often, there is no optimal solution. Some migration tools are unable to convert tables properly; others show weaknesses in converting character encoding or scale poorly. On the other hand, requirements vary across institutions and domains, and for each setting, very specific constraints apply that need to be considered. The decision for a tool is further complicated by the variation in the digital content that has to be preserved. The selection of the most suitable component to keep a type of digital object alive when the original technical environment ceases to exist is a highly complex selection problem with several peculiarities which conform to the scenario outlined in Section 1: highly homogeneous and well-specified core functionality across components, complex evaluation of quality across settings, and a high need for automation, standardisation, and documentation. The components need to be monitored continually and will likely be replaced in the future when the requirements or characteristics of source objects have shifted or the alternative options have improved so that preferences have changed as an effect. Thus, it is not a one-off component selection problem but a recurring issue, where continuous monitoring is needed to keep track of the performance of deployed components. Deviation from specified monitoring conditions leads to an identified need for re-evaluation and a potential revision of the selection at a later point in time. The *integration*, on the other hand, can in the extreme case be fully automated, when an already wrapped component is used inside a service-oriented repository architecture.

Digital preservation up to now takes place almost exclusively in large institutions with legal mandates to preserve the cultural heritage of our times. This implies a high level of responsibility and a corresponding need for accountability in decision making, as preserving digital records on a national level also means being able to prove authenticity [27,1]. Thus, trustworthiness is probably the most fundamental requirement that a digital repository preserving content over the long term has to meet. The Trusted Repository Audit and Certification Criteria (TRAC), a widely recognised step towards standardisation and certification of digital repositories, define a set of requirements to be followed in digital preservation processes [1]. Evaluating software components for digital preservation is a key function of the *preservation planning* activity in the ISO Reference Model for an Open Archival Information System (OAIS) [36]. In the methodology described in [71], evaluation is carried out in an 11-step procedure in the classic three phases of identifying requirements, evaluating alternatives and analysing results to arrive at a recommendation for a specific component. The underlying assumption is that decisions need to be based on evidence to ensure continued trust in a repository's operation [64].

The evaluation and selection approach described in this paper has been applied in a series of case studies conducted in collaboration with large institutions across Europe. Examples include the evaluation of preservation action components for electronic publications in a national library [13]; requirements specification for the preservation of web archives and electronic documents with national archives [71]; evaluation of conversion tools for large archives of scanned images in several national libraries [8,46] and an evaluation of components for preserving computer video games [29]. These case studies have led to recent refinements and extensions of the methodology. A core focus of work is the automation of measurements in a controlled environment.

2.3. Observations

The issue of evaluating and selecting software components is relevant in a wide range of scenarios, from component-based software development to web service selection and composition. From

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

the extensive discussion on the different properties of proposed approaches and investigation lines in software component evaluation and selection, several observations can be drawn.

- While there is a number of approaches for selecting software packages, the concrete evaluation procedure is hardly addressed explicitly. There is a need for evaluation techniques and supporting systems [40].
- Continued evaluation and monitoring after the selection is needed [79]; however, few approaches explicitly cover this aspect [40].
- The common perspective of component evaluation and selection research and practice is that '... it is not possible from a practical point of view to measure all the subcharacteristics for all parts of a large software product' [24]. Thus existing approaches prefer to rely on shared knowledge bases collecting manually obtained information about the properties of specific software packages. While this is certainly true for complex software systems with a large number of functions, it is feasible to measure critical quality criteria for components that are more narrowly defined in functional terms. In digital preservation the automated measurement of *accuracy* is deemed essential for ensuring that the cultural heritage of our times will remain accessible in the future [12,64].

The main research question we are focusing on is: how can we provide a trustworthy, reproducible and repeatable evaluation and selection method and tool that is scalable enough to be used in the described scenarios? The main claim of this paper is that controlled experimentation and automated measurements can be used to rigorously evaluate components for ranking and selection, provided that the following two conditions are met:

1. The functionality of components is homogeneous and well-defined.
2. The number of components and instances of the selection problem is sufficiently high to value the additional effort needed to setup a controlled environment and develop the tools and techniques needed for automated measurements.

We evaluate this claim, analysing real-world case studies and discussing the criteria defined therein for measurability. Hence, our primary evaluation goal compares the number of measurable criteria against those that defy measurement. The main questions are thus: (1) What categories of criteria have to be evaluated? (2) How many of them can be measured? (3) How large is the effort needed to take measurements? (4) How can we ensure the measurements are correct? We will discuss these questions in the light of our work in Section 5.

The next section describes an empirical methodology for the specification of requirements and the evaluation of potential components using controlled experimentation. We discuss the general concepts of the selection method and describe the supporting tool and infrastructure. Section 4 will describe the categories of criteria that need to be evaluated in our target area and discuss the question of measurement for each of them, while Section 5 will return to the questions posed above.

3. A framework for automated component evaluation and selection

3.1. Introduction

In the previous sections we outlined the need for a software evaluation and selection method and tool which enables automated and auditable decision making. We argue that it is possible to use automated measurements of quality attributes by conducting controlled experiments with the candidate components, provided that the functionality offered by these components is homogeneous.

This section outlines the framework we use for component evaluation and selection. It is based on a distributed architecture of registries and services that enable the controlled and automated evaluation of software components. The method is based on a customized variation of utility analysis [78]. This section will outline the high-level workflow and the main concepts behind the method, and describe the tool architecture we have developed. In the next section, we will discuss how the framework is being put to use to collect evaluation data in a controlled environment through automated measurements.

3.2. Workflow

Fig. 1 abstracts the principal steps and building blocks of the evaluation environment. The steps of the workflow are similar to the general COTS selection process [51] (GCS), but include a final stage where the software product is integrated into the system, and a continuous monitoring activity after product integration:

1. define requirements,
2. evaluate components,
3. analyse results,
4. integrate product, and
5. monitor requirements, quality of service, and the environment.

This corresponds to standard MCDM workflows. The main differences are that importance weightings are refined in the analysis phase, after measurements have been taken, and that monitoring and re-evaluation are an integral part of the method, relying on automated measurements and QoS specifications. The next sections will give a high-level overview of the main stages of the workflow. A detailed description of the 14 individual steps that these stages consist of is provided in [8].

3.2.1. Requirements definition

Requirements definition relies on a variation of multi-attribute utility analysis, which was originally developed to evaluate public infrastructure projects and has been applied to a wide range of

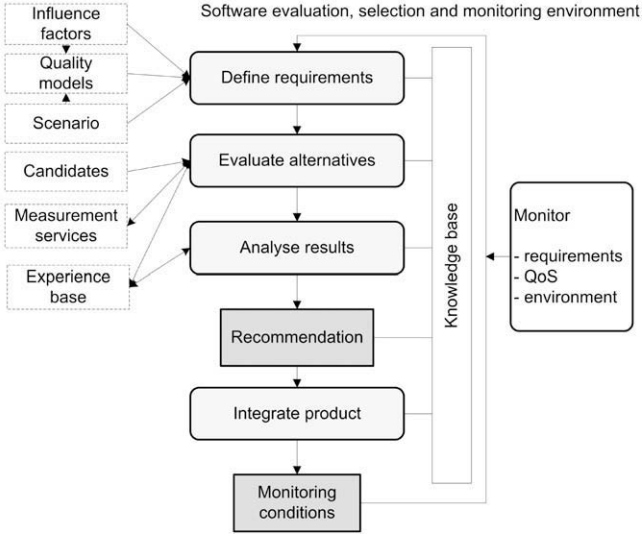


Fig. 1. Software evaluation, selection and monitoring.

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

selection problems [65,78]. Goals and criteria are specified in a hierarchical manner, starting at high-level goals and breaking them down until quantifiable criteria are found at the bottom level of the hierarchy. The problem of incommensurable values [2] is tackled by defining a *utility function* which transforms measured values to a common utility value which can be compared across alternatives and aggregated in the goal hierarchy. Relative importance factors on each level of the hierarchy model the preferences among the stakeholders.

Existing quality models provide sophisticated specification of quality criteria, metrics, and their relationships [16,24,35]. While these concepts provide for powerful modelling tools, they are sometimes difficult to use, especially for decision makers not familiar with them. In the component selection tool DesCOTS, the task of defining these models is thus transferred to COTS evaluation experts that model domains and evaluate products [28,61,62]. The hierarchical definition of requirements in the approach described here is not as formally strict as these models; hence, the requirements can be specified by domain experts themselves, without dedicated external assistance. Depending on the level of experience in a specific organisation with the selection scenario, the requirements definition process ranges from a simple reuse and customization of existing quality models and requirements trees to interactive group sessions, where software support aids in the definition process.

It should be noted that it is possible to use standard criteria catalogues such as the ISO/IEC 9126 and 25,000 families. In fact, some of these are part of a growing knowledge base in the software tool Plato. But the method itself does not constrain the exact hierarchy of criteria and allows different views on the requirements hierarchy. A common approach is to start from a standard catalogue and refine certain criteria such as *accuracy* with domain-specific quality models. This will be discussed in Section 4.1.

On the bottom level of the requirements tree, measurable criteria have to be defined such as *processing speed per megabyte* measured in milliseconds, or *output format is ISO standardised* with a binary value. These criteria are annotated with information on how to obtain the actual measurement data for the candidates during the evaluation stage. Several scales sometimes used in multi-criteria decision making approaches are not employed. For example, 'forced ranking' as establishing a preference order for the considered options is not used, since the level of reasoning and provided documentation is not considered sufficient. Traceability and transparency of evaluations call for breaking down such a ranking into the decisive aspects of which it is composed, so that each of these can be (preferably automatically) measured separately.

The outcome of the first stage is a thorough description of the scenario and the primary influence factors, and a completely specified requirements tree where every leaf node is assigned a measurable criterion. A knowledge base provides best-practice criteria catalogues that can be applied, further refined and re-inserted to the criteria catalogues, providing a feedback loop into the decision process. Typical trees contain between 50 and 150 requirements in about five hierarchy levels.

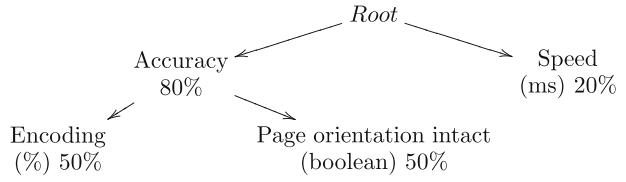


Fig. 2. Highly simplified requirements tree.

As a highly simplified example, consider the weighted example tree in Fig. 2, which contains three measurable criteria for converting text documents: correct encoding of characters measured in percent, the orientation of pages measured by a Boolean identity, and the speed of conversion, measured in milliseconds. The next section will describe the evaluation of these criteria, the calculation of utility functions, and the aggregated scoring and ranking of components.

3.2.2. Evaluation and analysis

Requirements evaluation takes advantage of the homogeneity of the problem space and follows an empirical approach. While some of the evaluation criteria may be retrieved from databases holding confirmed information about static attributes such as the price or the licensing model of candidates, specific quality criteria such as *accuracy* of operations or average *processing speed* for specific input data need to be evaluated in an evidence-based, empirical manner. Candidate tools are executed in a controlled environment, providing a thorough evaluation and evidence base in realistic experiments. A distributed web service infrastructure facilitates the discovery of potential tools and their invocation [7,11].

In principle, three categories of criteria need to be evaluated for each component:

1. Statically defined criteria are retrieved from a trusted database holding product information not subject to change.
2. Process- and performance-related characteristics can be measured automatically during tool execution in the experiments stage. The tools are invoked through a monitoring framework which is able to measure general process-related characteristics such as performance [9].
3. Criteria specific to the application domain, such as accuracy, are measured by an extensible architecture of measurement plugins, which is described in Section 3.3.

The evaluation of experiment results leads to a requirements tree fully populated with evaluation values in the respective scale of each criterion. This is the basis for the third phase, which corresponds to the GCS step *Analyse data and select product*.

Analysis of results consists of three steps:

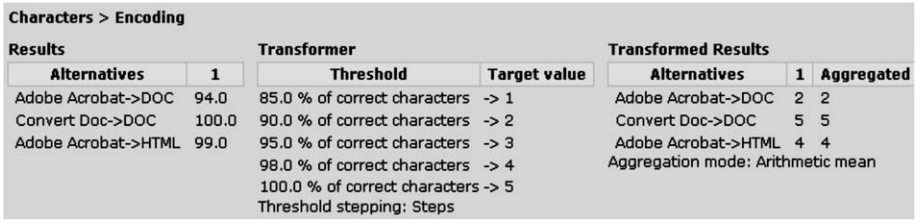
1. Transform values to a uniform scale,
2. Set importance factors, and
3. Analyse the outcomes to arrive at a candidate recommendation.

In order to apply aggregation and comparison of values over the tree hierarchy, a *utility function* is defined, which maps all evaluation values to a uniform target scale of commonly 0–5. Hereby, 5 is the optimum value, whereas 0 denotes unacceptable performance that serves as a drop-out criterion.

The utility function can be defined in a variety of ways. For numerical input values, we may rely on a simple linear transformation using threshold settings. Fig. 3 shows an example transformation setting for the attribute *Character encoding* with percentage values in the range of [0,100] and the corresponding transformation results. Using simple stepping, the resulting utility function $u(measure)$ in the example case is given in Eq. (1).

$$u(encoding) = \begin{cases} 0 & encoding < 85 \\ 1 & 85 \leqslant encoding < 90 \\ 2 & 90 \leqslant encoding < 95 \\ 3 & 95 \leqslant encoding < 98 \\ 4 & 98 \leqslant encoding < 100 \\ 5 & encoding = 100 \end{cases} \quad (1)$$

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001



Characters > Encoding

Results

Alternatives	1
Adobe Acrobat->DOC	94.0
Convert Doc->DOC	100.0
Adobe Acrobat->HTML	99.0

Transformer

Threshold	Target value
85.0 % of correct characters	-> 1
90.0 % of correct characters	-> 2
95.0 % of correct characters	-> 3
98.0 % of correct characters	-> 4
100.0 % of correct characters	-> 5

Threshold stepping: Steps

Transformed Results

Alternatives	1	Aggregated
Adobe Acrobat->DOC	2	2
Convert Doc->DOC	5	5
Adobe Acrobat->HTML	4	4

Aggregation mode: Arithmetic mean

Fig. 3. Transformation of evaluation results.

More commonly we use piecewise linear interpolation. Let v be the evaluation value of a candidate and t_i the list of thresholds in monotonically increasing order with $i = 1, 2, 3, 4, 5$. With the resulting breakpoints (t_i, i), the utility function $u(measure)$ is given in Eq. (2). For thresholds in decreasing order, the equation is adjusted accordingly

$$u(measure) = \begin{cases} 0 & \forall v < t_1 \\ i + \frac{v - t_i}{t_{i+1} - t_i} & \forall t_i \leqslant v < t_{i+1} \\ 5 & \forall v \geqslant t_5 \end{cases} \quad (2)$$

In the supporting software tool Plato, a default utility function is calculated on the basis of the measured values, assigning a value of 1 to the lowest and 5 to the highest measured value and using linear interpolation for the rest. This provides a useful starting point for setting the acceptable limits in a decision scenario. Other transformations include logarithmic and exponential interpolation.

Existing component selection methods usually favour the definition of target ranges for requirements before the actual values and capabilities of potential components are known. They rely on negotiation of conflicting values during package selection [16]. Transforming actually measured values after knowing the results allows trade-off decisions and negotiation of acceptable values based on a trustable knowledge of reality. In bid evaluation, the actual evaluation values should be anonymised to avoid decisions to be influenced by internal biases.

The result of the transformation step is a fully populated, evaluated and transformed requirements tree with default weighting. The next step is to revise these default weights to reflect the actual priorities and preferences of the stakeholders.

There has been considerable discussion on the question of importance weighting in component selection methods. Several methods using WSM have earned criticism for the fact that weight settings need to be specified upfront, in a situation where little or nothing is known about the actual performance and differences of candidate components. Furthermore, the reduction to a single number and the corresponding ranking is considered too simplistic by many. The AHP on the other hand is often considered too effort-intensive and complex, with the number of pairwise comparison exploding with the size of the requirements tree [53,60].

In the presented approach, relative importance factors are balanced on each level of the tree after evaluation values for all candidates are known and utility functions have been defined. This deviates from the standard Utility Analysis workflow, but has proven more useful in the considered selection scenario in numerous case studies. Again, as for the transformation settings, the candidates should be anonymised in bid evaluation scenarios. In order to safeguard against potentially negative effects of minor variations in the weighting on the stability of decisions, a sensitivity analysis is performed. In this step, several hundred iterations are automatically computed, randomly varying weights in a certain percentage margin around the given value to identify potential changes in the ranking of components. We are currently investigating the narrowly focused use of robust, but effort-intensive ranking models such as AHP for ranking a small number of critical high-level factors as well as competing factors that show high sensitivity during the analysis. Fig. 4 shows the supporting software tool balancing the weights of criteria.

In the final step, visual analysis of results allows a comparison of performance values not only on the root level, but on all levels of the tree hierarchy. Several aggregation methods are supported, of which the most relevant are weighted multiplication and weighted sum. The score of each node n_i is determined by the weighted score of its k children c_j, given that the relative weight of all its children sums up to 1. The score of the leaf nodes is provided by the utility function $u(measure)$ with the measured evaluation value as input variable.

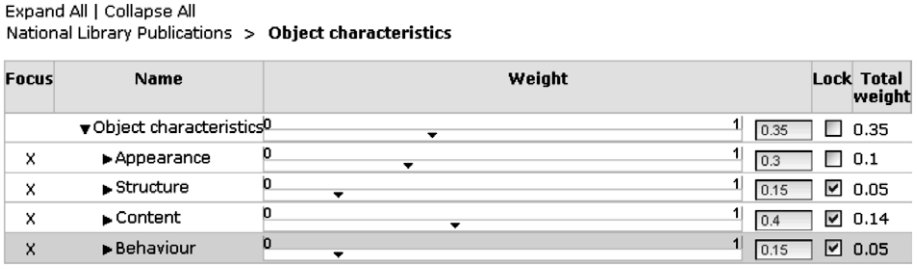
For both sum and multiplication, we seek a weighted aggregation function where nodes with a weight of 0 result in a neutral element and the target range is the same as the input range. For weighted sum, this is the well-known linear combination calculating a weighted score as given in Eq. (3)

$$u^s(n_i) = \sum_{j=1}^{k} w(c_j) u^s(c_j) \quad (3)$$

For weighted multiplication, the values are taken to the power of the weight, as given in Eq. (4)

$$u^m(n_i) = \prod_{j=1}^{k} u^m(c_j)^{w(c_j)} \quad (4)$$

The aggregated scoring obtained thereby serves for filtering out candidate components with unacceptable evaluation values – any



Expand All | Collapse All

National Library Publications > **Object characteristics**

Focus	Name	Weight		Lock	Total weight
	▼Object characteristics	0 — 1	0.35	☐	0.35
X	▶Appearance	0 — 1	0.3	☐	0.1
X	▶Structure	0 — 1	0.15	☑	0.05
X	▶Content	0 — 1	0.4	☑	0.14
X	▶Behaviour	0 — 1	0.15	☑	0.05

Fig. 4. Setting importance factors.

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

score of 0 at the criterion level will be reflected as a root score of 0, allowing the decision maker to quickly analyse the root cause of the drop-out. For the remaining alternatives, we generally use the weighted sum aggregation provided in Eq. (3) to find the candidate component best suited for the given scenario.

We use the transformation function of Eq. (2) and the thresholds defined in Eq. (1) to evaluate the tree defined in Fig. 2. We set the thresholds for *speed* to be {1500,1000,750,500,250} in decreasing order and define the loss of page orientation as unacceptable, resulting in Eq. (5)

$$u(orientation) = \begin{cases} 5 & \text{if true,} \\ 0 & \text{if false.} \end{cases} \quad (5)$$

Table 2 lists evaluation values and their respective utility. For example, Tool A has a measured speed of 600 ms, which translates to a utility value of 3.6. Figs. 5 and 6 illustrate the aggregation of values, showing the weighted aggregated value on each node to illustrate the absolute influence towards the aggregated parent value.

Tool C is being eliminated because it breaks the page orientation during conversion, which leads to a utility of 0 for *orientation*, as shown in Fig. 5. The multiplication has the effect of rejecting the component with a root score of 0, which can be traced back to the criteria responsible for this by following the graph to the leaf level.

Fig. 6 shows weighted sum results for Tool B, which does not exhibit unacceptable measures. We thus obtain the aggregated weighted result scores provided in Table 3. Notice that while Tool A is faster than Tool B, the superior evaluation for *encoding* of Tool B is weighted stronger and determines the final ranking.

Table 2
Example evaluation results and utility values.

Component	Encoding	Orientation	Speed
Tool A	98%	*True*	600 ms
Tool B	100%	*True*	800 ms
Tool C	100%	*False*	500 ms
Tool A	4	5	3.6
Tool B	5	5	2.8
Tool C	5	0	4

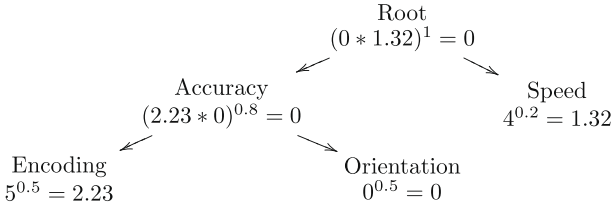


Fig. 5. Weighted multiplication results for Tool C.

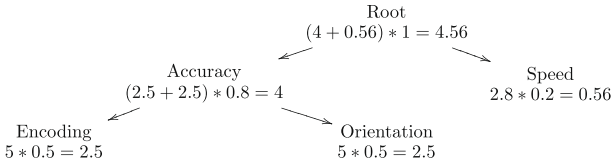


Fig. 6. Weighted sum results for Tool B.

Table 3
Aggregated values.

Component	Weighted multiplication	Weighted sum
Tool A	4.28	4.32
Tool B	4.45	4.56
Tool C	0	–

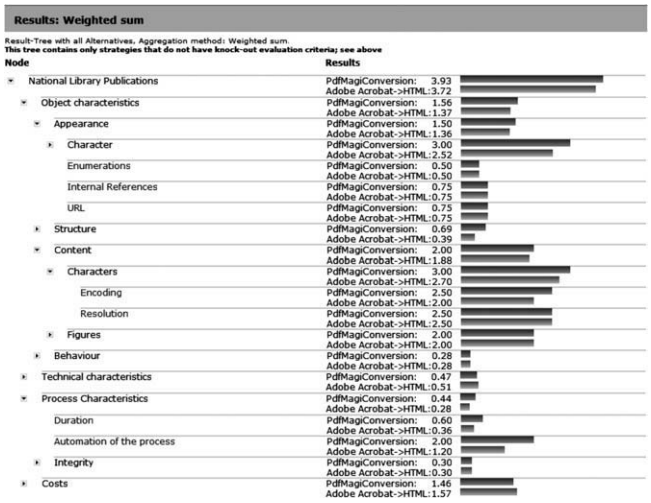


Fig. 7. Visualisation of results.

The supporting software tool Plato allows interactive exploration of the tree hierarchy as shown in Fig. 7. At the leaf level, all measured values and utility functions are linked and accessible to the decision maker. The definition of acceptance criteria in the utility function further provides a gap analysis which clearly points out both strengths and limitations of candidates on all levels of the hierarchy. Plato thus supports an in-depth analysis of the specific strengths and weaknesses of each candidate component and an evidence-based decision for a specific component, which is the outcome of the third stage. The complete evaluation, transformation and aggregation is used as an evidence base for supporting the decision for recommending one of the candidate components. The method furthermore allows for the selection of multiple components that are considered to be complementary, in case none of the alternatives alone completely satisfy the needs to a sufficient degree.

3.2.3. Integration and monitoring

Integration is the final stage of the workflow, where integration procedures define the component's interfaces to the system under consideration and explicitly state evaluation conditions that need to be monitored continuously during operation of the chosen software. These conditions can be largely deducted from the requirements and defined in service-level agreements [42].

Both general Quality-of-Service (QoS) attributes such as performance, throughput, resource usage, etc., as well as specific QoS criteria such as accuracy need to be continuously monitored during operation to ensure that the used component actually keeps fulfilling the requirements as expected [9]. Any deviation in QoS of the level measured during the experiments is an indication of either an incomplete evaluation procedure, or a change in the environment that needs to be addressed, such as a sudden increase in data volume. Depending on the type and severity of the deviation, this may lead to a re-iteration of the evaluation procedure, where the original scenario is taken as a starting point and revised according to changes in the environment, technology, or the requirements.

The problem of *architectural mismatches* is often encountered in CBSD, primarily when dealing with coarse-grained components where controlled experimentation is rarely applicable. Selecting a seemingly optimal component based on an analysis that focuses

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

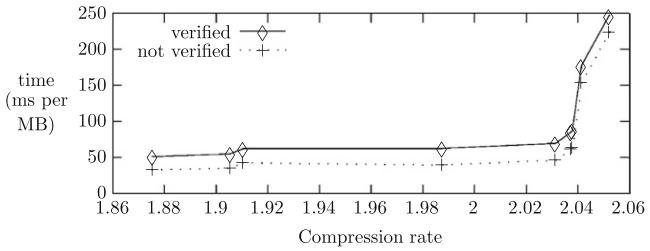


Fig. 8. QoS trade-off between compression rate and performance.

on functional issues can lead to serious cost- and budget overruns when implicit architectural assumptions prove to be incompatible and conflicting [26].

Since our approach does not lead to the recommendation of a component without fully testing it in a realistic controlled environment, this aspect is less likely to have an effect; mismatches are prevented or detected early [25]. If a component can be successfully evaluated in a controlled environment on realistic conditions and real data, the risk of a serious mismatch on either an architectural or a lower technical level is quite limited; furthermore, platform constraints, etc. are part of the evaluation procedure. If a component cannot be evaluated, this is a documented outcome of an experiment and will in the absence of a re-run experiment lead to a rejection of the component. The method and tool allow feedback loops, should experimentation lead to the discovery of such problems. Evaluators then return to earlier stages to refine requirements, reconsider acceptable values, or re-balance importance factors. For example, if it turns out the optimally performing tool is incompatible with a server environment or protocol or simply too slow, but all other components produce unacceptable results quality-wise, it may be necessary to reconsider performance requirements or reduce expectations in terms of quality.

The accumulated experience from the knowledge base can be compared to specific values obtained in the experiments. It can furthermore inform the definition of monitoring conditions during the final integration stage. Fig. 8 shows a resulting trade-off curve between compression rate and execution speed for an audio conversion program with and without internal verification of the converted output. Verification adds considerable overhead, but this might be quite acceptable in many cases. On the other hand, the highest compression settings achieve minimal benefits in terms of compression rate, but are computationally very expensive. These observations can be used to derive optimised parameter settings for deployment.

3.3. Software architecture

To support the method described in the previous section, we have been developing distributed tool support to enable guided decision making and automate the task of measurements [7,11]. The resulting software, a planning tool called *Plato*, is targeted at the digital preservation community and freely available online. While the frontend itself has been developed in the context of digital preservation, the evaluation, selection and monitoring architecture is independent of the target domain. We are currently working towards releasing a generalised, domain independent version of the web frontend for the public.

Fig. 9 shows the schematics of the core architectural building blocks. A knowledge base layer holds information about existing reusable quality criteria, associated measurements and the modules where these measures can be obtained, as well as accumulated quality measurements on specific components and classes of components. This knowledge base is a collaborative distributed environment where users can share and reuse quality models. Current work in the PLANETS project[2] is extending this knowledge base by analysing and abstracting requirements trees from case studies and making them reusable across organisations. The requirements catalogues primarily focus on *accuracy* criteria, which are the central consideration, but also contain detailed models of the risks associated with certain outcomes of processing data with the candidate components.

The evaluation and monitoring core covers the concepts of the last section, while flexible adaptation layers enable the dynamic integration of components discovery and invocation. The discovery of candidate preservation action components is facilitated through web service registries, filtering components by the object types they can operate on. For instance, in the case of the Word document collection, potential conversion tools are listed. These tools are located in different technical environments, but can be accessed through wrappers in a uniform way. The candidate tools are applied to a previously defined set of sample objects, and the outcome is evaluated against the requirements, relying on automated measurements. That means that all considered components are executed in their respective deployment location inside a controlled environment that monitors their behaviour and collects data about their resource utilisation [9]. This meta-information on the service execution is delivered along with the component output and mapped semi-automatically to the specified requirements.

On a general level, performance-related criteria are often considered as a requirement, especially for large-scale systems. Our extensible monitoring approach allows tools to be invoked through a service-oriented architecture, where quality information measured during program execution is delivered as meta-information on the service execution [9]. Measurement adaptors currently available include an extensible monitoring framework for QoS measurement of arbitrary tools wrapped as web services, and domain-specific accuracy measurement for file conversions in digital preservation [12]. While in some cases, manual intervention is still required, the objective is to fully cover the measurements needed to evaluate the requirements.

The next section will illustrate how this architecture supports the concrete evaluation procedures and measurements in a specific digital preservation scenario.

4. Using automated measurements to evaluate migration components

This section describes the influence factors to be taken into account when evaluating migration components for a digital repository, discusses the measurement of domain-specific quality criteria and presents a taxonomy of criteria in digital preservation. This is used as a basis for a discussion on automated measurements and can be applied similarly in other domains.

4.1. Requirements definition

The high-level influence factors that have to be taken into account when deciding which component provides the best fit for a particular digital preservation solution include

1. *Technologies and standards.* Compliance to file format and metadata standards is a prime concern, and on a technical level, compatibility of candidates to existing infrastructures usually a non-negotiable requirement.

[2] http://www.planets-project.eu.

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

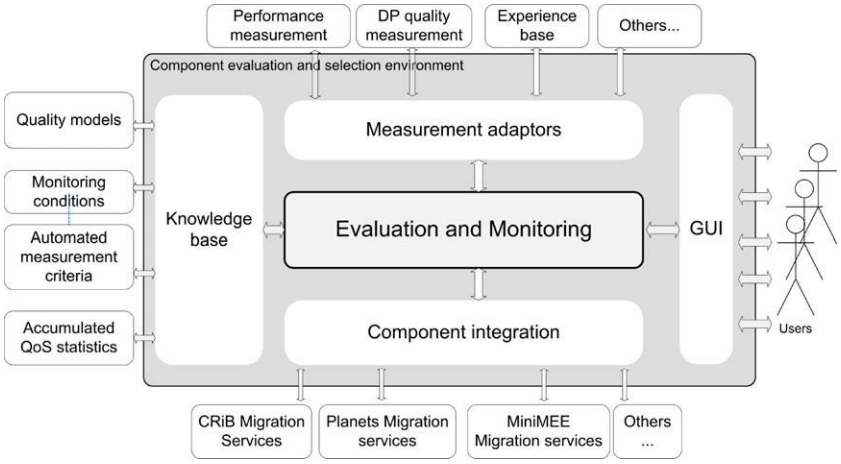


Fig. 9. Integration architecture.

2. *User requirements* include desired modes of accessing digital content in a repository and file formats supported by a certain group.
3. *Object characteristics* refer to the central concern of *accuracy* stating that digital content must be preserved as accurately as possible, i.e. the authenticity has to be secured.
4. *Legal constraints and policies* may prohibit the use of certain technologies or formats.
5. *Organisational constraints* may pose restrictions on prices, storage technologies, or procedures.

Some of these factors, such as institutional policy models or business constraints, may partially be modelled in a formal way, thus allowing automatic reasoning and derivation of environment constraints that can support the evaluation procedure. Others, such as object characteristics, are semi-automatically constructed by analysing the properties of the digital objects on which the candidate software components need to operate and by specifying which characteristics need to be preserved.

The requirements specification breaks the high-level goals down to quantifiable criteria. Abstract goals such as *accurately preserve the significant properties of objects* are iteratively refined until a level of measurable properties is reached. Thus, at the bottom level of a typical requirements tree in digital preservation, accuracy-related criteria can be found such as *keep correct encoding of characters* or *keep image width unchanged*. These criteria are *empirical* in the sense that they are to be measured during the experiments stage. In contrast, aspects such as *costs per object* can be retrieved from a database and do not need to be measured. More complex object criteria arise for example in the context of web archiving tools. Consider the visitor counter on a web page – when transforming the representation of the web page for preservation purposes, a tool could leave the counter active, i.e. each subsequent access in the repository changes its value; it could remove the counter; or it could *freeze* it at the number it showed when it was harvested.

These object criteria are generally called *significant properties* of digital objects and also referred to as *object characteristics* [19]. They are often classified in five high-level aspects [34,66].

1. *Content* means the actual content of an object, such as the words in a document.
2. *Appearance* denotes visual and audible characteristics such as the layout.
3. The *Structure* of an object describes the relationships of its subcomponents.

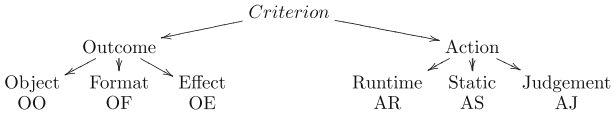


Fig. 10. Taxonomy of criteria in digital preservation.

4. *Behaviour* describes the interactive experience a user has when viewing the object, and
5. The *Context* refers to relationships to other digital objects.

ISO SQuaRE suggests standardising domain-specific extensions elaborating on certain aspects of the quality model. Recent efforts such as INSPECT[3] strive to move in that direction. PLANETS is modelling organisational constraints and developing an ontology for significant properties of digital objects, for which the XCL languages provide the technical starting point [12].

These characteristics are obviously highly domain specific. In machine translation, for example, a combination of comparison metrics obtained by established algorithms such as BLEU [59] or others [47,20] can be used for the measurements.

4.2. A taxonomy of criteria

To design and evaluate a full-coverage measurement framework for the target domain of digital preservation, we have created a taxonomy of criteria that differ in the information sources they depend on to obtain measurements. This taxonomy is depicted in Fig. 10. Fundamentally, all criteria requiring measurement refer either to the action, i.e. the component, or the outcome of an action, that is a rendering or transformation of a digital object. These two categories can be further distinguished to describe general effects of the outcome, such as certain properties of object that are desired; criteria describing the format of the objects; and criteria describing the abovementioned significant properties. On the other hand, action components exhibit properties that are static and descriptive in nature, properties that can be measured at runtime, and some properties that need to be judged by experts.

The taxonomy is in principle orthogonal to the structure of our hierarchy of goals, i.e. an evaluation goal might be connected to measurable characteristics of different categories. For instance, the general goal of inducing minimal costs may include both the

[3] http://www.significantproperties.org.uk/.

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

price per object, i.e. per execution, of a component, and runtime characteristics such as resource utilisation that imply a certain level of hardware expenditures.

We have thus identified the following categories:

1. Properties of the outcome of applying a component.
   (a) *Object*. This category entails all properties of digital objects that result from the application of a preservation action component on these objects. On the one hand, this includes desirable properties; on the other hand, this category refers to all significant properties that have to be kept intact. Measurements have to be applied to the outcome of the preservation action components' application and might include comparing the outcome to the original objects.
   (b) *Format*. This category comprises criteria on desirable characteristics of the file and data formats digital content shall be represented in. This is a central decision criterion in digital preservation since a significant portion of the risks to digital content lies in the form of representation and its understandability. Measurements of these criteria are applied by analysing the format of the outcome and getting additional information on known properties of certain formats from trusted external data sources such as the Pronom Technical Registry [74].
   (c) *Effect of outcome*. This refers to any other effects caused by the application of a certain component, such as the storage costs resulting from converting to certain formats with higher compression. Typically, these effects are calculated based on measured inputs. For costs, recognised cost models such as LIFE [5] are used with measured properties as input parameters.

2. Properties of the components, i.e. the action taken.
   (a) *Runtime*. This category entails runtime properties of a component such as performance and resource utilisation. Measurements need to be taken in a controlled environment.
   (b) *Static*. Criteria of this category can be obtained from trusted sources. Where not found, they need to be evaluated manually with appropriate documentation.
   (c) *Judgement*. This category is sometimes relevant, but should be kept to a minimum. Usability is a prime example where judgement is necessary. In digital preservation this is hardly relevant since the components to be evaluated are not to be applied by an end user. In other cases, this has more importance. In any case, proper documentation of evaluation values is essential.

In machine translation, a taxonomy would likely be very similar apart from the *format* category, which still may be present, comprising structural characteristics of the resulting translation following a standard representation. Object criteria may further include the presence of confidence values or alternative translation candidates for certain terms.

4.3. Automated evaluation in digital preservation

The automated evaluation of criteria depends on the category according to the taxonomy outlined above. The measurement of process-related criteria such as performance can be covered with a generic approach. On a domain-specific level, more complex measurements may be needed to cover the desired characteristics, particularly at the level of significant properties. To evaluate these characteristics automatically, thorough description of digital objects are needed to enable automatic reasoning and comparison. This *characterisation* relies on formal languages such as the eXtensible Characterisation Languages [12]. Characterisation of the status quo takes place when defining the characteristics of the collection of digital objects on which the component to be selected is going to operate. Thus it supports the automated characterisation of digital objects, i.e. the extraction of significant properties into an abstract representation that can then be used for empirical evaluation.

Fig. 11 shows a simplified abstraction of the core elements of the requirements and evaluation model. This model is closely related to the taxonomy described above.

Each `PreservationActionTool` (or `MachineTranslationTool`) is evaluated through applying it on `SampleObjects` in a controlled experiment. This creates an `ExperimentResult` that constitutes part of the evidence base. A `Criterion` is a measurable `Requirement`. It can be associated with a tool (`ToolCriterion`) or vary with every object a tool is applied to (`ObjectCriterion`). In the latter case, it can be mapped to an `ObjectProperty`. These properties are measured of the original `SampleObject` and the `ExperimentResult`, and the obtained values are compared through a comparison metric. Tool criteria, on the other hand, are associated with a `ToolProperty` and evaluated in a `ToolEvaluation`. The properties themselves may be partly obtained from registries; others can be derived from software quality models, while some may need to be specified explicitly.

A typical example of a criterion that will be measured at runtime and vary with every experiment is execution speed. We have conducted a series of experiments to evaluate different aspects of the evaluation and monitoring architecture, such as the correlation and accuracy of measurements obtained with different profiling tools [9]. Fig. 12 shows the processing time of two conversion tools

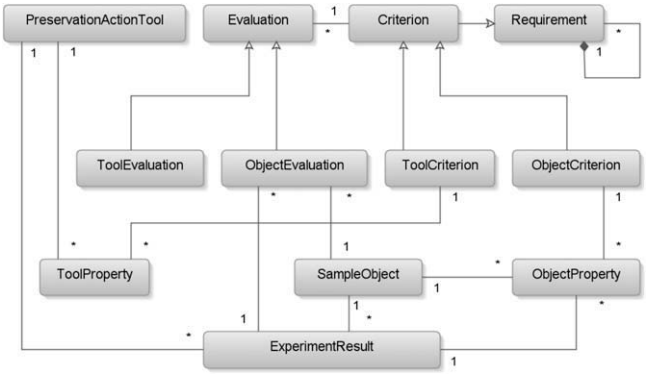


Fig. 11. Requirements, criteria and evaluation in digital preservation.

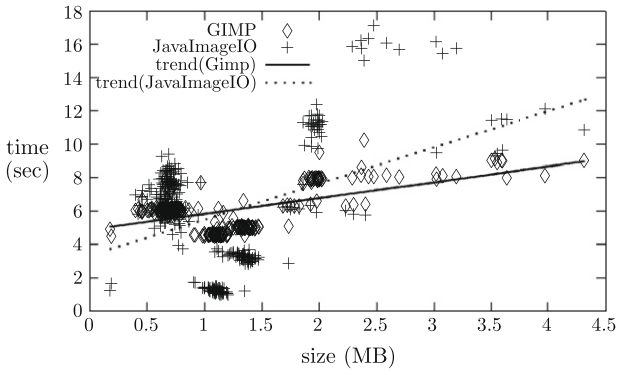


Fig. 12. Processing speed of two conversion services.

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

Fig. 13. Visualisation of an examplary conversion error.

running on the same environment when converting 312 JPEG image files. Linear regression shows the general trend of the performance relation, with a root mean squared of residuals of 1.01 for GIMP and 3.36 for JavaImageIO. The Java tool is faster on smaller images but outperformed on larger files. It further is noted that the *conversion quality* offered by GIMP is certainly higher: Fig. 13 shows a visualisation of a conversion error introduced by the simple Java program when converting an image with transparent background from GIF to JPG. Changed pixels are shown in grey in the figure and indicate that the transparency layer has been lost during migration. This difference is being measured by quality evaluators relying on different, complementary tools. For example, ImageMagick *compare* reports that 59.27% of the pixels are different in the migrated file, with a root mean squared error of 24,034. In most cases in digital preservation, the information loss introduced by a faster tool will be considered much more important than its speed, which will be reflected by the utility function.

4.4. Pulling it all together

The output of the evaluation stage is a well-founded, documented recommendation for one of the components. This documentation is considered essential for ensuring the trustworthiness of a digital repository [1]. In some cases, however, a single component alone might not be satisfying. For example, many conversion tools for electronic documents have problems with entirely preserving the layout as it was displayed in the original environment, whereas migrating a document to an image loses the future potential for full-text search access. In these cases it may thus be desirable to combine both approaches and select multiple components for the incorporation into a preservation system.

The exact mode of integrating the chosen component depends on the infrastructure in question. The planning tool Plato can automatically generate an XML-based workflow specification called *executable preservation plan*, which specifies the component to be used and all parameter settings. It can be deployed directly into the Planets Interoperability Framework that provides an infrastructure for digital preservation services [43,68]. We are currently extending Plato with modules for generating executable preservation plans for different widely used repository systems.

The evidence gathered during the controlled experiments is highly valuable for defining service-level agreements and monitoring conditions that can be watched continuously after integration of the preservation action component into a digital repository. A variety of factors in the technical and organisational environment as well as the repository itself have to be monitored to ensure continued successful operation. Environmental conditions are understood in the broad sense and cover factors such as the rate of incoming content or the distribution of file formats that are contained in an archive. On the concrete QoS level, QA mechanisms based on structured characterisation of digital objects as described above can be included, while on an organisational level, changing high-level policy decisions may alter the priorities or introduce new requirements. An automated re-evaluation and threshold check can trigger an alert that leads to a re-iteration of the evaluation procedure as described in [8].

5. Evaluation

The last sections described a framework and tool for improving component evaluation and selection in specific scenarios by using controlled experimentation and automated measurements as primary means of data collection. We described a taxonomy and model of criteria and their automated evaluation.

We noted earlier that evidence is an essential precursor to trustworthiness, and that an entity's trustworthiness has to be evaluated in the realistic context of an action. Thorough documentation is needed to ensure reproducibility of evaluation experiments. One of the primary benefits of automated measurements is the degree of evidence and documentation that is produced along with the evaluation. We claim that under our stated conditions, these benefits can be achieved for a large fraction of the decision criteria.

We evaluate this by quantitatively assessing the coverage of automated measurements with respect to criteria used in real-world decisions. To answer the questions posed in Section 2, we analyse a number of case studies that have been carried out during the last years with and without supervision and assistance. Table 4 provides an overview of cases. All were searching for an optimal conversion tool for preserving different types of images or documents in large repositories run by organisations such as national libraries, national archives, or large research foundations. Detailed discussions of several of these case studies can be found in [8,46,71]. While most of the studies were carried out with our assistance, two of them were carried out independently without consultation, using the publicly available software platform which is increasingly being used by libraries and archives for decision support.

The effort to conduct an evaluation depends on the types of digital objects, the level of automated measurements that is available, and the experience an organisation has with digital preservation in general and the decision making procedures in particular. Documentation of organisational policies is often scarce; this is beginning to change [69]. Often, the first time evaluation and selection is carried out, a number of organisational factors have to be established once, which adds to the effort needed for the first decision process. As a general rule, a selection procedure will take a few days, involving several stakeholders for the requirements definition.

5.1. Coverage of automated measurements

The criteria in Table 4 correspond to the categories of the taxonomy described in Section 4.2. For each case study in the list, we provide the type of institution taking the decision, the type of content in need of conversion, and number of decision criteria falling into each category. The bottom row summarises the distribution of the criteria. Of the 317 criteria that had to be evaluated, all fall into one of the categories of the taxonomy. More than 50% describe the *accuracy* of preserving the authenticity or significant properties of objects, while another 13.6% refer to desired characteristics of formats resulting from the application of components. Of the requirements on the components, their measurable runtime behaviour accounts for almost 20% of the criteria. All of these characteristics can be measured in a fully automated way as outlined in Section 4.3 and discussed in detail elsewhere [9,10,12].

This leaves 5% of criteria that fall into the categories *judgement of tool characteristics* and 3.8% of criteria that refer to general effects

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

Table 4
Distribution of criteria in case studies on file conversion.

Nr.	Type	Institution type	Supervised	Object format	Total	OO	OF	OE	AR	AS	AJ
1	Documents	Library	Yes	PDF	44	27		2	1	10	4
2	Documents	Library	Yes	PDF	33	19			4	8	2
3	Images	Library	Yes	TIFF-5	24	8	6	1	3	3	3
4	Images	Library	Yes	GIF	28	5	3	3	3	13	1
5	Images	Library	Yes	TIFF-6	40	10	12	1	3	10	4
6	Documents	Research inst.	No	PDF	47	22	12	2		10	1
7	Images	Library	Yes	Various	33	18	10	2	1	1	1
8	Documents	Archive	Yes	Word perfect 5×	38	35				1	2
9	Documents	Library	No	Various	30	20		1	1	7	1
					317	164	43	12	16	63	19
					100%	51.7%	13.6%	3.8%	19.9%	6%	5%

Table 5
Categories, examples and data collection methods.

Category	Abbr.	Percentage	Example	Data collection and measurements
Outcome object	OO	51.7	*Image pixelwise identical* (RMSE)	Measurements of input and output
Outcome format	OF	13.6	*Format is ISO standardised* (boolean)	Measurements of output, trusted external data sources
Outcome effect	OE	3.8	*Annual bitstream preservation costs* (€)	Measurements of output, trusted external data sources, models, partly manual calculation and validation, sharing
Action runtime	AR	19.9	*Throughput* (MB per ms)	Measurements taken in controlled experimentation
Action static	AS	6	*Price per CPU* (€)	Trusted external data sources, manual evaluation and validation, sharing
Action judgement	AJ	5	*Documentation about activity* (excellent, sufficient, poor)	Manual judgement, sharing

of outcomes, some of which have to be evaluated and calculated in a manual way.

Table 5 summarises the taxonomy's categories, mapping abbreviations of Table 4 to the corresponding terms and providing examples as well as the information sources needed for evaluation. For most of the criteria, there exist either trusted information sources or known means to automatically collect measurements to evaluate them. To improve the reliability of the remaining criteria, we are extending the evaluation platform to enable experience sharing and provision of aggregate statistics about such judgements. This sharing also benefits aggregated statistics of measurements taken in the controlled environment on different input data and will lead to a collaborative benchmarking platform.

5.2. Threats to validity

As noted above, there is still a certain percentage of criteria that cannot be measured automatically and have to be judged by experts. This judgement naturally entails the risk of not being reproducible and exhibiting certain biases. The sensitivity analysis described in Section 3.2.2 addresses this issue, but cannot provide guarantees. The usage of AHP is highly desirable for these criteria and currently in development.

To ensure measurement reliability, the digital preservation domain is investing significant efforts in defining benchmarking corpora and criteria for stratification of test data [56]. Annotated benchmark data will provide the means for validating measurement accuracy of quality assurance tools such as the XCL languages [12].

6. Discussion and outlook

This paper outlined an evidence-based software evaluation and selection framework which relies on controlled experimentation and empirically evaluates candidate components in controlled experiments. The procedure follows the general COTS selection process and tailors the concept of utility analysis to allow quantified evaluation of components against hierarchically defined requirements.

We showed that in specific scenarios, a majority of the criteria can be evaluated by applying automated measurements in a controlled environment under realistic conditions. This not only reduces the effort needed to evaluate components, but also supports trust in the decisions because extensive evidence is produced in a repeatable and reproducible way and documented along with the decision in a standardised and comparable form.

While the supporting software tool and the case studies are domain specific, we believe the approach is applicable and useful in a wider context. We are currently investigating its application to the

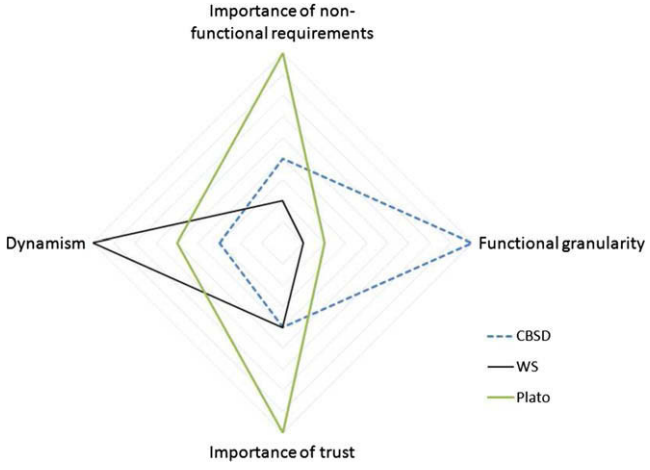


Fig. 14. Comparison of foci in component selection scenarios.

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001

automated evaluation of natural language translation software [41].

Component selection and evaluation is a continuous problem space ranging from CBSD to web services and other dynamic scenarios as summarised in Table 1 in Section 2. Fig. 14 contrasts in a simplified perspective three scenarios for component selection: Web service selection on the dynamic end of the spectrum is highly dynamic, deals with fine-grained and precisely specified components, and relies entirely on automated evaluation. CBSD on the other end deals with any granularity of functions and is less dynamic. The degree of trust needed and the importance of NFRs certainly vary in each case, but are in general not as high as for example in digital preservation. With our method and the planning tool Plato, the selection and integration is followed by continuous monitoring. Deviations from specified expectations lead to a re-evaluation that can result in a reconfiguration or replacement of the component. The method is applicable to fine-grained components with a homogeneous, well-defined feature set, and the focus lies on automated measurements obtained in a controlled environment.

Current work is geared towards the automatic derivation of requirements from modelled constraints and environment factors; tracing and monitoring of influence factors and continuous impact assessment; and recommender systems for eliciting key influence factors and suggesting sets of requirements from the knowledge base. We are further investigating the usage of established quality model construction schemes such as [16,24] to formalise and further standardise best-practice subcharacteristics, attributes and metrics and describe correlation-free sets of measures for specific scenarios that recur often in digital preservation, based on our experiences with accuracy and risk criteria in the case studies cited above. In parallel, we are generalising and extending tool support to other domains and conduct case studies evaluating software components in different domains such as language translation software, with a focus on automated measurements.

Acknowledgements

Part of this work was supported by the European Union in the 6th Framework Program, IST, through the PLANETS project, Contract 033789.

References

[1] Trustworthy Repositories Audit & Certification (TRAC): Criteria and Checklist, Center for Research Libraries, OCLC Online Computer Library Center, 2007. http://www.crl.edu/PDF/trac.pdf.
[2] Incommensurable values, Stanford Encyclopedia of Philosophy, May 2009. http://plato.stanford.edu/entries/value-incommensurable/.
[3] Carina Alves, COTS-based requirements engineering, in: Component-based Software Quality, LNCS, vol. 2693, Springer, Berlin, Heidelberg, 2003, pp. 21–39.
[4] Carina Alves, Jaelson Castro, CRE: a systematic method for COTS components selection, in: XV Brazilian Symposium on Software Engineering (SBES), Rio de Janeiro, Brazil, 2001.
[5] P. Ayris, R. Davies, R. McLeod, R. Miao, H. Shenton, P. Wheatley, The LIFE2 final project report, LIFE Project, 2008. http://eprints.ucl.ac.uk/11758/.
[6] Paul Baker, Mark Harman, Kathleen Steinhfel, Alexandros Skaliotis, Search based approaches to component selection and prioritization for the next release problem, in: Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM '06), IEEE Computer Society, 2006, pp. 176–185.
[7] Christoph Becker, Miguel Ferreira, Michael Kraxner, Andreas Rauber, Ana Alice Baptista, JosT Carlos Ramalho, Distributed preservation services: integrating planning and actions, in: Birte Christensen-Dalsgaard, Donatella Castelli, Bolette Ammitzbll Jurik, Joan Lippincott (Eds.), Research and Advanced Technology for Digital Libraries, in: Proceedings of the 12th European Conference on Digital Libraries (ECDL'08), LNCS, vol. 5173, Lecture Notes in Computer Science, Aarhus, Denmark, September 14–19, Springer, Berlin/Heidelberg, 2008, pp. 25–36.
[8] Christoph Becker, Hannes Kulovits, Mark Guttenbrunner, Stephan Strodl, Andreas Rauber, Hans Hofman, Systematic planning for digital preservation: evaluating potential strategies and building preservation plans, International Journal on Digital Libraries (IJDL), December 2009. http://dx.doi.org/10.1007/s00799-009-0057-1.
[9] Christoph Becker, Hannes Kulovits, Michael Kraxner, Riccardo Gottardi, Andreas Rauber, An extensible monitoring framework for measuring and evaluating tool performance in a service-oriented architecture, in: Proceedings of the 9th International Conference on Web Engineering (ICWE'09), LNCS, vol. 5648, Springer, 2009.
[10] Christoph Becker, Hannes Kulovits, Michael Kraxner, Riccardo Gottardi, Andreas Rauber, Randolph Welte, Adding quality-awareness to evaluate migration web-services and remote emulation for digital preservation, in: Maristella Agosti, Jose Borbinha, Sarantos Kapidakis, Christos Papatheodorou, Giannis Tsakonas (Eds.), Research and Advanced Technology for Digital Libraries, in: Proceedings of the 13th European Conference on Digital Libraries (ECDL 2009), LNCS, vol. 5714, Springer, September 2009, pp. 39–50.
[11] Christoph Becker, Hannes Kulovits, Andreas Rauber, Hans Hofman, Plato: a service oriented decision support system for preservation planning, in: Proceedings of the 8th ACM IEEE Joint Conference on Digital Libraries (JCDL'08), 2008.
[12] Christoph Becker, Andreas Rauber, Volker Heydegger, Jan Schnasse, Manfred Thaller, Systematic characterisation of objects in digital preservation: the extensible characterisation languages, Journal of Universal Computer Science 14 (18) (2008) 2936–2952. http://www.jucs.org/jucs_14_18/systematic_characterisation_of_objects.
[13] Christoph Becker, Stephan Strodl, Robert Neumayer, Andreas Rauber, Eleonora Nicchiarelli Bettelli, Max Kaiser, Long-term preservation of electronic theses and dissertations: a case study in preservation planning, in: Proceedings of the 9th Russian Conference on Digital Libraries (RCDL 2007), Pereslavl, Russia, October 2007. http://rcdl2007.pereslavl.ru/en/program.shtml.
[14] Matt Blaze, Sampath Kannan, Insup Lee, Oleg Sokolsky, Jonathan M. Smith, Angelos D. Keromytis, Wenke Lee, Dynamic trust management, IEEE Computer 42 (2) (2009) 44–52.
[15] Juan P. Carvallo, Xavier Franch, Carme Quer, Requirements engineering for COTS-based software systems, in: Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC'08), ACM, New York, NY, USA, 2008, pp. 638–644.
[16] Juan Pablo Carvallo, Xavier Franch, Carme Quer, Determining criteria for selecting software components: lessons learned, IEEE Software 24 (3) (2007) 84–94.
[17] Alejandra Cechich, Mario Piattini, Antonio Vallecillo (Eds.), Component-Based Software Quality, Springer, 2003.
[18] Vittorio Cortellessa, Ivica Crnkovic, Fabrizio Marinelli, Pasqualina Potena, Experimenting the automated selection of cots components based on cost and system requirements, Journal of Universal Computer Science 14 (8) (2008) 1228–1255. http://www.jucs.org/jucs_14_8/experimenting_the_automated_selection.
[19] Angela Dappert, Adam Farquhar, Significance is in the eye of the stakeholder, in: Research and Advanced Technology for Digital Libraries. Proceedings of the 13th European Conference on Digital Libraries (ECDL 2009), 2009.
[20] George Doddington, Automatic evaluation of machine translation quality using n-gram co-occurrence statistics, in: Proceedings of the Second International Conference on Human Language Technology Research, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2002, pp. 138–145.
[21] Schahram Dustdar, Wolfgang Schreiner, A survey on web services composition, International Journal of Web and Grid Services 1 (2005) 1–30.
[22] Abdelkarim Erradi, Piyush Maheshwari, Vladimir Tosic, Ws-policy based monitoring of composite web services, in: Proceedings of the Fifth European Conference on Web Services (ECOWS'07), IEEE Computer Society, Washington, DC, USA, 2007, pp. 99–108.
[23] Miguel Ferreira, Ana Alice Baptista, Jose Carlos Ramalho, An intelligent decision support system for digital preservation, International Journal on Digital Libraries 6 (4) (2007) 295–304.
[24] X. Franch, J.P. Carvallo, Using quality models in software package selection, IEEE Software 20 (1) (2003) 34–41.
[25] D. Garlan, R. Allen, J. Ockerbloom, Architectural mismatch: why reuse is still so hard, IEEE Software 26 (4) (2009) 66–69.
[26] David Garlan, Robert Allen, John Ockerbloom, Architectural mismatch or why it's hard to build systems out of existing parts, in: International Conference on Software Engineering, 1995, 0:179.
[27] A.J. Gilliland-Swetland, P.B. Eppard, Preserving the authenticity of contingent digital objects: the InterPARES project, D-Lib Magazine 6 (7/8), 2000. http://www.dlib.org/dlib/july00/eppard/07eppard.html.
[28] Gemma Grau, Juan Pablo Carvallo, Xavier Franch, Carme Quer, DesCOTS: a software system for selecting COTS components, in: Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04), IEEE Computer Society, Washington, DC, USA, 2004, pp. 118–126.
[29] Mark Guttenbrunner, Christoph Becker, Andreas Rauber, Evaluating strategies for the preservation of console video games, in: Proceedings of the Fifth International Conference on Preservation of Digital Objects (iPRES 2008), London, UK, September 2008.
[30] Mark Harman, Bryan F. Jones, Search-based software engineering, Information and Software Technology 43 (14) (2001) 833–839.
[31] Jane Hunter, Sharmin Choudhury, PANIC – an integrated approach to the preservation of complex digital objects using semantic web services, International Journal on Digital Libraries: Special Issue on Complex Digital Objects 6 (2) (2006) 174–183.

[32] ISO, Systems and software engineering – measurement process (ISO/IEC 15939:2007), International Standards Organisation.
[33] ISO, Information technology – software product evaluation – part 1: general overview (ISO/IEC 14598-1:1999), International Standards Organization, 1999.
[34] ISO, Information and documentation – records management (ISO 15489-1:2001), International Standards Organization, 2001.
[35] ISO, Software Engineering – product quality – part 1: quality model (ISO/IEC 9126-1), International Standards Organization, 2001.
[36] ISO, Open archival information system – reference model (ISO 14721:2003), International Standards Organization, 2003.
[37] ISO, Document management – electronic document file format for long-term preservation – part 1: use of PDF 1.4 (PDF/A) ISO/CD 19005-1, International Standards Organization, 2004.
[38] ISO, Information technology – open document format for office applications (OpenDocument) v1.0 ISO/IEC 26300:2006, International Standards Organization, 2006.
[39] ISO, Software engineering – software product quality requirements and evaluation (SQuaRE) – measurement reference model and guide (ISO/IEC 25020:2007), International Standards Organisation, 2007.
[40] Anil S. Jadhav, Rajendra M. Sonar, Evaluating and selecting software packages: a review, Information and Software Technology 51 (3) (2009) 555–563.
[41] John Hutchins, Compendium of Translation Software, 15th ed., The European Association for Machine Translation, January 2009. http://www.hutchinsweb.me.uk/Compendium-15.pdf.
[42] Alexander Keller, Heiko Ludwig, The WSLA framework: specifying and monitoring service level agreements for web services, Journal of Network and Systems Management 11 (1) (2003) 57–81.
[43] Ross King, Rainer Schmidt, Andrew N. Jackson, Carl Wilson, Fabian Steeg, The planets interoperability framework: an infrastructure for digital preservation actions, in: Proceedings of the 13th European Conference on Digital Libraries (ECDL'2009), 2009.
[44] Jyrki Kontio, OTSO: a systematic process for reusable software component selection, Technical Report, College Park, MD, USA, 1995.
[45] Jyrki Kontio, A case study in applying a systematic method for COTS selection, in: Proceedings of the 18th International Conference on Software Engineering (ICSE-18), 1996, pp. 201–209.
[46] Hannes Kulovits, Andreas Rauber, Markus Brantl, Astrid Schoger, Tobias Beinert, Anna Kugler, From TIFF to JPEG2000? Preservation planning at the Bavarian State Library using a collection of digitized 16th century printings, D-Lib Magazine 15 (11/12) (2009). http://dlib.org/dlib/november09/kulovits/11kulovits.html.
[47] Lucian Vlad Lita, Monica Rogati, Alon Lavie, Blanc: learning evaluation metrics for mt, in: HLT '05: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Morristown, NJ, USA, 2005, pp. 740–747.
[48] N.A. Maiden, C. Ncube, Acquiring COTS software selection requirements, IEEE Software 15 (2) (1998) 46–56.
[49] E. Michael Maximilien, Munindar P. Singh, Toward autonomic web services trust and selection, in: Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04), ACM, New York, NY, USA, 2004, pp. 212–221.
[50] A. Daniel, T. Menasc, QoS issues in web services, IEEE Internet Computing 6 (6) (2002) 72–75.
[51] Abdallah Mohamed, Guenther Ruhe, Armin Eberlein, COTS selection: past, present, and future, in: Proceedings of the ECBS '07, 2007, pp. 103–114.
[52] National Library of Australia, Guidelines for the Preservation of Digital Heritage. Information Society Division United Nations Educational, Scientific and Cultural Organization (UNESCO), 2003. http://unesdoc.unesco.org/ulis/cgi-bin/ulis.pl?catno=130071.
[53] Cornelius Ncube, John C. Dean, The limitations of current decision-making techniques in the procurement of COTS software components, in: COTS-based software systems, LNCS, vol. 2255, Springer, Berlin/Heidelberg, 2002, pp. 176–187. http://www.springerlink.com/content/t8pp09vhjdncg913/.
[54] Cornelius Ncube Neil, A.M. Maiden, PORE: procurement-oriented requirements engineering method for the component-based systems engineering development paradigm, in: Development Paradigm, International Workshop on Component-Based Software Engineering, 1999, pp. 1–12.
[55] Thomas Neubauer, Christian Stummer, Interactive decision support for multiobjective COTS selection, in: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07), IEEE Computer Society, Washington, DC, USA, 2007, p. 283b.
[56] Robert Neumayer, Christoph Becker, Thomas Lidy, Andreas Rauber, Eleonora Nicchiarelli, Manfred Thaller, Michael Day, Hans Hofman, Seamus Ross, Development of an open testbed digital object corpus, DELOS Digital Preservation Cluster, Task 6.9, 2007.
[57] M. Ochs, D. Pfahl, G. Chrobok-Diening, B. Nothhelfer-Kolb, A method for efficient measurement-based cots assessment and selection method description and evaluation results, in: Proceedings of the 7th International Symposium on Software Metrics 2001, METRICS 2001, 2001, pp. 285–296.
[58] Michael Ochs, Dietmar Pfahl, Gunther Chrobok-Diening, Beate Nothhelfer-Kolb, A cots acquisition process: definition and application experience, in: Proceedings of the ESCOM-SCOPE 2000, Shaker Publ., Munich, Germany, 2000, pp. 343–353.
[59] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu, Bleu: a method for automatic evaluation of machine translation, in: ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, Morristown, NJ, USA, 2002, pp. 311–318.
[60] Anna Perini, Filippo Ricca, Angelo Susi, Tool-supported requirements prioritization: comparing the AHP and CBRank methods, Information and Software Technology 51 (6) (2009) 1021–1032.
[61] Carme Quer, Xavier Franch, Xavier Lopez-Pelegrin, DesCOTS-EV: a tool for the evaluation of COTS components, in: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05), IEEE Computer Society, Washington, DC, USA, 2005, pp. 457–460.
[62] Carme Quer, Xavier Franch, Xavier Lopez-Pelegrin, DesCOTS-SL: a tool for the selection of COTS components, in: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), IEEE Computer Society, Washington, DC, USA, 2006, pp. 358–359.
[63] Shuping Ran, A model for web services discovery with QoS, SIGecom Exchanges 4 (1) (2003) 1–10.
[64] Seamus Ross, Andrew McHugh, The role of evidence in establishing trust in repositories, D-Lib Magazine 12 (7/8) (2006).
[65] Richard Roth, Frank Field, Joel Clark, Materials selection and multi-attribute utility analysis, Journal of Computer-Aided Materials Design 1 (1994) 325–342.
[66] Jeff Rothenberg, Tora Bikson, Carrying authentic, understandable and usable digital records through time, Technical Report, Report to the Dutch National Archives and Ministry of the Interior, The Hague, Netherlands, 1999.
[67] Thomas L. Saaty, How to make a decision: the analytic hierarchy process, European Journal of Operational Research 48 (1) (1990) 9–26.
[68] Rainer Schmidt, Ross King, Fabian Steeg, Peter Melms, Andrew Jackson, Carl Wilson, A framework for distributed preservation workflows, in: Proceedings of the Sixth International Conference on Preservation of Digital Objects (iPRES 2009), 2009.
[69] Pauline Sinclair, Clive Billenness, James Duckworth, Adam Farquhar, Jane Humphreys, Lewis Jardine, Ann Keen, Robert Sharpe, Are you ready? assessing whether organisations are prepared for digital preservation, in: Proceedings of the Sixth International Conference on Preservation of Digital Objects (iPRES 2009), 2009.
[70] H. Skogsrud, H.R. Motahari-Nezhad, B. Benatallah, F. Casati, Modeling trust negotiation for web services, Computer 42 (2) (2009) 54–61.
[71] Stephan Strodl, Christoph Becker, Robert Neumayer, Andreas Rauber, How to choose a digital preservation strategy: evaluating a preservation planning procedure, in: Proceedings of the 7th ACM IEEE Joint Conference on Digital Libraries (JCDL'07), June 2007, pp. 29–38.
[72] Stephan Strodl, Christoph Becker, Andreas Rauber, Digital preservation, in: Handbook of Research on Digital Libraries: Design, Development, and Impact, Information Science Reference, 2009.
[73] Sotirios Terzis, The many faces of trust, IEEE Computing Now, April 2009. http://www2.computer.org/portal/web/computingnow/archive/april2009.
[74] The National Archives of the UK, PRONOM – the technical registry, January 2008. http://www.nationalarchives.gov.uk/pronom/.
[75] M. Tian, A. Gramm, H. Ritter, J. Schiller, Efficient selection and monitoring of QoS-aware web services with the WS-QoS framework, in: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence 2004, WI 2004, 2004, pp. 152–158.
[76] Axel van Lamsweerde, Goal-oriented requirements engineering: a guided tour, in: Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE'01), Toronto, Canada, 2001, pp. 249–263.
[77] K. Vijayalakshmi, N. Ramaraj, R. Amuthakkannan, Improvement of component selection process using genetic algorithm for component-based software development, International Journal of Information Systems and Change Management 3 (1) (2008) 63–80.
[78] P. Weirich, B. Skyrms, E.W. Adams, K. Binmore, J. Butterfield, P. Diaconis, W.L. Harper, Decision Space: Multidimensional Utility Analysis, Cambridge University Press, 2001.
[79] Y. Yang, Jesal Bhuta, B. Boehm, D.N. Port, Value-based processes for cots-based applications, IEEE Software 22 (4) (2005) 54–62.

Please cite this article in press as: C. Becker, A. Rauber, Improving component selection and monitoring with controlled experimentation and automated measurements, Inform. Softw. Technol. (2010), doi:10.1016/j.infsof.2010.02.001