Short Term Preservation for Software Industry

Daniel Draws, Sven Euteneuer, Daniel Simon, Frank Simon
SQS Research
Stollwerckstraße 11
D-51149 Cologne, Germany
+49 2203 9154 0

{daniel.draws,sven.euteneuer,daniel.simon,frank.simon}@sqs.com

ABSTRACT

In today's literature digital preservation and its concepts are usually connoted with long term views on the lifecycle of IT systems and software. In addition to that long term view we believe that concepts available for digital preservation are also useful in short term views where the life span of systems and software is limited to a significantly shorter timeline. In this paper we discuss three different real-world use cases that benefit from DP concepts on a short term basis.

Keywords

Software Escrow, Short Term Digital Preservation, Quality Risk Management

1. Overview

In most literature digital preservation (DP) is associated with a very long term view on systems: According to the Digital Preservation Coalition it is defined as the "series of managed activities necessary to ensure continued access to digital materials for as long as necessary" [2]. This paper utilises the general ideas of digital preservation for shorter term use cases, such as software escrow and due diligence. The aim is to demonstrate that DP's capabilities are important not only in large scale, long term projects but to extend DP to a much wider range of project types and a broad customer base making use of outsourced IT development activities and delivery of IT services. The demonstration of a commercial use case for DP is another goal of the paper.

1.1 Background Scenario "IT Outsourcing"

The basic scenario for our short-term digital preservation addresses the well-established concept of outsourcing that aims to "subcontract responsibility for all or part of an IT function to a third-party service provider that managed and operates the work" [8]. Today, over 7% of all IT-budgets are spent towards outsourcing contracts and this ratio will – accordingly to analyses by Gartner – increase dramatically to 25% for 2020. Interestingly enough the current hype of cloud computing is one specific type of outsourcing and will account for 70% of the overall outsourcing budgets in 2020.

The fundamental concept for all outsourcing contracts is to delegate responsibility (and risks) to a third party. The advantages of doing so are obvious:

- From a client perspective outsourcing enables focusing oneself on one's core competencies in business. For instance, for an insurance company software development and test or IT operations are not core competencies and therefore instead of retaining complete IT testing or IT operations departments they could be subjected to outsourcing them to a specialised third party.
- Outsourcing providers usually have specialised in their fields and can leverage cross-customer synergies and provide more expertise. Expectations are that providers will be able to deliver a service in a more efficient and effective way at a higher level of quality.
- Since outsourcing needs some level of standardisation with regards to definition of services and interfaces between the involved organisations it usually fosters more advanced payment models, i.e. paying per transaction, per value added or per outsourced process step. This facilitates commercial planning processes and budgeting.

On the other hand, delegating responsibilities introduces new risks as an undesired side effect: purchasers become dependent on external providers. These risks need to be managed pro-actively: What happens if an outsourcing provider goes bankrupt or if it is acquired by another company and the new company discontinues this service? What if prices are increased without any justification? Usually, the purchaser only has a black box view onto the service provider with a clear focus solely on the "what to deliver". The service provider is the only stakeholder knowing "how to deliver" and access to this knowledge is at risk if the service provider terminates the contract. Without the specific knowledge it is difficult to keep the service alive, e.g. by handing over the service delivery to another service provider – or maybe by insourcing it again. The two standard risk mitigation approaches are "software escrow" for the case of the provider going out of business and "due diligence" for the case of insourcing the service at a fair price.

The aforementioned risks and their respective mitigation approaches provide the background to DP in a short term perspective: If DP allows for "ensuring continued access" (to IT systems and services) it can be applied to mitigate risks of outsourcing contracts by limiting the impact of third party dependencies. If an outsourcing contract is complemented by a properly set-up DP initiative, the impacts of providers going bankrupt are limited since the DP activities ensure required knowledge is preserved and ready to be transferred to a different party. The challenge shifts towards assuring the stored information is complete and up to date rather than to preserving for a long period of time.

1.2 Overview of the document

This background scenario laid out in the previous section is utilised for the structure of the rest of the document: In Section 2 an established mitigation concept called *software escrow* covering risks associated with providers is presented, and the limitations of the current approach in practise are explained – being the reason for make use of DP. In Section 3 the specific DP concepts required to meet these challenges are revisited in the context of software escrow. In Section 4 an improved software escrow service utilising short-term digital preservation is laid out. In Section 6, several real-world use cases are discussed in the light of this improved concept. The paper closes with an outlook for future work and a summary in Section 6.

2. Software Escrow as Risk Mitigation

The risk of having dependencies to external third parties is not unusual to most industries outside of IT. However, most of the mitigation actions in real life simply change the relationship to the external partner by acquisition and integration into the own organisation. A recent study by Boston Consulting Group and UBS [3] indicates that nearly one in five of the companies surveyed intends to undertake at least one acquisition. At least 18% of the respondents stated "Access intellectual property and R&D" as main driver for M&A activities. So these acquisitions bypasses the risks introduced by outsourcing by changing the relationship to the third party.

The only technique that really mitigates the outsourcing risks while leaving the legal entity status of the outsourcing partner unaffected is outlined in the following subsections, followed by illustrating some pitfalls that motivate our improved approach.

2.1 Escrow Services

A well-established service to reduce the risks generated by strong dependencies to 3rd parties is to establish a so called "Escrow Service". A software escrow is a three-party arrangement, similar to a trust: "*An independent trustee – usually a firm in the business of doing technology escrows – is appointed is the escrow agent for licensor and licensee. The parties enter into a three-way agreement. The licensor delivers a copy of the source code to the escrow agent, and is usually required to deliver a source code update whenever it delivers a corresponding object code update to the licensee under the corresponding license agreement. Upon occurrence of a triggering event, and only then, the escrow agent delivers the escrowed source code to the licensee*". [12]

The risk mitigation approach is as follows: The software purchaser (i.e. the licensee) and the software provider (i.e. the licensor) maintain their legal status and even the level of information to be exchanged between both parties is unchanged. This is an important prerequisite to secure the intellectual property (IP) of the supplier.

In daily business the role of the trustee, the so-called Escrow Agent, does not affect the IP discussion as he receives all information (such as the source code) solely to file away. However, if a so called escrow clause is triggered (e.g. if the supplier goes bankrupt), and only then, the trustee hands out all information to enable the licensee (in the case of software escrow: the software purchaser) to enable the continued operation and maintenance of the licensed application.

This type of service is well established in todays IT market. Market leader NCC for example reports a revenue of 17,9m£ only in the UK with over 100 FTEs [14].

2.2 Pitfalls

During the worldwide financial crisis in 2009/10 some of our customers faced a scenario where the software escrow case occurred but the risks that should have been mitigated revealed their full impact as some key information stored in some digital artefacts were not available. The typical pitfalls around the established software escrow service can be classified into

- Missing artefacts: Software is more than only source code: *"A set of computer programs, procedures, and associated documentation concerned with the operation of a data processing system; e.g. compilers, library routines, manuals, and circuit diagrams."* [10] A software escrow service considering only the source code fails to account for the holistic nature of software. Nowadays a lot of implementation work is done outside the source code proper. Typical examples are models for code generation, architectural views, testware, technical documentation, used libraries, configurations of development environments etc. Without these additional digital artefacts the source code has only limited value: It cannot be understood, analysed, changed or outsourced to another vendor. The more complex and developed the applied technology (e.g., .Net or J2EE) the more business logic is stored in artefacts outside the source code.
- Low quality of deposited material: In many cases the deposited source code was either incomplete or inconsistent with the corresponding binary code. The source code was not commented, could not be analysed in any efficient way and did not follow standard software engineering techniques such as modularisation and decoupling. Exhuming a code basis with these attributes does not allow to re-compile/re-build the application and hinders any maintenance work that is necessary to adjust the application due to changed requirements.

If these pitfalls occur in real life the consequences can be devastating: It can start from the need for investing a large amount of money to conduct software-archaeology before continuing maintenance work and goes up to the complete re-development of the application being under software escrow.

2.3 Challenges

Consequently, the key challenge to be addressed in order to make the Escrow Service work in practice and avoid the aforementioned mistakes is to answer the following question:

How can we make sure that all relevant information for taking over an IT system exist and are of appropriate quality?

This is the point where we hope to bring in DP tools and concepts like [5]. Similar to software escrow services, DP tries to preserve digital artefacts necessary for assuring their availability over time. For software escrow, we need to preserve complete business processes (including tools, external knowledge etc.) at a sufficient level of quality of the preserved artefacts. This same is valid for DP (at least for digital objects), so the capability for reuse is obvious.

3. Software Escrow View on DP

3.1 How long is long-term?

Typically, DP is connoted with the aspect of long term preservation and most of the concepts of DP have been developed with the long term views (decades rather than months or years) in mind. We believe that the concepts developed so far are also very valuable in the case of software escrow and can be applied beneficially for much shorter periods of time. In some cases the timespan may only be a couple of months and we make use of a slightly different view on 'long-term'. Consider the definition for 'long-term' given by the OAIS:

"Long Term is long enough to be concerned with the impacts of changing technologies, including support for new media and data formats, or with a changing user community. Long Term may extend indefinitely." [5]

To our experience, Digital Preservation is not only required in the 'long-term' from a time based understanding as changes in technology can occur much more frequently. From the software escrow point of view involved parties have to keep information fit for purpose across the lifecycle of technologies (or any other kind

of significant change in the context of the information that would normally render the respective information useless).

3.2 Basic Preservation Process in TIMBUS

According to TIMBUS project [19], one of the most up-to-date project funded by EU around Digital Preservation, the high level process of DP comprises three stages (cf. Figure 1).

- Expediency: In this, the fundamental steps need to be taken to determine what should be preserved.
- Execution: After the expediency has been established it is necessary to actually execute the DP preservation activities (e.g. conserving and archiving artefacts).
- Exhumation: In this stage, the preserved artefacts are brought back from the libraries into live environments to take up the regular 'business activities'.

Note that the stages and their activities are independent of the timespan of a preservation project, so it does not depend on the long-term view.

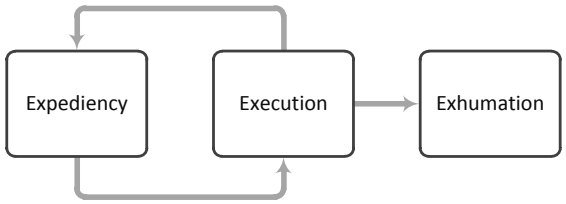

Figure 1: The three phases of the digital preservation process

3.3 Models in DP

From our knowledge DP research so far has already taken care of "How to" preserve digital information by focussing on preservation processes and the lifecycle of different media, data formats and storage technologies (cf. [21], [22]). For DP in general (and software escrow in particular) the still open (but crucial) question is: What is the relevant information to be preserved and what digital objects (DO) contain this information? That is often not easy to answer because the boundaries of the system to be preserved are difficult to identify and usually debatable. For example, in almost all practical cases software depends on and makes use of third party components and libraries. To what extent do these third party artefacts need to be preserved? Where is the line between relevant context and the (for the time being) non-relevant context? This challenge increases when using Cloud services like SaaS or PaaS.

Our approach to answering the question of contexts to the best possible extent lies in using explicit models for the context of the systems to be preserved. The aim of these models is to preserve not only the DOs itself but additionally to capture the semantics of the objects. We make use of well-established architecture frameworks for specific domains as a starting point to identify and structure the DOs. Well known examples for these architecture frameworks are the NATO Architecture Framework (NAF) [19], the Zachmann framework [6] or The Open Group Architecture Framework (TOGAF) [20]. It is expected that these architecture models are describing a holistic view in their domain.

4. Software Escrow Modelling Approach

The key to a successful Digital Preservation that can be completely used for software escrow is the holistic scrutiny of artefacts, their components and their respective properties. The vehicle we apply to fulfil this requirements of digtal objects is the so called quality risk management framework (QRM) [8]. The QRM framework has been used successfully as a foundation for project risk management and its concepts and ideas are applied in conjunction with DP concepts as to improve the success rate of software escrow.

4.1 Overview of the QRM Framework

The generic risk management framework consists of several components, whose instantiation is crucial for holistic software escrow. The overall QRM framework is depicted in Figure 2. The setup of the framework is explained in the following paragraphs.

Firstly, we identify the relevant DOs required for digital preservation by building a taxonomy for the context. In the framework these objects are named *control objects* (cf. Figure 2, "1"). Secondly, the quality attributes of digital objects are inventoried and classified – in terms of the QRM framework these are named *control attributes* (cf.Figure 2, "2"). In order to identify the essential aspects for digital preservation the Cartesian product of control objects and controls attributes is determined in a third step. The product of (control object, control attribute) is called a *control*

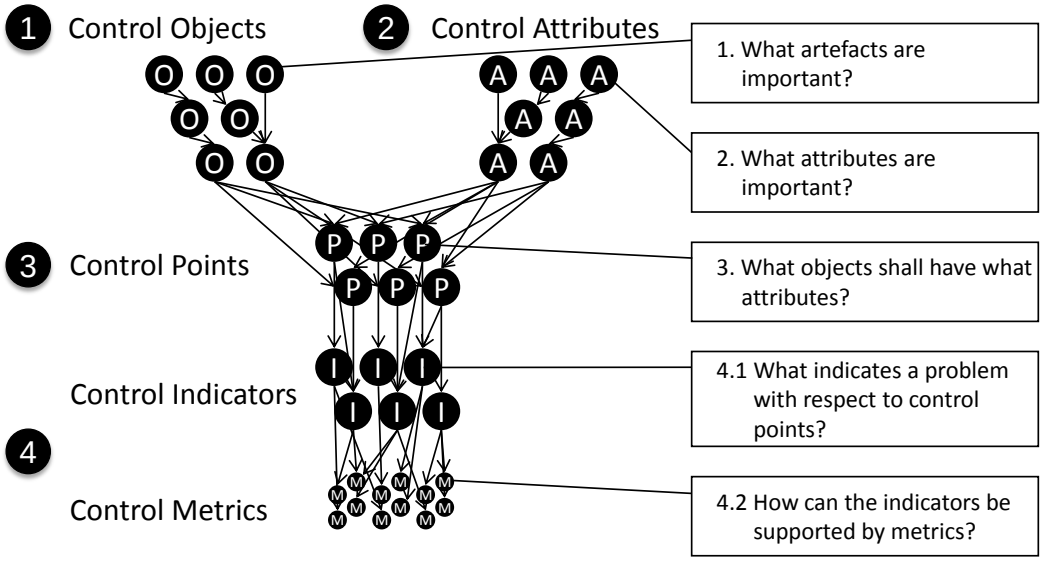

Figure 2: The QRM framework and its components

point (cf. Figure 2, "3"). For each of the control points we determine its relevance on a Likert scale (e.g., ++ - very high, to -- – very low) indicating its priority for subsequent steps.

When control points are defined we have laid out the full view on what to preserve with which priority and which criticality. As a third step, we define *control indicators* and *control metrics* (cf. **Figure 2**, "4") supporting the control points with tangible information based on the artefacts.

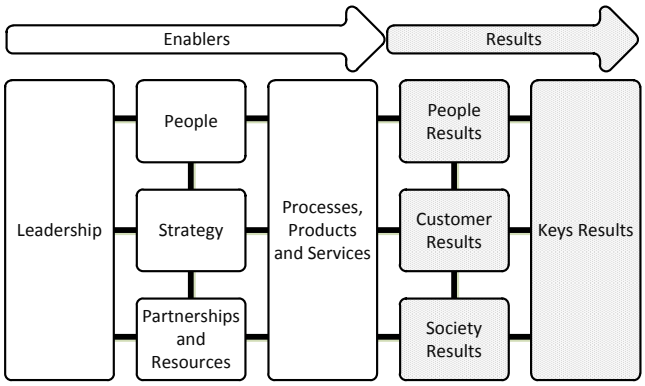

Figure 3: EFQM excellence model as initial context setting for software escrow in DP

How to apply these sequent steps in general? The solution is to reuse existing catalogues from other disciplines. For the development of the taxonomy (see above) the reuse of standard models for the relevant context is possible. For example, if there is a need to preserve the organisational context we can make use of a well-defined model such as the European Framework for Quality Management (EFQM) – model [15]. This model defines objects and their attributes for evaluating organisations and provides a valuable source for modelling the contexts to be preserved. A catalogue of preservable objects could be a taxonomy based on the EFQM-model which is illustrated in Figure 3.

But reuse can be done on the attribute level as well: they are independent from the objects in the first phase and can be derived from established standards such as ISO9126 [10], QUINT2 [17] and research in [2].

Both, the catalogue of digital objects and quality attributes are refined for the specific purpose of software escrow in the following Sections.

4.2 A Catalogue of Digital Objects for Escrow

In the software escrow use case for DP we need to ensure that the full set of digital objects required for the maintenance and evolution of a software system is preserved for all agreed releases of the software from the software provider's repositories. A first and simple approach to preservation of what is required in the software escrow case starts intuitively with the software's source codes. In case the software escrow partners are aware of the effort it takes to re-build the executable software system from the source code the compiled and ready-to-run executables are preserved additionally. To our experience, these types of digital objects are considered in the first place as it is one of the most obvious artefacts of value for a software purchaser.

However, this is by no means all it needs for a successful exhumation of the software at a later point in time. The IEEE Standard Glossary of Software Engineering Terminology reveals for good reasons a far wider definition of software [10], taking into account a far more holistic set of artefacts worth to be preserved.

A significant proportion of the artefact types and artefacts mentioned in [10] (compilers, library routines, manuals, documentation) is usually in practise not considered as a part of a software purchase and therefore tends to be neglected in the escrow preservation process. Examples for important documentation are the software architecture, the programmer's manual and other 'internal' documentation usually only required for maintenance purposes. (Which is exactly what the purchasing party wants to take over in case of the escrow exhumation.) More detailed taxonomies for documentation can be derived from text books such as Sommerville [17]:

- System Documentation
  - Requirements
  - System Architecture
  - Program Architecture
  - Component Description
  - Source-Code-How-To
  - Maintenance Guide
  - Environment Description
- End User Documentation
  - Functional Description
  - Reference Manual
  - Installation Manual
  - System administrators guide

In the case of software escrow exhumation, the software purchaser needs to take over the full maintenance process for the software under escrow. To be in a position to pick up these tasks in an efficient way, artefacts beyond the end user view are required. A first incomplete and project specific list contains

- configurations of the software and build environment
- the build environment itself and other third party tools and libraries
- software models and modelling tools
- test tools and test ware (tests, test data, automation, …)
- licenses to run the aforementioned tools and make use of 3rd party libraries
- licenses for intellectual property

Note that the escrow should ensure that for example licenses are issued for the purchaser, not for the original software developer. If license management is enforced by technical means it must be ensured licenses (and the depending tools) can be used for the purchaser.

Our current experience leads us to the taxonomy depicted in Figure 5. This taxonomy is usually used as a starting point for a more detailed elicitation and determination of the project specific DOs for software escrow. So far, we have seen a number of re-occurring DOs across different projects. But due to various reasons (e.g. software application domain terminology, business culture, or simply project lingo) it seems that the taxonomies are most useful if tailored to the project's context.

4.3 Quality Attributes of Digital Objects for Escrow

After having determined what to preserve for software escrow in the previous section, we address the properties of what to preserve in more detail. In many cases the exhumation already fails concerning a very simple attribute of the DOs – their existence. As many artefacts are forgotten or ignored the exhumation cannot be successful.

Digital Objects			
Interfaces	Documentation	Test Environment	Build Environment
Services Used	System	Infrastructure	Operating Systems
Data feeds	Requirements	Executables	Compilers
Network connections	System Architecture	Configurations	Programming Languages
Operating Systems	Program Architecture	Licenses	Runtime Environments
Services Provided	Component Description	Test cases	Configurations
Data feeds	Source-Code-How-To	Tests	3rd party libraries
Network connections	Maintenance Guide	Regression Tests	Licenses
Operating Systems	Environment Description	Test data	Applications
Business Processes	End User	Test scripts	Binaries
Intellectual Property	Functional Description	Automation	Executables
Patents	Reference Manual	Test reports	Database Schema
Algorithms	Installation Manual	Design Environment	Source Code
Methods	System administrators guide	Models	Configurations
Contracts	Processes	Configurations	
SLAs	Business Processes	Modelling tools	
UP	IT Processes	Licenses	
Supply Chains	Supporting Processes		

Figure 5: Digital objects for software escrow

However, even if the artefacts do exist, the software purchasers must make their expectations towards the DOs explicit. If the purchaser has to take up maintenance activities they have to have an interest for example not only in the existence of relevant documentation but also in the quality of the respective documents. Thinking in terms of software engineering a good starting point for attributes of software artefacts is ISO 9126 [11] with QUINT2 [17] extensions (ISO 9126 has been superseded by the ISO 25000 series but remains a useful guidance for the purposes of this paper). Additionally, we enrich these attributes by attributes derived from research in the field of digital libraries [2]. The top level categories proposed can be reused by generalising their intended meaning from software to general artefacts. They are the following:

- Reliability: A set of attributes that bear on the capability to maintain the level of performance under stated conditions for a presumed period.
- Usability: A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.
- Portability: A set of attributes that bear on the ability of artefacts to be transferred from one environment to another.
- Maintainability: A set of attributes that bear on the effort needed to make specified (and consistent) modifications to artefacts.
- Functionality: A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
- Efficiency: A set of attributes that bear on the relationship between the level of performance of the software or the processes and the amount of resources used, under stated conditions.

The complete list of control attributes for software escrow is listed in Figure 4.

As before, the control attribute taxonomy needs to be tailored to the project context of the software escrow to be of most use. Having defined the taxonomy of control attributes independently from the specific control objects allows for an truly holistic view in the next step.

4.4 Software Escrow Control Points

Both the list of control objects and the list of desired control attributes can now be contrasted with each other. This can simply be done by calculating the Cartesian product of the two taxonomies and yields a matrix of all possible combinations of DOs with quality attributes. As we produce a full matrix we include potentially meaningless combinations of control objects and control attributes, we can now additionally prioritise the control points (e.g., on a scale from "not relevant" to "very important"). We can also use a more fine granular scale, but for illustrative purposes

Control Attributes(ISO9126 and QUINT2, DL)			
Reliability		Maintainability	
	Maturity		Analyzability
	Fault tolerance		Changeability
	Recoverability		Stability
	Availability		Testability
	Degradability		Manageability
	Relevance		Reusability
	Significance		
Usability		Functionality	
	Learnability		Suitability
	Understandability		Accuracy
	Operability		Interoperability
	Explicitness		Compliance
	Customizability		Security
	Attractivity		Confidentiality
	Clarity		Integrity
	Helpfulness		Availability
	User-Friendliness		Traceability
	Accessibility		Completeness
	Consistency		Preservability
Portability		Efficiency	
	Installability		Time Behaviour
	Replaceability		Resource Behaviour
	Adaptability		Similarity
	Conformance		
	Timeliness		

Figure 4: Control attributes for digital objects in software escrow

and in practical use in past projects already the simply five step scale proved effective. The prioritisation of the control points must be agreed between purchaser and vendor as it will guide the escrow agent in subsequent steps of the software escrow to assess both completeness and adequacy of the preserved DOs.

4.5 Indicators and Metrics

The sheer act of defining the control points itself already provides benefits as it clarifies what to look at and which attributes are relevant to which artefacts. In a final step, the control points are associated with *control indicators* and *control metrics* (cf. Figure 2, "4"). Control indicators help to identify quality risks by making use of simple metrics. The following example was arbitrary selected and only illustrates the idea in general. The control point ("Requirements", "complete"), being very important for both Digital Preservation and for software escrow, could be supported e.g. by an indicator "98% of requirements have an ID". The supporting metrics are (a) count requirements, (b) count requirements with ID, (c) compute the ratio of (a) and (b).

In practise, indicators are most successful when expressed in terms of non-desired properties. For example, it is very difficult to assess the quality of requirements written in natural language. Rather than trying to measure the quality directly, it is attempted to identify the "bad" requirements by searching for terms like "to do", "tbd", etc. If we identify one of the search terms in the context of a requirement (a task that can even be automated to some extent) we assume the requirement's quality is low: In this case it does not make sense to digitally preserve them nor does it make sense to be part of any software escrow.

By following this pattern of negating quality the set of indicators comprises a safety "net" of things we do not want to see. Having a sufficient number of indicators significantly reduces the risk of missing a bad "smell" and allows for re-adjustment of the quality model over time.

5. Applying DP in Software Escrow Use Cases

The concepts described in the preceding sections are applicable to a multitude of use cases. In the following we will outline three of those use cases that highlight the value that the application of DP-techniques can add to software escrow.

5.1 Holistic Software Escrow

The overarching software escrow process starts when the two parties – software purchaser and software provider – agree the terms and conditions of the escrow contracts with the help of the software escrow agent. The software purchaser identifies the need for software escrow and subsequently both software purchaser and software provider prepare for an escrow agreement. The necessary steps for the execution in addition to the usual software rollout and maintenance procedures, and commitment on the triggers of the software escrow exhumation case are determined. The software escrow processes from the viewpoint of DP can be illustrated in . When comparing Figure 1 and Figure 6 it becomes obvious that these processes constitute a direct application of the DP processes to the software escrow problem space.

The first step, called software escrow expediency, aims at establishing (first iteration) or refining/revising (further software releases) the DOs, their quality attributes and the QRM model including the respective control points required. Secondly, per release of the software under escrow the software provider makes the preserved assets available to the escrow agent and bundles the DOs with their QRM model. The escrow agent then can ensure the completeness and adequate quality of the artefacts provided by the software provider (utilising technical support) without disclosing the preserved information to the software purchaser. If the escrow case does not occur before the next release of the software product the escrow process is hibernated. The agreed software maintenance and support is delivered by the provider. Typically, software products are updated from time to time and consequently, after every new rollout of a new release of the software under escrow the escrow process is triggered again.

In case the predefined events terminating the existence of the software provider occur, escrow exhumation is triggered. The software escrow agent hands over to the software purchaser all

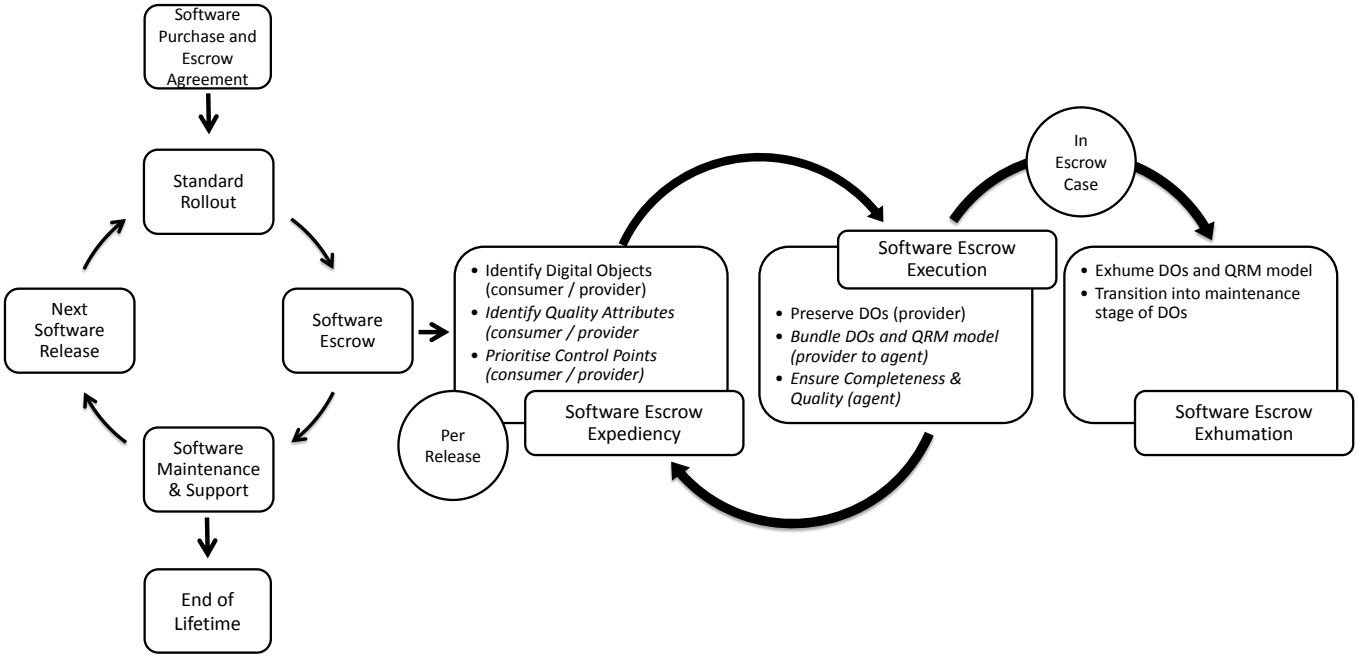

Figure 6: The high level software escrow process

assets in his behold. The software purchaser then may take the necessary step as to re-vitalise the software maintenance and support activities either on his own or with the support of a different software supplier.

This approach is well-established in the software and IT industry and there is a variety of vendors that offer this software escrow service. Figure 8 illustrates the roles and tasks for the software escrow approach, in which the software provider hands over certain assets being part of their intellectual property to an escrow agent who safely files away and manages access to those assets. The software purchaser simply gets the executable software, just as would be the case without the software escrow agreement.

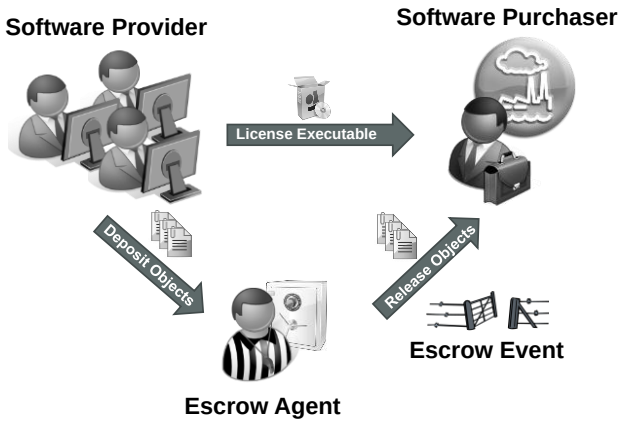

Figure 7: Roles and tasks of the software escrow scenario

Usually, the assets would remain in possession of the escrow agent until the contract between software provider and software purchaser terminates or until other events render the escrow unnecessary.

Unfortunately, with software escrow, risks come into effect at a late point in time: If and only if the escrow event occurs the software purchaser will get access to the escrow asset base, and only then is he able to determine the suitability of stored assets.

The holistic software escrow, utilising the Digital Preservation research area, goes beyond the provisioning of a simple storage and management service by the escrow agent by utilizing the power of the approach detailed in the preceding sections (cf. Section 4). By including an appropriate amount of quality assurance into the escrow process upon software escrow expediency and software escrow execution, the rate of success upon software escrow exhumation can be greatly increased, increasing trust with the software purchaser and enabling the software provider to ask for higher compensation for the software escrow option.

5.2 Ex-Post ESCROW Analysis

As discussed previously, there already exists a market for software escrow that has developed primarily in the Anglo-Saxon countries, where players are offering a fairly basic escrow service with limited success, as more often than not the quality of deposited assets is insufficient for exhumation of the software at a later point in time.

Subsequently, there is a need for the software purchaser to determine the course of action in such a situation. The fundamental question that needs answering is whether it is worthwhile to invest into re-engineering the system based on its available artefacts or whether the system needs to be rebuilt from scratch, discarding whatever was supplied as part of the escrow effort.

The procedure of choice for evaluating the various alternatives and for answering this underlying question is to estimate the respective investments for the relevant alternatives. For the alternatives that target the re-use of assets from the escrow this requires transparency about the status quo as well as enough data to support a reliable estimation of effort necessary to transform those assets into value for the business.

An ex-post escrow analysis as depicted in Figure 7 yields the necessary transparency by

- identifying the software purchaser's vision, goals and subsequent requirements towards the system
- making use of the software escrow object catalogue to define a target state of required assets
- making use of the software escrow attribute catalogue to map the software purchaser's requirements to and prioritize attributes
- conducting a gap analysis to identify the gap between this target state and the status quo
- estimating the effort required to fill this gap by reverse engineering of those parts of the system that are missing, incomplete, or outdated

When comparing this list of activities with the tool box supplied by DP, it quickly becomes evident that we can draw heavily on DP techniques to conduct the abovementioned tasks. The following sections detail this link.

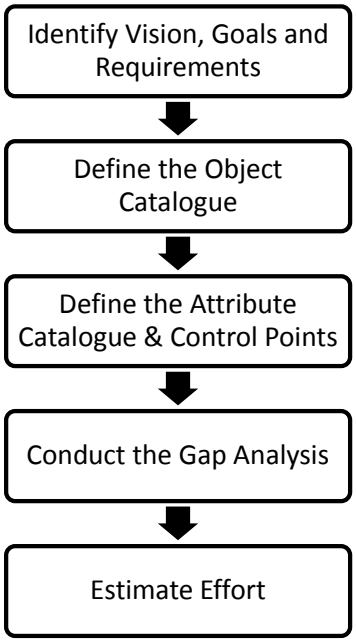

Figure 8: The ex-post software escrow process

5.2.1.1 Identifying Vision, Goals and Requirements

Before any of the alternatives can be evaluated and ranked against each other, a precise definition of the target state is required. The required documentation to define this target state consists of strategic visions, broken down into goals and objectives which in turn decompose into a set of requirements that operationalise these goals (similar to the first steps in the GQM approach [2]).

Some of this information will be available already, while others may not. In any case, existing documentation needs to be reviewed to gauge its accuracy and actuality before it may be used as a foundation for the analysis. Any documentation that does not

yet exist needs to be made explicit. The discipline of Requirements Engineering has developed a proven set of methods to extract these requirements using a variety of techniques [15].

5.2.1.2 Defining the Object Catalogue

Once the high level target state is known, the preparation of the actual gap analysis can commence. The first step towards this gap analysis comprises the tailoring of the escrow object catalogue to fit the specific requirements in place (cf.Figure 2, "1"). The standard escrow object catalogue depicted in Figure 5 is used as a starting point from which all irrelevant objects are stripped.

The result of this activity comprises a catalogue of objects that need to be present within the software escrow asset base for the assets to be used in the intended fashion. This catalogue constitutes the starting point for setting up an escrow quality model that will be used to support the subsequent gap analysis.

5.2.1.3 Defining the Attribute Catalogue & Control Points

More often than not the sheer existence of an artefact is not enough to render it a useful asset that supports the viability of a system.

Subsequently, each of the objects in the object catalogue must be part of an overall QRM model and needs to be defined that establishes and operationalizes the expected quality for this specific object (cf. Figure 2, "2"). The generic escrow attribute catalogue depicted in Figure 4 serves as starting point for the analysis of each escrow object that determines how the previously identified requirements apply to each of the escrow objects (cf. Figure 2, "3").

These combinations of escrow objects with escrow attributes are result in the escrow control points. For the above mentioned reasons, not all combinatorial possible combinations of escrow objects and escrow attributes are meaningful. The meaningful ones to be considered require a prioritization due to economic reason, as discussed in Section 4.4.

Additionally, for each such escrow control point, a suitable verification method needs to be determined and documented, including all parameters that may have an influence on the result of the verification. The verification method can be an in-depth analysis of the control object with regards to the respective control attribute in the most complex case or, in simpler cases can be supported by – simple – indicators and metrics.

5.2.1.4 Conducting the Gap Analysis

The actual execution of those verification activities involves the application of methods and techniques from the quality assurance discipline to the software escrow asset base in order to determine the degree of gap that may exist between target state and the artefacts contained in the asset base.

The results of these verification activities are mapped to the relevant software escrow control points (cf. Figure 2, "4"). Doing this guarantees traceability from verification results back to individual escrow objects and attributes as well as the ability to aggregate the results.

Once verification activities have concluded, the so annotated QRM model is used to systematically identify the gaps between verification results and target state.

5.2.1.5 Estimating Effort

Finally, the gap analysis results are used to inform the effort estimation that makes the cost of using the escrow asset base explicit by attaching a figure to it.

For each of the gaps identified, viable mitigations need to be identified and for each of those expected effort and cost needs to be estimated. In addition to this, all other direct and indirect costs, such as costs arising from the need to license third party intellectual property in order to use the escrow asset base need to be considered. This holistic estimate of costs associated with (re-)use of the escrow asset based can then be used as part of a larger evaluation and decision making process that ranks all potential alternatives against each other using the predefined requirements.

In supporting this decision making, the ex-post escrow analysis can contribute as much value to the business as is possible for a post-mortem analysis. While it is certainly able to create transparency regarding the viability of the deposited escrow asset base it cannot bring back artefacts that have not been deposited, be it intentionally or for want of knowledge that certain artefacts are required. In a worst case scenario, the ex-post analysis can only establish that the deposited assets are without any value to the software purchaser and thus do not constitute a viable alternative. The ability to influence the course of action before any damage is done is a luxury that is only afforded to the holistic software escrow documented in Section 5.1 and utilising the Digital Preservation knowledge base.

5.3 Due diligence Analysis

Mergers and acquisitions (M&A) of companies are a risky undertaking. Recent studies find that almost two thirds of all mergers and acquisitions fail, for instance resulting in a split along old corporate borders [9].

In those cases where the corporate management needs to report to a diverse group of owners and other stakeholders such as with publicly traded companies there is a subsequent requirement to prove to owners and stakeholders that corporate management is diligent in executing the merger or acquisition by closely scrutinizing the partner or acquiree. One integral part of this scrutiny is a financial valuation of the organization and all its assets.

IT systems and the software and applications that drive those systems are usually part of the tangible assets that are owned by any modern company. Unlike real estate, a corporate fleet or factory buildings with production lines inside, software is notoriously difficult to value correctly. In addition to this, more complex M&A scenarios may require parts of the affected companies and their assets to be severed from the rest of the organization, for instance to be sold off separately because of regulatory concerns.

All this calls for both the precise valuation of software and IT assets in general and for the ability to safely deposit assets into escrow while the M&A transactions are being finalized by all affected parties.

Subsequently, this use case constitutes a hybrid between a software escrow, where software assets are being put into Escrow and an ex-post Escrow analysis where an extant asset base is evaluated in terms of its future viability for the intended use cases.

This dichotomy becomes transparent when inspecting the methodological building blocks necessary to conduct this analysis:

- In a first step all IT and software assets relevant for the analysis need to be surveyed and mapped to an overall IT and software landscape that shall serve as input to the valuation. In terms of the QRM framework, the establishment of the control objects catalogue is a useful tool to achieve a comprehensive overview of existing assets.
- The valuation of assets itself can be conducted using the procedure detailed in Section 5.2, with the only real dif-

ference being that the assets to be analysed are not part of an software escrow asset base. This requires additional preparatory activities to collect and collate all the required assets for the analysis. From the QRM framework perspective, this step corresponds to the establishment of the control attributes catalogue and the subsequent elicitation of control point.

- Once the asset base is complete and the ex-post analysis has been conducted, a first value estimate can be delivered. The estimation can make use of the QRM model by supporting the various control points with price indicators and thereby systematically derive a transparent overall assessment.
- Depending on the discussed influencing factors, some assets may need to be put into software escrow for the duration of time during which the merger or acquisition is being executed.
- In order to diligently curb risk, all parties involved need to ensure that a holistic software escrow is instituted to make sure that whatever is put into escrow conforms to its estimated value after the merger or acquisition has been finalized.

6. Summary and outlook

In this paper we have laid out use cases taken from the industry scenario "IT outsourcing" that mitigate specific risks by making use of software escrow services and due diligence analyses. To support the use cases in practise we exploit the concepts of DP and apply the QRM framework to provide an holistic view on quality risk management. We have pointed out that the results from DP research are not limited to long-term views but can also be deployed in typically short-term scenarios.

Derived from our experiences we suggest a new definition of a holistic software escrow based on the definition made in [12]: "*An independent trustee is appointed as the escrow agent for licensor and licensee. The parties enter into a three-way agreement. The licensor delivers a copy of* ***all source artefacts needed to build the object code and maintain the software*** *to the escrow agent, and is usually required to deliver a update of the* ***artefacts*** *whenever it delivers a corresponding object code update to the licensee under the corresponding license agreement. Upon occurrence of a triggering event, and only then, the escrow agent delivers the escrowed artefacts to the licensee".*

Currently, we are evaluating the suggested definition and further use cases –in the context of the TIMUBS project [19] – for applying QRM and DP in various contexts for the benefit of our customers.

7. Acknowledgements

Parts of this work have been supported by the European Union in the TIMBUS project [19]: "Digital Preservation for Timeless Business Processes and Services", Grant Agreement Number 269940

8. References

[1] American Society for Quality/ISO 8402:1994
[2] Basili, V., Caldiera, G., Rombach, H. D.: The Goal Question Metric Approach. In: Encyclopedia of Software Engineering. John Wiley & Sons, 1994
[3] BCG The Boston Consulting Group: "M&A: Ready for Liftoff? A survey of European Companies' Merger and Acquisition Plans for 2010", December 2009
[4] Beagrie, N., Jones M. (maintained by Digital Preservation Coalition): Preservation Management of Digital Materials: The Handbook, 2008, available at http://www.dpconline.org/advice/preservationhandbook
[5] Consultative Committee for Space Data Systems: Reference model for an Open Archival Information System recommendation for space data system standards. Washington D.C, 2002
[6] Zachmann, J.A.: A framework for information systems architecture, IBM Systems Journal 26 (1987)
[7] Goncalves, M.A., Moreira, B.L., Fox, E.A., Watson, T.L.: ''What is a good digital library?'' – A quality model for digital libraries, Information Processing and Management 43 (2007)
[8] EFQM Excellence Model 2010. http://www.efqm.org/
[9] Fulmer, R.M., Gilkey, R.: Blending Corporate Families: Management and Organization Development in a Postmerger Environment, The Academy of Management Executive Vol. 2, No. 4 (Nov., 1989)
[10] IEEE Standard Glossary of Software Engineering Technology, 1983
[11] International Standard ISO/IEC 9126, Part 1, Software engineering – Product quality – Quality model, Beuth-Verlag, Berlin (2001)
[12] Meeker, H.: "Thinking outside of the Lock Box: Negotiating Technology Escrow", Computer & Internet Lawyer, No 9, 2003
[13] NATO Consultation, Command and Control Board, NATO Architecture Framework (NAF), http://www.nhqc3s.nato.int
[14] NCC Group plc: "Preliminary Annual Results for the year ended 31 May 2010", July 2010
[15] Pohl, K.: Requirements Engineering, Springer, 2010
[16] Simon, F., Simon, D.: Qualitäts-Risiko-Management, Logos Verlag, November 2010
[17] Software Engineering Research Centre Netherlands, "Kwaliteit van Softwareprodukten – Ervaringen met een kwaliteitsmodel", http://www.serc.nl/quint-book.
[18] Sommerville, I.: Software Engineering, 8th edition, Pearson Education (2007)
[19] TIMBUS, http://www.timbusproject.net/
[20] The Open Group Architecture Framework, http://www.opengroup.org/togaf
[21] PLATO, The Preservation Planning Tool, http://www.ifs.tuwien.ac.at/dp/plato
[22] H. Neuroth, A. Oßwald, R. Scheffel, S. Strathmann, M. Jehn, "nestor-Handbuch: Eine kleine Enzyklopädie der Langzeitarchivierung", 2009