Migration of Complex Original Environments – Verification and Quality Assurance Challenges

Functional Longterm Archiving Group
Albert-Ludwigs University Freiburg
Hermann-Herder Str. 10
79104 Freiburg i. B., Germany

ABSTRACT

Preparing a digital artefact's runtime environment for emulation can be seen as an alternate form of migration taking place layers below the objects of interest like application, operating system, or even on the level of the machine. The latter one is especially attractive for complete original environments or specifically complex objects living in a digital ecosystem of one or more machines. Full system preservation can be formally described as a migration function operating on the level of the hardware. As a result of the full system preservation process parts of the digital object's software and/or hardware runtime environment might be altered. Starting from an intuitive approach "the object renders" or "the system boots properly in the emulator" the requirements for an authentic full system preservation process need to be defined, controlled and measured in a formal way. The authenticity and completeness of the future rendering result could be endangered and require some form of quality assurance.

1. INTRODUCTION

In certain situations traditional object-centered migration strategies may fail or may be too expensive or too complex to achieve reasonable results. The concept of using emulation as a preservation strategy is to change and adapt parts of the system which do not or only to a small degree influence the digital artefact of interest. Objects like computers of famous personae, electronic business processes, scientific e-labs, many pieces of digital art or large database setups cannot be migrated like traditional digital material, e.g. images or text files, without risking the loss of significant information and/or the context. These type of digital objects usually contain complex configurations, and have be developed gradually over time or are (partly) interactive. In addition, some of the artefacts may be customized for special purposes e.g. highly specialized tool-chains paired with non-standard system tweaks.

The outlined problem can be solved by migrating complete systems instead of single artefacts [5], which means to operate on a more general level, i.e. on the lower levels in the software-hardware stack. Instead of migrating the artefacts directly, the physical hardware is replaced by an emulator. In contrast to programming interfaces of operating systems, the functionality of single applications or the description of data formats, the hardware specifications are usually openly available for most of today's computer architectures. No application and operating system needs to be re-implemented. The functionality of a given hardware platform is usually much better documented compared to (proprietary) operating systems or applications. By now we run a couple of system migrations for various platforms and offering several emulated original environments [7]. Most of the system migrations were successful, some suffered degradations like a reduced screen resolution.

The aim of a preservation strategy is the reliable, authentic access to a preserved digital artefact. To make the emulation strategy achieve such a goal, the involved processes and workflows needs to be well understood, reproducible and quality controlled. Ideally, it can be proved in some or other way that a certain action produces (exactly) the expected outcome. One question is whether it would be possible to develop a semi-formal model for full system migration and system emulation. Here it should be discussed whether established concepts in migration strategy like significant property analysis can be applied and, if this is the case, to which degree.

The challenge in format migration is comparison of the relevant object properties before and after the process. Ideally, the set of characteristics should not differ at all. To check the results of migration workflows, approaches like property description languages like XCDL were introduced and discussed [1, 2, 6]. Unfortunately, the whole procedure depends on a complete understanding of formats and thus measuring differences or closeness is not trivial at all. By moving "down" in the software-hardware-stack to the system layer, the single artefacts are not touched any more, but the machine layer is modified. This approach has a significant advantage: The number of artefacts (here: complete original environments) is much smaller than single objects. Nevertheless, the problem of quality assessment is not easily accomplished. Up to now we used the "intuitive approach" by comparing the rendering results of relevant objects side-by-side. If the objects behaved (exactly) the same way on the original and emulated hardware, we presumed an equivalence. In this approach we simply ran several actions like opening a MS-Word 2.0 document, querying a database or executing a certain sequence of game-play. Nevertheless, the approach is basically hands-on and allows only cursory checks. It works for rather simple artefacts and bases on the assumption that, if a given set of characteristics can be verified, these characteristics can be assumed for the whole artefact. [4]

The various artefacts have different specific requirements regarding objects' authenticity such that the user's perception of the system environments differs as well as the expected system functionality in emulation. Thus, a couple

of open research questions remain. It remains, for example, to be seen how quality assurance can be implemented in emulation-based preservation workflows.

2. SYSTEM MIGRATION FUNCTION

Following a couple of experiments we want to formally describe system migrations to evaluate whether it is possible to "prove" the completeness regarding certain characteristics after the transformation. This would be helpful to show that system emulation is a valid option to preserve long-term access to complex artefacts and research environments. Full system preservation and to a certain point emulation of standard original environments can be described as a migration on the machine layer. The migration can be interpreted as a specific function mapping a set of characteristics from the original domain O – the original software and hardware environment – into a co-domain – the emulated environment E. Optimally, this transformation is a homomorphism and everything "behaves" in the co-domain exactly as it did in the original one. Unfortunately, in the real world the domain can be larger than the co-domain. Not only, do we not have emulators for every hardware architecture ever built, but the existing emulators may lack a few of the original's characteristics.[1] Nevertheless, in many preservation scenarios a few technical limitations do not necessarily decrease the quality and usability of the result.

Computers of different architectures still share common characteristics as they conform to the von-Neumann or "stored-program computer" model. The permanent storage of a computer system usually contains all necessary software components to boot the machine into working state, allowing interaction with humans or other computers. Beside the permanent storage the computer architecture with all its peripherals influences how a particular machine operates. Following these considerations, we split the *migration function* into two main components (Fig. 1), the *preservation of the permanent storage* (A) and the *mapping onto the emulated hardware and its configuration* (B).

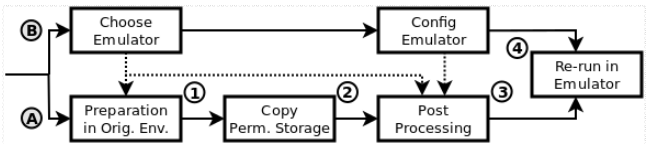

Figure 1: Conceptual System Migration Model

The decomposed migration function can be evaluated and the disk transforming steps can be split in sub functions with $A = A_1 \circ A_2 \circ A_3$: The preparation step (1) is executed within the original environment and may include actions like filesystem de-fragmentation or resetting of passwords and the conversion to a compatible configuration set depending on the emulator chosen. This action is atomic and should yield exactly the same results when run on the original platform or within the emulator. The second step (2) is the block-wise copying of the permanent storage to a container file [8]. This file can then be mounted into the emulator's hosting environment (3) if a driver for the original filesystem is available. Under these preconditions, required hardware drivers depending on the emulator chosen can be added or replaced. After that, the container can be migrated into the appropriate format accepted by the emulator. All these actions are reversible. In (3) it would even be possible (just as a thought experiment) to re-insert the original driver set and dump the container file back to a hard disk compatible to the original machine. From this point on the system migration is finalized, as the rendering of the preserved system in the emulator (4) is not part of the migration function.

In a perfect world part B of the migration function is an identity transformation – one machine is simply replaced by a compatible one. This conforms to reality to a certain extend as e.g. X86 operating systems and applications execute on a wide range of different types of hardware and yield (exactly) the same results on similar input. Different machines of the same architecture can be specialized for a certain purpose, like CAD, scientific computing or gaming. Thus, depending on the requirements of the preserved artefact transformations from one type of peripheral card to another one are irrelevant. The same is true for e.g. IP network connections. The actual type of physical transport does not really matter when requesting a particular web page. Nevertheless, emulators are not perfect and might not fulfill all requirements perfectly.

Possible characteristics of migration functions for different systems were derived in an iterative process of previous experiments. While general mappings for the various computer architectures can be easily made, a couple of practical tests need to be run to ensure that a particular operating system can be run on a particular hardware emulator. From this a mapping matrix of working original environment emulator combinations can be drawn to support the emulator selection and configuration process (B). Finally, the procedures for disk imaging need to be verified to ensure an identity transformation from the permanent storage to it's container file representation.

3. HOW TO EVALUATE?

An established way to verify a successful migration is to render the artefact in a compatible viewer [4]. This action corresponds to step 4, to boot the preservation target in the chosen and configured emulator. If the transformations were identity mappings, then the rendering quality of the object solely depends on the renderer. The immediate results varied in the experiments run. While various old Apple Macintosh systems (OS 7.5 on M68k, OS 8.5, 9.0 on PPC) were immediately usable in either BasiliskII or SheepShaver more was to be done for X86 systems to successfully complete step (4) and start the environment. A couple of changes to the original configuration (of various Windows, OS/2 2.1, Linux systems) were to be made, e.g. that the original desktop screen resolution in OS/2 got demoted to standard VGA as no compatible driver is available to support the original desktop size. Another issues in OS/2 (client-server network consisting of at least two machines) was the network connection, as the original TokenRing network was mapped to Ethernet as no emulated hardware equivalent was available in any emulator. Other problems were the harddisk controller drivers, which needed to be replaced to successfully start e.g. a Windows 2000 system again [3]. Thus, formally, a couple of (relevant) changes were made to the original in-

[1]This is not an entirely new problem as software is usually made to run on a wide range of different types of hardware (e.g. the huge and widely varying X86 family), which may not all have identical features.

stallations to re-run the original environments again.

The various experiments have established knowledge on particular systems regarding setup and configuration particularities. Much of it should be re-usable on similar tasks, simplifying the imaging of systems of the same type as the hardware environments are usually fixed and limited e.g. regarding available operating systems. From this, a matrix can be created about which system can be preserved in which way. This can be used as input for part B of the migration function.

Quality Assurance.

Just as a TIFF or PDF can be roughly quality assured by rendering and comparing the original and migrated version, the same should apply to system migration. If the machine boots up properly to a certain state which allows user or machine interaction and the start of the artefact or business process in question, the operation can be assumed to have succeeded. Optimally, the original user of the system or number of systems checks upon this as she or he should have the best knowledge of the original behavior [7].

The intuitive approach would suggest, that the reduction of screen resolution or the exchange of network adaptors, even changing from TokenRing to Ethernet in the full system preservation cases should have no direct influence on the rendering outcome of most artefacts. Nevertheless, e.g. a non-functional network like experienced when trying to use the OS/2 client-server environment, would challenge the migration result. For applications like CAD, 3D, computer games and art a different sound card and graphics adaptor may have more significant influence compared to applications like preserved scientific environments or famous personae laptops.

As the objects considered for system imaging are much more complex, the measurement of success and completeness of the workflows should be attached to the different subfunctions. While the quality assurance for A_1, A_2, A_3 is discussed and solved by projects like Bitcurator[2] and the B part of the function can be refined through experimenting, the rendering part (4) is more complex.

The renderer, in our case the emulator, usually affects lower layer components of the hardware-software-stack like the CPU or hardware drivers of the operating system. This should not, but definitely can affect the object of interest, which is usually a certain application like a piece of digital art, computer game, scientific workflow or business process. Just think of a system which received a CPU update to a Pentium with the f00f bug in it. Such differences are hard to spot and difficult to test automatically. As most probably the concepts of traditional migration can only be partly applied, new metrics are to be found to describe complete systems in a way which allows proper (automatic) verification.

4. DISCUSSION

Formalization of the associated workflows helps to better understand and describe the procedures. Migration and object rendering are distinct operations. The proper migration of the original environment is a necessary condition for future access. The sufficient condition is a good enough emulator to fulfill the requirements on authentic rendering and future user experience. This understanding helps to focus future emulator research, development, and quality control both on migration processes and computer platform emulators. Research and technical questions:

- Does the general approach makes sense at all?
- Can system migration be formalized to achieve verifiable results?[3]
- How to prove the authentic reproduction of the environment? Would it be possible to define broad enough test sets (which ideally can be run automatically)?
- How to preserve a complex scientific environment of a non-standard system?
- Is it be possible to produce standardized workflows for environment preservation?

Additionally, a cost model should be developed to properly open system migration and emulation of original environments based preservation strategies to proper preservation planning and cost calculation. For every kind of migration a certain amount of resources and associated costs per object are required to confirm that all information and context is preserved post-migration. The total costs can be significantly reduced by moving down in the software-hardware-stack as the number of objects to be considered decreases.

5. REFERENCES

[1] C. Becker, A. Rauber, V. Heydegger, J. Schnasse, and M. Thaller. A generic xml language for characterising objects to support digital preservation. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 402–406. ACM, 2008.

[2] C. Becker, A. Rauber, V. Heydegger, J. Schnasse, and M. Thaller. Systematic characterisation of objects in digital preservation: The extensible characterisation languages. *J. UCS*, 14(18):2936–2952, 2008.

[3] E. Cochrane, D. von Suchodoletz, and M. Crouch. Database preservation using emulation – a case study. *Archifacts*, (April):80–95, 2013.

[4] M. Guttenbrunner and A. Rauber. Evaluating emulation and migration: Birds of a feather? In H.-H. Chen and G. Chowdhury, editors, *The Outreach of Digital Libraries: A Globalized Resource Network*, volume 7634 of *Lecture Notes in Computer Science*, pages 158–167. Springer Berlin Heidelberg, 2012.

[5] M. G. Kirschenbaum, E. L. Farr, K. M. Kraus, N. Nelson, C. S. Peters, G. Redwine, and D. Reside. Digital materiality: preserving access to computers as complete environments. In *6th International Conference on Preservation of Digital Objects (iPRES2009)*. Stanford University, California, 2009.

[6] G. Knight and M. Pennock. Data without meaning: Establishing the significant properties of digital research. *International Journal of Digital Curation*, 4(1):159–174, 2009.

[2]See the project homepage at `http://bitcurator.net`.

[3]The (successful) boot tests on the migrated original systems can be seen as black-box tests, where non-obvious issues introduced via CPU speed and type, RAM size and other base system components may not surface.

[7] K. Rechert, I. Valizada, D. von Suchodoletz, and J. Latocha. bwFLA – a functional approach to digital preservation. *PIK – Praxis der Informationsverarbeitung und Kommunikation*, 35(4):259–267, 2012.

[8] I. Welch, N. Rehfeld, E. Cochrane, and D. von Suchodoletz. A practical approach to system preservation workflows. *PIK – Praxis der Informationsverarbeitung und Kommunikation*, 35(4):269–280, 2012.