A Virtual Experience on the Very First Italian Computer

GIOVANNI A. CIGNONI, FABIO GADDUCCI, and STEFANO PACI, Department of Computer Science, University of Pisa, Italy

Despite their temporal proximity, the technologies of the early computers are far from us. Yet, they are part of the history of science and technology, and they deserve to be studied and popularized. Being machines made to run software programs, they should be exhibited running. Unfortunately, old machines still in working condition are extremely rare. Restoring or rebuilding an old computer is a hard, expensive task: the original components are rare, and the technology is forgotten and sometimes lost. The research needed to re-understand those computers has to adopt experimental archaeology methods: rebuilding old hardware/software requires proceeding by hypotheses and experiments.

However, a rebuilt or restored computer is a unique exemplar and a precious specimen: it is not suitable to let people interact with it. A more flexible solution is to use software simulations. First of all, simulation is a valuable tool to carry out the experiments needed to study past technology. Second, the simulators are virtual replicas that let people fully understand the old machines by interacting with them without jeopardizing those precious relics of the past.

This article presents the virtual rebuilding of the first computer made in Italy: the Macchina Ridotta (MR) of the University of Pisa. The MR was dismantled after few months of intensive usage to cannibalize the materials for a second computer. As a consequence, the MR disappeared from later chronicles and for many years was ignored by historians. When we attempted to reconstruct the MR history, we found that the survived documentation was far from complete. Simulation proved to be the key tool to support the experimental approach adopted for understanding the MR technology, rebuilding it, and assessing its achievements. The MR simulator is now used at the Museum of Computing Machinery of Pisa as a mean to truly experience a working session on the MR—a typical computer from the 1950s. The exhibit and the workshops, by exploiting the accurately reproduced characteristics of the MR, address popularization of computer science from several perspectives: from technological mechanisms to scientific foundations, passing through the representation of computers in popular culture.

Categories and Subject Descriptors: I.6.3 [**Simulation and Modelling**]: Applications; K.2 [**Computing Milieux**]: History of Computing—*Hardware*; K.3 [**Computing Milieux**]: Computers and Education

General Terms: Experimentation, Design, Verification

Additional Key Words and Phrases: Computer history, experimental archaeology of computer science, virtual replicas

ACM Reference Format:

Giovanni A. Cignoni, Fabio Gadducci, and Stefano Paci. 2015. A virtual experience on the very first Italian computer. ACM J. Comput. Cult. Herit. 7, 4, Article 21 (February 2015), 23 pages.
DOI: http://dx.doi.org/10.1145/2629484

Authors' addresses: G. A. Cignoni, F. Gadducci, and S. Paci, University of Pisa, Department of Computer Science, largo Pontecorvo 3c, 56127 Pisa, Italy; emails: cignoni@di.unipi.it, gadducci@unipi.it, and s_paci@di.unipi.it.

DOI: http://dx.doi.org/10.1145/2629484

1. INTRODUCTION

Computers are ubiquitous components of today's technology and participate in our everyday life. Although they made their appearance not very long ago, their history is dense and largely unknown. Such a history deserves to be studied and popularized. Primarily for its own sake, and for the advice that the successes and failures of the past may carry to the modern pratictioneers. Often, it may still be narrated by the same protagonists of those events (e.g., see the opening remarks in Tatnall [2012]).

Besides their interest for the researchers, old computers are fascinating machines for many people; they can be used to raise interest in computer science and technology and even become a way to teach how modern computers work, as the principles and basic operating mechanisms are still the same (see Akera and Aspray [2004] for a comprehensive survey, Harms and Berque [2001] for a hands-on experience on a computer from the 1970s, Impagliazzo and Samaka [2013] for the most recent entry on the use of history as a teaching tool, and Rojas and Hashagen [2000] for one of the few volumes devoted to comparative studies on the architectures of the first computers).

The seminal report entitled *The Public Understanding of Science* (*PUS*) [Royal Society 1985] opened a rich debate about science popularization. Among other issues, the *PUS* stated that the traditional role of schools has to be assisted by other institutions. Museums in particular can transfer knowledge by using captivating means and providing experiences that may not be perceived as "teaching." In this perspective, museums must not be conceived for people who already have some interest in science and technology but to attract the general public and to raise a new curiosity.

It is a commonsense statement that early computers have a natural appeal to a specific public: informatics practitioners, technology lovers, those who become sentimental in front of the computer they worked or played with as youngsters. There is also a growing audience of *retrocomputing* enthusiasts and collectors. For these people, a traditional exposition of old machines with a tagline full of technical data suffices. However, this is not the public we need to attract.

Drawing the attention of visitors is a major concern for scientific museums. The topic has been debated for a long time, and it was the focus of a recent IFIP congress on the history of computing [MHCR 2013]. A typical "trick" relies on providing the visitor with a sense of awe: big technological artefacts are immediately perceived as being out of the ordinary, and exploiting the "sense of wonder" in exhibitions (e.g., see Greenblatt [1991]) has already been applied to science and technology museums [Blyth 2013]. Computers of the past, particularly those made in the 1950s and 1960s, are often such kind of machines, with an additional, distinctive old-fashioned look. A running old computer with lights, changing screens, spinning tapes, and so on may truly act as an attraction point.

Unfortunately, early computers are extremely rare today, and those preserved are usually not in working condition. In many cases, some historical machines simply no longer exist, and thus the only viable option is to rebuild them [Sale 1998]. The effort to restore (and maintain) an old computer in working condition is impressive. Apart from the materials, the main challenge in restoring (or rebuilding) and operating an old computer is the amount of research needed to understand its technology. As an example, the Zuse Z3 rebuilding project required 5 years of preparatory work [Rojas et al. 2005]. Nevertheless, due to historical and educational interests, several museums have succeeded in the enterprise and now exhibit working old computers (see Section 2.2). But, of course, visitors are not allowed to have their hands on such precious relics.

Software simulation is a worthy alternative to recreate virtual, albeit realistic and accurate, replicas of old computers that can be safely used in installations, thus letting persons interact with them to get a full view of past technology both from the hardware side (the working machine) and the software side (the running programs). As an additional benefit, being software themselves, the virtual replicas can also be easily distributed, allowing visitors to continue their experience at home. (For a successful

experience with the seminal ENIAC, see Zoppke and Rojas [2006] and the references therein for further case studies.) Most importantly for the historians, simulation is also a valuable tool for understanding or even rediscovering the technology of an old computer.

This article presents the results of the HMR [2013] project, which was carried out at the Department of Computer Science of the University of Pisa. HMR stands for *Hacking the Macchina Ridotta*, with the 1957 Macchina Ridotta (Smaller Machine, MR) being the first Italian computer [Cignoni and Gadducci 2012]. The primary goal of the project was to virtually rebuild the MR, as the original was dismantled in early 1959 to reuse its materials in the construction of another computer. Due to the scarce documentation retrieved, HMR had to adopt experimental archaeology methods to understand the technology of the MR and to rebuild it—hence, the reference to the hacker culture and its curiosity about investigating computer details. Thus, we used modelling and simulation as the key tools to formulate and validate the rebuilding hypotheses. The in-depth technological assessment of the MR needed to develop the simulator led towards a more faithful reconstruction of the events surrounding the successful building of the first Italian computer. As a further result of our project, the simulator is now used to show the MR at the Museum of Computing Machinery in Pisa.

Our article presents these achievements, focusing on the two advantages of modelling and simulation as a way to interact with the past. It allows researchers to validate rebuilding hypotheses and the public to experience the look and feel of a computer from the 1950s. Section 2 introduces the experimental archaeology methods as applied to computer science. Section 3 describes simulation as a tool for historical research and presents our modelling and simulation techniques. Section 4 summarizes the HMR results on the rediscovery of the first Italian computer. Section 5 shows the use of the MR simulator to popularize and teach computer science.

2. EXPERIMENTAL ARCHAEOLOGY OF COMPUTER SCIENCE

According to the largely agreed on definition of *computer* as a Turing-complete machine with a stored-program architecture, the year zero of the computer age has to be set in 1948 when the *Small Scale Experimental Machine* (better known as the *Manchester Baby*) successfully ran its first program [Burton 2005]. If we want to include other renowned electronic ancestors like the ENIAC [Marcus and Akera 1996] or the Colossus [Flowers 1983], or electromechanical devices like the Zuse machines [Rojas 1997], we may backdate the beginning by a decade. Even counting far remote devices such as the tabulating machines [Fierheller 2006], we still stick to the 20th century.

Given such a short timespan, and considering that the discipline developed in times of which the risk of loss of documents and evidences is supposed to be lower (somehow downgrading the urgency for their research), at first it may seem premature to talk about the "archaeology" of computer science.

2.1 How Old Are Old Computers

Computer science has been always characterized by its extremely rapid progress. Technologies have a very short lifecycle: they arise, succeed for few years, then quickly become obsolete, and eventually are replaced and forgotten. Modern hardware and software engineering experts are distant from the technologies used decades ago. Old machines appear as arcane devices to today's historians: they may figure the general picture, but it is difficult for them to fully understand the details needed to restore an old computer to working conditions or rebuild it from scratch.

Often, only a few parts of the original machine survive, and frequently they are in bad condition. The original documentation, recovered by digging into the archives, may not be complete. When it is available, it uses obsolete notations and has to be deciphered like a lost language. If all of these considerations are taken into account, it seems appropriate to use the term *archaeology*.

Because of the many gaps in the knowledge gathered from recovered documents and evidences, the reconstruction of a computer from the past is like solving a big technological puzzle. The reconstruction is a continuing challenge: it proceeds by hypotheses that must be experimentally proved.

In fact, *experimental archaeology* is a well-defined research methodology. It concerns the study of technologies of the past, carried out by attempts to recreate and use ancient artefacts. The *Damascus Steel* is one of the best-known case studies [Reibold et al. 2006]. Few swords forged with this technology remain, but the metallurgical technique is lost and the debate is open: is Damascus steel different from Indian *wootz* steel or Japanese *folded* steel? Another interesting case is the *trebuchet*, a huge siege machine whose tricky physics has been lost in time [Hansen 1992].

With respect to other technologies of the past, old computers present the same characteristics that make the adoption of experimental archaeology methods worthwhile. There is only one, positive difference: given the relative proximity, it is sometimes possible to involve the original developers in the rebuilding challenge. It is not possible to ask them to remember the many details of such complex machines, but they can help to set the boundaries of the knowledge of the time and to assess the general soundness of the rebuilding hypotheses: would they have made the same choices?

2.2 Restoring and Rebuilding Projects

Many research initiatives exist that have already applied the methods of experimental archaeology to recover and bring to life a few of the preeminent relics of computing history. Without any claim of completeness, we briefly give an account of some of the most intriguing.

Many UK projects are sponsored by the Computer Conservation Society [CCS 2013]. Their results are shown, for example, at the National Museum of Computing [TNMOC 2013], the London Science Museum [SM 2013], and the Manchester Museum of Science and Industry [MOSI 2013].

Colossus was the machine used at Bletchley Park during WWII to decrypt the Axis transmissions encoded with the *Lorenz SZ42* [Copeland 2004]. Besides such a historical value, Colossus was one of the first electronic machines. However, its technology was almost lost: fearing information leaks, the Colossi and their documentation were destroyed during the Cold War. The reconstruction started in 1993 based on the few surviving documents and the memories of scientists and engineers [Sale 1998]; the replica has been functioning since 2007. In 2011, the rebuilds of the *Tunny Machine*, an electromechanical reverse-engineered Lorenz, and the *Heath Robinson*, the electromechanical ancestor of the Colossus, were completed [TNMOC 2013]. The Bletchley Park Trust Museum [Bletchley Park 2013] exhibits a working replica of the 1940 *Bombe*, the electromechanical device used to decrypt the Axis transmissions encoded with the *Enigma*. The Bombe is not a true computer, but a relevant precursor whose rebuilding project started in 1996 and ended in 2007.

The National Museum of Computing [TNMOC 2013] exhibits a number of working old computers, the result of impressive restoration (and continuous maintenance) projects. Among them, it is worth citing the 1951 *WITCH*, based on the fascinating *Dekatron Tubes* and restored in 2012, and a 1961 *Elliot 803B*. The museum also is carrying on a project to rebuild the 1949 Cambridge *EDSAC*.

In 1998, the Computer Conservation Society completed the reconstruction of *The Baby* to celebrate 50 years of stored-program computing [Burton 2005]. Since then, the working replica has been part of the exhibits of the Manchester Museum and is regularly demonstrated to the public—being 16 years old, the replica is becoming an old computer as well.

In 1985, the London Science Museum started a project to build the Babbage's *Different Engine No.2*; the machine is now part of the permanent exhibition [Swade 2005]. It is worth noting that this machine is not a replica: Babbage was never able to build it; however, the 1985 project proved that it would have

been feasible with the engineering capabilities of the Victorian age. A project attempting to build the more complex Babbage *Analytical Machine* is under way [Plan28 2013].

In the United States, the restoration of an original 1960 *Digital PDP1*, probably the most innovative and advanced commercial computer of its time (featuring a graphical display with a point and click device), started in 2004. The machine is now on display at the Computer History Museum in Mountain View, California [CHM 2013], and it is the protagonist of special events, like the periodical runs of *Spacewar!,* the very first interactive computer game developed at MIT on a PDP1.

As well from the United States, it is worth mentioning the recovery of the complete blueprints of the *Block I AGC*, the guidance computer installed on the control module and lunar lander of the *Apollo* missions. This project's original documentation is published online, with step-by-step instructions for building one's own replica of the Block I [NASA Office of Logic Design 2013].

The German 1941 *Zuse Z3* was destroyed during a bombing raid. Its relevance is due to a recent proof that, under particular assumptions, it can be considered the first Turing-complete machine [Rojas 1998]. A first replica was built by Zuse himself in 1961, which now resides at the Deutsches Museum [Deutsches Museum 2013]. A second one was made in 2001 [Rojas et al. 2005]. Lastly, a third full-featured replica was completed in 2011; there is a plan for exhibition at the Konrad Zuse Museum in Hünfeld, Germany [Zuse 2013; KZM 2013].

Among French projects aimed at putting old computers back to working conditions, we mention the *Digital PDP 9* restoration carried out by the Aconit Association [Aconit 2013].

Finally, as an example of a project in the related field of analog calculators, procedural computer graphics has been used to rebuild astrolabes, both as virtual replicas and as a way to engrave actual ones by driving numerically controlled machinery [Zotti 2008].

Our rebuilding project started in 2006 at the Department of Computer Science of the University of Pisa. The aim of HMR, since the beginning, has been to virtually rebuild the MR. Apart from the challenge of bringing back to life the first Italian computer, the project was motivated by the belief that without an actual (simulated) machine to play with, it is difficult to correctly assess its technological characteristics and evaluate the historical context surrounding its realization.

3. MODELLING AND SIMULATION FOR VIRTUAL REBUILDING OF OLD COMPUTERS

The idea of virtually rebuilding the MR stemmed from a background in modelling and simulation. Methods and tools used by the HMR project are the results of a parallel research activity. From this perspective, we consider the virtual MR as a case study in the modelling and simulation of hardware systems. This section shows how these techniques can be useful for rebuilding virtual replicas of old computers, using as a running example our experience with the two versions of the MR: the first design dated 1956 (MR56) and the second one dated 1957 (MR57; see Section 4 for historical details). As an introductory step, our presentation begins with a discussion on terminology.

3.1 Simulation versus Emulation

Usually, a virtual rebuild of an old computer is termed an *emulator*, even if we prefer to use the more general term *simulator*. The difference between the two terms is subtle. According to the *IEEE Standard Glossary of Software Engineering Terminology* [IEEE610 1990]

—A simulator is a device, computer program, or system that behaves or operates like a given system when provided a set of controlled inputs.

—An emulator is a device, computer program, or system that accepts the same inputs and produces the same outputs as a given system.

For a given input/output interface, an emulator is thus indistinguishable from the real thing. In some cases, virtual rebuilds of old computers are actually emulators. The renowned *Multiple Arcade Machine Emulator* [MAME 2013] (in all its variants and portings) is an emulator of the hardware of the cabinets used for penny arcade video games, working with the code of the original game ROMs.

A simulator, on the other hand, benefits from more degrees of freedom, as it implements a model of the system that, depending on convenience, can ignore or approximate a number of details. As far as the model is clearly declared, the simplification with respect to the real thing is not a negligence. In the practice of modelling and simulation, it is fundamental to focus the effort on the characteristics of the system of which we are interested to study. When applied to the virtual rebuilding of old computers, this degree of freedom is an additional opportunity. For instance, considering our virtual replica of the MR57, we approached different rebuilding issues according to different criteria:

—*Machine language*: The virtual MR57 is a true emulator because it interprets every program written in the MR57 machine language and produces exactly the same outputs of the original MR57.

—*Timing*: The virtual MR57 is a real-time emulator with a measured error; the simulator runs on ordinary PCs in a multitasking environment, so it is not possible to guarantee that each single machine instruction is executed with perfect timing. However, the simulator displays the current cumulated error between real time and the simulator time; tests show that such an error remains in the order of 1/100 s even in several-hour-long runs.

—*I/O devices*: The binary signals at the connections are emulated with the preceding time limitations, but the peripherals of the MR57 (tape readers, tape punchers, and teletypewriters) are simulated in a very simple way that does not consider many of the mechanical characteristics of those devices.

—*Control panel*: The MR57 console has been simulated as a graphical interface. Layout and look are accurate, even if the proportions of the single items were adapted to ease user interaction through mouse/touch devices. Great care was devoted to simulate the lights to represent with a reasonable degree of realism their actual and characteristic behaviour.

Modelling a machine at different levels of detail is a valuable convenience when we want to simulate an old computer both for research and popularization purposes. In the MR57 case, for instance, the I/O devices are at the moment grossly simulated because we are still researching about their behaviour, especially in the interesting (yet undocumented) cases of errors and edge conditions.

Another example is related to obtaining an accurate visual look and feel of the control panel: the lights needed a more fine-grained simulation than the switches. For the gas-filled, cold-cathode *Z50T* triodes used for the panel lights, we had to model the ionization states using a discrete approximation with timing accurate to the microsecond. The switches are instead less relevant in the virtual representation of the panel: the interaction is forced through a mouse/touch interface that has no physical feedback, so we did not model details like the mechanical characteristics of the internal spring of the switches.

3.2 Why Simulation

To popularize the history of computing, at least three questions need to be answered: how hardware worked, which software was run and for which purposes, and how people interacted with those machines. An exhibition of switched-off computers is a poor description of such a history—and a gloomy sight. A computer should always be shown running; otherwise, it represents at most half of the story, the one limited to the relics value. Furthermore, switched-off machines often lead to overrating the exterior design, which in some cases is appreciable, but an add-on to the concentration of science and technology inside.

Old computers can also be used to teach computer science. They are odd enough to arise curiosity, although the principles and basic mechanisms are still the same. In addition, they are simple and transparent: unlike modern computers, they expose all of their inner workings without hiding them behind user-friendly interfaces. Therefore, they can be understood down to their details.

To answer the questions mentioned earlier, restoring original machines to working conditions and showing them running is an appealing solution. However, switching on the original machines exposes these relics to damage. Moreover, substituting faulty parts to achieve and maintain working conditions may compromise authenticity. Building hardware replicas is a solution to avoid jeopardizing the originals. It is also a challenging enterprise when the original has been lost (or was never built).

Working originals and hardware replicas are worthwhile to museums: they make it possible to set up appealing exhibitions and to organize periodic demonstrations or sporadic switch-on events of popular acclaim. They are great assets to capture both the public and the attention of the media.

Yet hardware does not last forever, and many materials, such as particular vacuum tube types or germanium transistors, will be more and more difficult to retrieve as time goes by. Replicas and restored machines have a limited lifespan or, due to the substitution of original components with equivalent modern counterparts, are destined to be increasingly different from the originals.

The best way to experience an old machine is to personally interact with it and its unfriendly user interfaces. For example, for researchers investigating the delay of an old software project, lots of insight may come by trying to code, verify, and debug a program in the same conditions as the pioneers. For the general public, the hand-on experience, even on simplified tasks, is appealing and rewarding.

However, working originals or replicas are huge pieces of hardware and are tied to their physical locations. Researchers, and generally anyone who wants to see them, has to visit those museums where they are on display. With respect to the purpose of popularization of computer science, this fact reduces the number of potential users. Moreover, originals and replicas are valuable items: it would be unthinkable to let the visitors of a museum or some students have their hands on such precious relics. Such objects are used for demonstrations, but people can simply attend and not interact.

On the other hand, simulators are software, and they can be exhibited and demonstrated in a museum, as well as infinitely replicated and (freely) distributed. In addition, smaller museums may demonstrate them. Simulators may be easily distributed using the Internet: the number of enthusiasts who download a simulator to play with at home is likely low, but the service is worth the effort if we consider the researchers and the schools. If a class is introduced to the simulator during a demonstration carried out at a museum, the experience can be continued using the simulator in a lab.

As in other fields, simulation can enhance the user experience with educational and entertainment purposes. (See, for instance, *USARSim* [Carpin et al. 2007], a robot simulator for teaching engineering, and *SuperCharged!* [Squire et al. 2004], an edutainment tool for electromagnetism physics.)

3.3 Modelling before Simulating

There are simulators for many computers of the past, from the earliest machines to the more recent ones. Few hardware restoring/rebuilding projects provided simulators of the target machines. As an example, during the Manchester Baby rebuilding, several simulators were developed and made available through the Internet [Napper 2013]. Those simulators were later used for programming contests issued for the Baby anniversaries in 1998 and 2008. Two of them, one in Java and one in *SystemVerilog* [IEEE1800 2009], are now freely available and used as case studies at the ECAD and Architecture practical classes at the University of Cambridge [2013]. As a side effect, simulators are able to mitigate the traditional rivalry among universities: the simulator of the Baby is used at Cambridge, home of the EDSAC, which in turn has its simulator at Warwick [EDSAC 2013].

With respect to a hardware rebuild, the development of a software simulator may seem like a simpler task. There are surely some relevant advantages—no need for either searching (and purchasing) old components or providing the adequate physical room for the assembly activities. However, the relevant part of the challenge is the same: digging in the archives, retrieving and comparing all information sources, and understanding the old technologies. Furthermore, in the case of a simulator, the required knowledge may be even more demanding. Once a physical component is retrieved—from a thermionic tube to a whole device like a teletypewriter—putting it in its place is relatively simple: plug it and it will do its work. On the other hand, to simulate the same component requires more activities: the in-depth understanding of the inner working of the component, the development of the component model at the chosen level of detail, the implementation of the model in a simulator, and the integration of the simulator of the component with the simulator of the whole system.

An additional benefit of the simulation approach is the resulting *model*. The relevance of the model is twofold. First of all, it is a due step in the development of the simulator: modelling is the first phase of a proper simulation project, and the model is both the specification at the required level of detail of the system to be simulated and the specification of the simulator as a software artefact. Moreover, since simulation is a discipline targeted to the study of systems that is not possible nor convenient to study in real life, the model is an effective replacement to describe, to document, and to understand the system itself.

In our context, the model is a good way to perpetuate the specifications of the computers of the past in modern terms. Even if the original documentation is available, it uses outdated terminology and language and needs an interpretation effort to figure out what those machines were like. A model of the old computer in a modern formal language, like the SystemVerilog for the Manchester Baby, acts as a sort of *Rosetta Stone* and can help further research, as the language is a recognized standard [IEEE1800 2009]. For instance, descriptions in a common notation are good starting points to compare architectures, solutions, and performances of different machines from the same period.

In the HMR project, as a specification language for the MR we used UML [2013] and SysML [2013]. The choice originated from the results of another research project about modelling and simulation [Cignoni and Paci 2012]. UML is for specification of software systems; SysML is an UML specialization targeted to general systems. We exploit the close relationship between the two languages to specify a formal model of the system to be simulated and then to interpret such a model as a software specification able to drive the generation of the simulator source code. Results of the research were applied to a few real case studies and are part of the teaching activities carried out at the simulation course held for the applied computer science degree of the University of Pisa.

With respect to SystemVerilog, one of the advantages of UML/SysML is that the model can be read at different levels of detail. A complete and fully specified model is needed to automatically generate the simulator source code, and of course it is a complex model. However, during the development of the model, the ability to proceed by refinements is valuable. Moreover, different views are useful when the model is adopted to describe the inner working of an old computer, addressing purposes ranging from popularization to in-depth technical assessments. In the following, we show two of the UML/SysML diagrams composing the model of the MR57 (and its simulator).

3.4 Excerpts from the MR57 UML/SysML Model

Figure 1 shows the state machine diagram of the MR57. Details about events and conditions that control the transitions are hidden, yet the diagram tells a number of facts about the MR57. Arcs between states are labelled by different types of events that trigger the transitions:

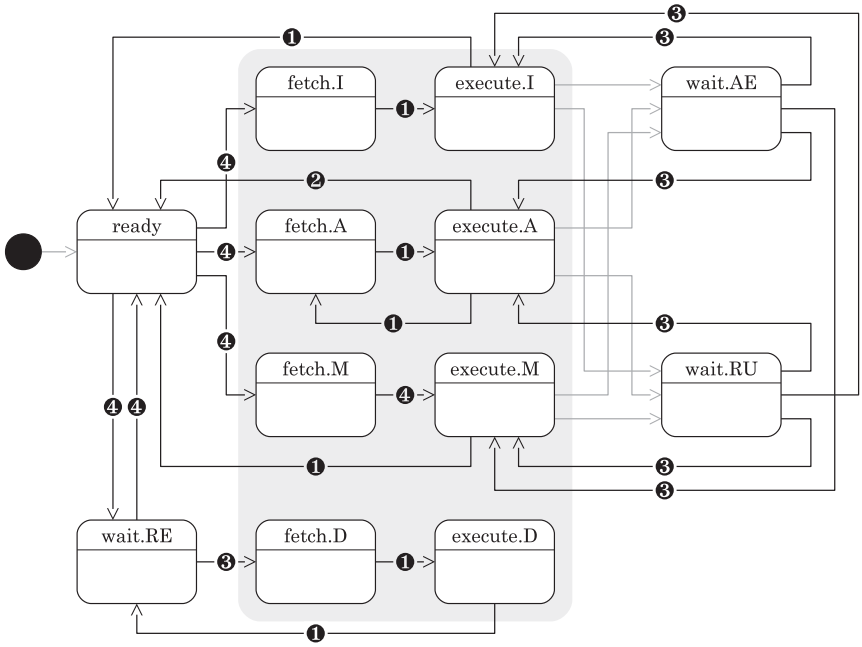

Fig. 1. A (simplified) state diagram from the model of the MR57.

—Machine clock events, occurring at the clock pulse every 4 or 8 μs when the clock pulse generator is enabled, labelled with ☆

—Programmed stop events, occurring as a result of the execution of the stop instruction or when a breakpoint match happens, labelled with 🕐

—I/O device-ready events, occurring when an I/O device terminates to perform a task such as printing or reading a character, labelled with 🕑

—Manual control panel events, occurring when the user operates the MR57 console, typically pressing the start button or setting the machine mode switches, labelled with 🕒.

The greyed arcs represent transitions that depend on the execution of machine instructions for I/O operations. The states included in the grey background may be collapsed in one general *running* state that hides the alternation of fetch/execute phases, thus providing a simpler view of the MR57.

The comparison of the two MR versions—the MR56 and the MR57—give us another example of the use of UML/SysML models for assessing old computers. At the same level of detail, the difference between the MR56 and MR57 state diagrams relies on the three states on the bottom (namely wait.RE, fetch.D and execute.D), which are not present in the MR56. These states concern the direct access mode that makes the input devices (especially tape readers) able to write directly in the MR57 memory, a MR57 improvement that makes the program loading much more practical and, in particular, elegantly solves the issue of booting the system software at machine startup.

Figure 2 shows the block diagram that describes the hardware components of the MR57. It is directly derived from one of the retrieved MR57 blueprints [MR/S/2 1957]: the names of the object instances correspond to the ones used in the original documents. The diagram includes the I/O streams (which are modelled as queues in our UML/SysML notation); the registers (nominally queues with a single position); and subsystems such as the binary adder, the memory, the clock generator, and the

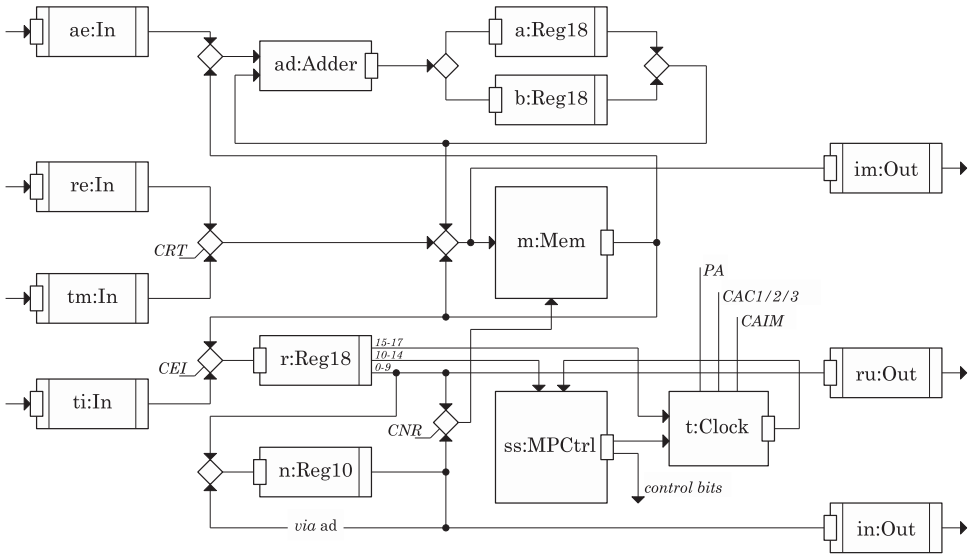

Fig. 2. A (simplified) Block Diagram from the model of the MR57.

microprogrammed control. Some of the logical switches are controlled by the internal *control bits*, and others by mechanical switches on the MR57 control panel (*CAC1/2/3*, *CNR*, *CEI*, *CAIM*, and *PA*). Some of the I/O streams are connected to I/O devices such as punched tape readers, teletypewriters, and tape punchers (*ae*, *re*, and *ru*), and others are connected to keyboards and indicators (*tm*, *ti*, *in*, and *im*) on the control panel. For more details about the MR57, see Sections 4 and 5.

The diagrams in Figures 1 and 2 are presented as examples of our modelling approach. They are only a little part of the diagrams that make the complete MR57 model, and for the purpose of this article, they are shown in a simplified view that omits many details and, in some cases (like the "*via* ad" and "control bits" shortcuts in Figure 2), even disrespects the UML/SysML syntax.

3.5 The MR Virtual Replicas

Even if hardware complexity cannot be avoided, the advantage of UML/SysML with respect to other languages, such as VerilogSystem, is that it provides a graphical notation that makes it possible to adapt the complexity (hiding part of it) to the audience to which the description is targeted.

The complete models of the two MRs include other types of UML/SysML diagrams such as class and activity. At their maximum degree of detail, the diagrams are quite elaborate, for instance, regarding the conditions and the actions specified on the transition arcs of the state diagram or the many connections among objects in the block diagram. According to our method of modelling and simulation, at this level of detail the UML/SysML specification can be used to generate the source code of the simulator.

However, in the case of the MR virtual replicas, this ability was not fully exploited. The code generator produces source code for a traditional batch simulator that outputs data to be later analysed. A simulator of this kind was useful in the research phase, when simulation was needed for studying, assessing, and validating the rebuilding hypotheses. However, it is not adequate for teaching and popularization, when the main goal is to have people interacting with the replica in a more immersive way. To develop real-time virtual replicas equipped with interactive interfaces, the simulator code had to

be manually reworked. The main interventions were the integration with the graphical user interface and the addition of timing management to achieve real-time accurateness, and to couple the simulator with the refresh cycle of the graphical interface. Other issues concerned code optimization, which was required to have a simulator run smoothly on ordinary hardware. In particular, the general-purpose event queue used by the generated code was replaced by a custom solution that takes advantage of the particular kind of events that characterized the MR models.

The MR56 simulator was the first to be developed. The documentation of the first MR design was fully retrieved, including an early listing of the system software. The simulator was primarily addressed to the emulation of the machine language, and it was used to "restore" its software (see Section 4). Written in Java to achieve platform independence, it provides a graphical user interface and a simplified simulation of the control panel, the tape reader, and the teletypewriter.

The MR57 simulator was a more difficult challenge. Despite that the MR57 was the version of the computer actually built and used for about half a year for computation services, the surviving documentation is far from complete. Simulation played a major role in the validation of the rebuilding hypotheses, and the simulator was able to also address some low-level issues, such as evaluating the performance benchmarks claimed by the MR57 designers or understanding the special operation modes that let input devices write directly in the memory. The simulator is written in C++ to achieve maximum performance. It is designed to be portable; for instance, for the graphical interface, it uses the cross-platform *WxWidgets* library [WxWidgets 2013], but so far only Linux is supported (Debian [2013] and Ubuntu [2013]) distributions. It provides a graphical user interface for the control panel with an accurate representation of the look and feel of the lights.

4. CONTRIBUTIONS OF THE VIRTUAL REPLICAS TO HISTORICAL RESEARCH

The *ELEA 9003* (or *Elaboratore Elettronico Aritmetico*, whose meaning is Arithmetical Electronic Computer) developed by Olivetti in 1959/60 and the *CEP* (*Calcolatrice Elettronica Pisana*—i.e., Pisa Electronic Computer) built by the University of Pisa in 1961, have for a long time been considered the first Italian digital computers. The latter was the final outcome of a long project carried out from 1955 to 1961 by Tuscan University, with the substantial participation of Olivetti.

The birth of the CEP is part of the founding myth of Italian computer science, and several authors have contributed to report the events in this way (see De Marco et al. [1999] and Parolini [2008], among others). Notably, the CEP project delivered a first fully functional computer already in 1957: the MR that we introduced previously. However, the relevance of the MR has been overlooked by previous researchers, and its accomplishments have been often underestimated—and sometimes plainly ignored. Actually, the MR was considered just the core part of the final CEP: its name, which in Italian may be interpreted as "partial machine," contributed to deceive the historians. The MR was rediscovered recently after a careful investigation of the technical documentation supported by simulation to assess the machine characteristics and help to understand all of its details.

A few persons who worked on the MR participated in our rebuilding project: Elio Fabri (mainly), Luciano Azzarelli, and Giuseppe Cecchini. They helped in the understanding of recovered documentation and in the reconstruction of missing parts of the MR blueprints. Given the complexity of the subject, they could not recall all the technical details needed for a proper reconstruction. In fact, what we did was a tentative-driven rebuilding that involved the protagonists as experts of old technology.

The resulting new reading of the early years of Italian computer science is summarized in the following sections. (See Cignoni and Gadducci [2012] for a detailed narrative that includes the proper acknowledgement of the historical and technological relevance of both versions of the MR.)

4.1 The CEP Project

In the early 1950s, the University of Pisa received (from the counties of Pisa, Lucca, and Livorno) large funding to build a synchrotron in Italy and to launch Tuscany as a prominent area for nuclear research. After the failing of the initiative, the Pisa Institute of Physics (currently the department of physics) suggested using the funding for building an electronic digital computer. The Nobel Prize Enrico Fermi was involved to second the proposal and win the resistance of the politicians and the doubts of parties inside the university (mainly the faculty of engineering).

In March 1955, the CSCE (*Centro Studi sulle Calcolatrici Elettroniche*—i.e., Centre for Studies on Electronic Computers) was formally established: Marcello Conversi, by then the director of the Institute of Physics, was appointed to be responsible for the centre. The CSCE was charged with the task of designing and building an electronic computer within 4 years: the target machine was named *CEP*.

Olivetti was involved in the CEP project since its inception. After previous attempts with other institutions, it likely considered the availability of funds as an opportunity not to miss: its willingness to be a partner of the CEP project allowed the firm to offer specialized personnel, skills, materials, and, in the final stages of negotiations, even a relevant direct financial investment. Formalized in May 1956, the collaboration began earlier with the participation of the Olivetti engineer Mario Tchou to the drafting of the 4-year plan for the project.

The CEP computer was completed in the first half of 1961, about a year and a half later with respect to the original plan. It operated for about 7 years and was subject to many enhancements. Despite being a remarkable machine (with features such as the microprogrammability and the mechanism of changing the instructions for passing parameters to subroutines), the computer world was radically changed over the years when the CEP was completed. The technology of the 1961 machine fully reflects the delay of the project and the consequent financial problems: although still an interesting device, it was not a state-of-the-art product. For instance, the vacuum tubes upon which the CEP was almost completely based were already being replaced by transistors. Additionally, things had changed in Italy. In 1961, more than 20 computers were already installed, a number of which were made by Olivetti. Indeed, the firm was fully reaping the fruits of its investments: the cooperation with the University of Pisa planted the seeds of the ELEA 9003, the first commercial Italian computer announced in late 1959, and the 6001, a modular computer able to fulfil a wider range of tasks, presented a year later. It is worth noting that both Olivetti products were fully transistorized.

Although the CEP computer arrived late, the experience was still rewarding for the university. The CEP project allowed the collection of a wealth of expertise and human resources, and these would grow year after year, resulting in the 1969/70 start of the degree in computer science, which by then was the only one in Italy and among the first in Europe.

4.2 The Primacy of the MR

It is noteworthy that CSCE delivered a fully functional computer—the MR—already in 1957. The first detailed design was delivered by the end of July 1956, the result of the work of a team of four researchers who started the CEP project: Alfonso Caracciolo di Forino, Giuseppe Cecchini, Elio Fabri, and Sergio Sibani. After a period of study and rethinking to improve the initial design, the MR was finally completed a year later: the revised design was apparently ready by April 26, 1957, yet the machine itself was announced as successfully working on July 24 of the same year.

In the following months, the MR was used to validate the solutions and to finalize the design of the "ultimate" CEP (*definitiva*, as it was called)—the machine that was planned as the final outcome of the project. More importantly, the MR was also used to provide computation services to other research fields outside the CEP project. In the few months of its life, the MR accounted for a

total of 150 machine hours "sold" for usage in external research projects. One of these projects exploited symbolic computation. At the time, the standard approach was instead numerical, whereas in this case, the MR produced the result as an exact expression. Its use for research and development of new programming techniques is further evidence of the versatility of the first computer built in Pisa.

As well, the MR was the machine used when the first educational activities of computer science in Pisa were held. In 1956, Elio Fabri offered an introductory course entitled "Programming of an Electronic Calculator": his experience as one of the MR designers was immediately used for knowledge transfer. Moreover, in early 1958, the National Institute for Nuclear Physics detached at the CSCE four researchers from its headquarters of Milan, Padua, Pisa, and Rome to learn how to use the MR, which at that time definitely was the most advanced machine in Italy.

Quite interesting is the comparison of the MR with other machines of its time. The 1957 MR proved to be an up-to-date device. Indeed, from a technological point of view, the MR adopted state-of-the-art solutions that were not easy to find in similar computers at the time, such as the following:

—*Parallel bit processing*: Most machines were "serial"—the bits of a memory word were processed one at time. The MR was instead "parallel": it was able to elaborate all bits of a word at once.

—*Ferrite core memory*: In the 1950s, computers implemented the main memory using a variety of different technologies: magnetic drums, acoustic delay lines, and Williams tubes were the preferred solutions. CSCE researchers adopted the ferrite cores, choosing an emerging technology that was destined to dominate for a couple of decades.

—*Microprogrammed control*: The implementation of machine instructions by means of programmable microinstructions stored in a read-only memory yet easily writeable by external intervention is recognized as a result of the British EDSAC 2 Project described in the seminal paper of [Wilkes and Stringer 1953; Wilkes 1986]. MR designers quickly adopted the solution, although it was simplified by the small set of instructions and implemented by using a less sophisticated technology (diodes instead of ferrite cores), making the MR one of the first fully microprogrammable machines.

To get an idea of the peculiarities of the MR, it suffices to say that none of those three design choices occurred on the two other computers present in Italy when the MR began to be designed the American CRC102, bought by the Polytechnic of Milan in the late 1954, and the British Ferranti MK1, which was installed in early 1955 at INAC in Rome. In addition, on the performance side, the MR was a fast machine. Thanks to a careful fine-tuning, the execution time of the instructions was reduced with respect to the project estimates, making it superior to the other machines on the market, particularly the IBM 704 installed in the French headquarters of IBM. Even if the IBM machine had more memory, had more flexible I/O devices, and was equipped with a Fortran compiler, surpassing it on the most straightforward benchmark was a remarkable achievement.

4.3 On the Two Versions of the MR

The proper assessment of the relevance of MR, as well as the identification of two different versions of the machine, were the most interesting achievements of our technological investigation. Indeed, since it was possible to retrieve almost all documentation of the MR design dated 1956, this version was the first target of a virtual rebuild [Cignoni et al. 2009]. Unfortunately, this first design represented a different, simpler, and less interesting machine.

We suspected that these were wrong descriptions of the MR eventually built in 1957 by matching the blueprints of the 1956 design with the few documents (some photos and the user manual) that were

identified as belonging to the MR57. The technologically driven investigation and further research in the archives resulted in the proof of two versions of the MR designs. The smoking gun was a short report, written by Caracciolo and Fabri, that summarized some of the differences. From a historical point of view, the differences between the two versions are also an acknowledgement of the remarkable effort carried out by the CSCE researchers in the refinement of the MR design.

However, the retrieved information was not sufficient to fully describe the MR versions. We got a complete specification of the MR56 hardware, but gaps still remain in the documentation of its software. For the MR57, the situation is even worse: the reports provide many clues, but lots of blueprints are missing. The only viable solution was to use simulators as workbenches to test and validate the rebuilding hypotheses.

The simulator of the MR56 was used to restore the system software. The memory of the MR56 (1,024 words 18 bits long) had the higher addresses reserved for system subroutines which make fully operable the machine providing multiplication, division, decimal conversion for output purposes, and an amazing program loader from the tape reader. We retrieved only an early draft of such code that, loaded on the simulator, was discovered flawed by many bugs. We eventually fixed it to restore the virtual MR56 to full working condition. It is still unclear if the MR56 remained only an early design or it actually existed for a while as a first stage in the building of the MR. In any case, the virtual replica proves that the design was feasible and helps to appreciate the differences between the initial design and the final full-fledged machine.

The simulator of the MR57 was much more important to validate the rebuilding hypotheses formulated following the clues scattered in the few retrieved blueprints and reports. The evolution from the initial design was relevant, because with respect to the MR56, the MR57 had the following:

—More input devices; for example, a special input register was added that makes it possible to write directly in the memory, and it was possible to connect a second tape reader or a teletypewriter to this register, thus adding a keyboard to the MR user interface.
—More output devices, such as a second teletypewriter equipped with a tape puncher.
—A mechanism to set hot breakpoints in the machine code and to activate them at runtime using switches on the control panel.
—An improved control panel to give more feedback; in particular, a display was added to show the value of the program counter.

Due to input devices now able to write directly in the memory, the program loader subroutine was no longer needed and the MR57 system subroutines required less memory space. The machine language changed a bit, thus making the MR56 software not compatible with the MR57.

These features are now implemented in the current simulator of the MR57. Further enhancements on the side of accurate simulation of I/O devices are still a work in progress, due to the many gaps in the surviving documentation for the MR57: this is precisely where the use of simulators for experimental archaeology comes into full swing.

5. THE USE OF THE VIRTUAL REPLICAS AS TEACHING TOOLS AT THE MUSEUM

The virtual reenactment of the past environment is generally focused on the development of immersive devices largely based on 3D graphics, room filling displays, and so on [Laycock et al. 2008; CyArk 2013]. The aim of these installations is to present users with a plausible and appealing experience so that the resulting enjoyment plays in favour of teaching/popularization. This approach has been used also to reproduce virtual environments that visually recreate the large installations of old computers [Berry

Fig. 3. The only surviving photo of the MR Manual Control Panel. (Archives of the University of Pisa.)

et al. 2011]. However, we focus on accurately recreate the computing behaviour of the old machines, which for our objectives is of pivotal importance.

Furthermore, it is inherently difficult to provide immersive environments, as far as their association with enjoyability is concerned. From this point of view, a machine from the past offers fewer opportunities than the yard of a medieval castle under siege. Lastly, the public is often biased by a deeply rooted tradition of computer representation that, while effective, is not correct.

HMR also rebuilt hardware replicas of some parts of the MR. For instance, the replica of the six-bit binary adder (one third of the MR real adder) was, according to the retrieved documentation, the very first component to be built and successfully tested in early 1956. The replica was made from the original blueprints using components and tools of the period—a few of them recovered from leftover spare parts. Additionally, a modern version of the six-bit adder is available: it is made by small, handy parts that students can easily play with. The logic and the modular architecture are the same as the original adder, but today the electronic implementation uses components to reduce the size and to work with low and safe voltage. These hardware replicas are used in teaching workshops about binary arithmetic, Boolean logic, and basic architecture of digital circuits.

As we already mentioned, we rebuilt the virtual replicas of the two versions of the MR. In the following, we focus on the MR57 simulator and on how it is used in the teaching workshops that the Museum of Computing Machinery offers to middle and high schools. The MR57 simulator reenacts the final MR, which was more relevant for historical and technological issues. Moreover, it emulates the MR at a higher level of detail, making it possible to popularize different issues in computer science. As well, as discussed in the Introduction, such a simulation is effective in higher education, since old machines can be used to better show and discuss principles and mechanisms of computer science.

5.1 The Manual Control Panel of the MR57

As expected, the user interface of the MR57 simulator reproduces that of the MR57. Figure 3 shows the only surviving photo of the MR57 with the *Quadro di Controllo Manuale* (QCM, or Manual Control Panel)—that is, the user interface of the machine. Figure 4 presents the virtual QCM as reproduced by the simulator. Among the QCM components are the following (with their original names):

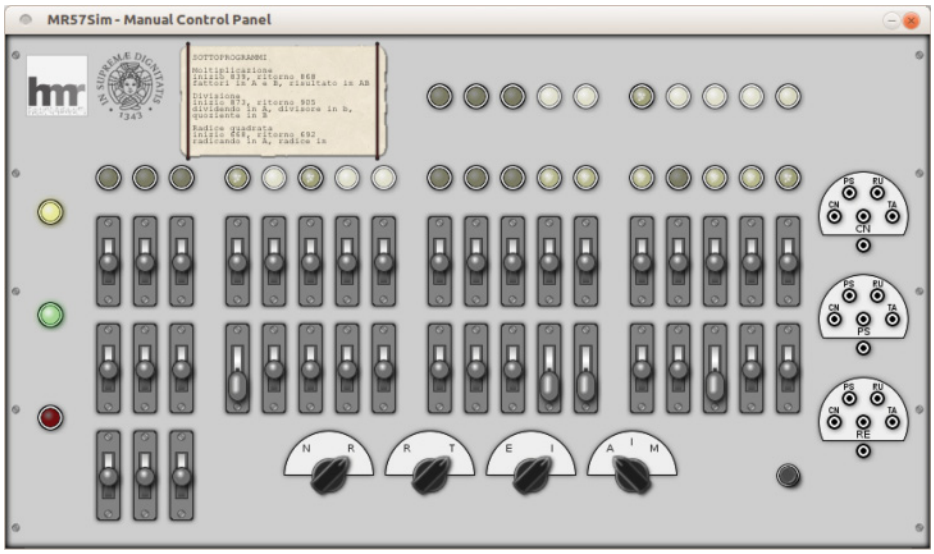

Fig. 4. The Manual Control Panel as reproduced by the simulator of the MR.

—*Indicatore del Numeratore* (IN, counter display, 10 lights, top right): displays the current binary value of the program counter;

—*Indicatore della Memoria* (IM, memory display, 18 lights, under IN): displays the binary value of the last written word on the MR memory; however, since the ferrite core memories have to be rewritten after each reading, IM shows also the last read word;

—*Tastiera della Memoria* (TM, memory keyboard, 18 vertical switches, under IM): used to bit-wise set a word to be written in memory;

—*Tastiera delle Istruzioni* (TI, instruction keyboard, 18 vertical switches, under TM): used to bit-wise set an instruction (e.g., in Figure 4, an unconditional jump instruction to the memory address 100 is set on TI, bits 0 through 9 are the address, and bits 10 through 14 are the instruction code);

—*Chiavi di Arresto Condizionato* (CAC1/2/3, keys for conditional stop, 3 vertical switches, bottom left): used to allow the program to stop at breakpoints defined in the program;

—*Commutatori dei Modi di Funzionamento* (switches for operating modes, 4 switches with two or three positions, bottom centre, used in combination to set the MR in different operating modes), from left to right: CNR (address switch), the memory address of the instruction operand is taken from the N program counter or from the lower 10 bits of the R register; CRT (memory access switch) connects the memory input to RE or to TM; CEI (instruction switch) connects the input of the instruction register to TI (External instructions) or to memory (Internal instructions); CAIM (stepping options switch) sets the MR to Automatic continuous execution (until program ends), or to manual step by Instructions or Microinstructions;

—*Pulsante di Avviamento* (PA, start button, bottom right): used to enable the clock pulse generator and thus to start the working cycle of the MR.

The QCM of the MR57 presents the typical characteristics of a machine from the 1950s: bit-wise interaction, ability to execute machine instructions directly, information about the state of the machine given through relevant registers, and every other interaction delegated to batch I/O of programs and data through devices such as punched tape readers and teletypewriters.

Facing such an interface, so different from today standards, people may feel how difficult it was to work with the early computers. Unfortunately, this is the sort of immersion that makes them run away. Popularizing computer science is a demanding task and requires some sort of tutoring. In our experience, it is mandatory the presence of an entertainer who guides visitors in a walk-through, presenting them with tasks of increasing difficulty.

5.2 Unveiling Mechanisms of Computer Technology

The very spartan MR57 user interface can be exploited to show the inner workings of a computer. The basic mechanisms implemented in the MR57 are still present in modern computers; they are just hidden under many layers of abstraction and user-friendliness.

A typical demonstration session with the MR57 simulator starts with the execution of single machine instructions directly set on TI using 5 bits (10 through 14, counted starting from the right) for the instruction code and 10 bits (0 through 9) for the operand, which usually is a memory address. For instance, here are the detailed steps needed to use the MR as a counting machine:

—Set the first instruction on TI, the instruction code must be 0 (00000 in binary), corresponding to the *QM* instruction that transfers the TM current value to a cell of memory, and the operand must be the address of the cell used to store the value of the counter, say 0 (0000000000) for simplicity.

—Set the step value for the counter on TM; any number greater than zero will be fine.

—Check that CNR is set to the default position R (*registro*, register).

—Check that CRT is set to the default position T (*temporizzatore*, pulse generator).

—Set CEI to E (*esterno*, external) so that machine instructions are read from TI.

—Set CAIM to I (*per istruzioni*, by instructions) to execute a single instruction and stop.

—Check that the green light is on—that is, the MR57 is ready to start.

—Press PA; the MR57, executing the instruction on TI, loads the content of TM in memory and stops.

—IM now displays the value set on TM and currently stored in memory.

—Set another instruction on TI; the code must be 27 (11011), corresponding to the $n + A$ instruction, which loads the value of a memory cell in the register A; the operand must be left untouched so that it still refers the counter address.

—Check the green light and press PA again; the MR copies the value from memory to the A register, which now contains the step value of our counting machine; IM does not change as a memory read operation was performed, but the binary value is the same of the previous write operation.

—Set the last instruction on TI; the code must be 7 (00111), corresponding to the A+M instruction, which adds the value of A to a memory cell; again, the operand must be left untouched.

—Check the green light and press PA; the MR does a counting step, and IM displays the current value of the counter.

—Keep pressing PA to continue counting.

The description of the preceding process is verbose, but in practice it does not require much time and explains, through a practical example, many issues of computer internal architecture and mechanisms, from the binary representation of both instructions and data to the role of memory and registers to perform operations and store results.

The MR57 simulator is also accurate in the real-time reproduction of the machine performance. Continuing the previous counter example, it is possible to perform a benchmark to assess the number of additions that the MR57 executed per second, which is a common measure to express the speed of a computer at the time. Provided that the counter step was set to 1:

—Set CAIM to A (*automatico*, automatic) to continuously execute instructions.
—Leave CEI to E to always execute the instruction set on TI.
—Press PA and then watch the MR endlessly count and display the counter current value on IM.

It is enjoyable to watch the MR counting and displaying the counter value on IM, which progressively increases, overflows, and then starts again. Although the lower bits change too fast, the cycle is observable on the three to four most significant ones, and it is possible to invite the public to manually mark the time between overflows. Considering that a full cycle requires 2^{18} additions and using the average of few samples to mitigate the error of manual timing, it is easy to guide people to record a performance of about 70,000 additions per second—a good runner for the time.

A further step in the MR57 exploration is to run simple applications. Thanks to being able to directly read a memory image from a punched tape, loading a program was quite an easy task on the MR57, mostly a matter of careful handling of the paper tapes. However, once a program was read into the memory, it was not obvious how to actually start it. The program is a sequence of instructions, and the computer has to begin the execution of the program from its starting instruction—that is, it must jump to the first instruction of the program. Today, it is all hidden behind a mouse click and the operating system takes care of the details. On the MR57, the process is completely transparent, and the user is responsible for doing all steps correctly. The user must know the address of the first instruction of the program, and the loading procedure must be accordingly carried out:

—Set a jump on TI; the instruction code must be 16 (10000), corresponding to the *Z* instruction, which is an unconditional jump, and the operand must be the address of the first instruction of the program—say, for example, 100 (0001100100).
—Check that CNR and CRT are set to their defaults, R and T.
—Set CEI to E and CAIM to I.
—Check the green light and press PA; the MR57 executes the jump, and IN displays the new value of the program counter, providing feedback that it now points to the first instruction of the program.
—Set CEI to I (*interno*, internal) so that the MR57 reads the instruction from its memory.
—Set CAIM to A.
—Check the green light and press PA; the program starts.

The procedures that we describe may seem a bit hard to be proposed to the general public. Depending on the attending people, the simulator has to be presented at different levels. At the simpler level, it helps to give a practical demonstration of a computer of the 1950s with all of its oddities. However, when we consider teaching, the simulator has proven to be a valuable tool to make the lessons at the museum more enjoyable. In particular, for middle- and high-school students (our primary target and all “native gamers”) the simulator is able, after a while, to induce that sort of good addiction that pushes many users to try more challenging tasks. As a matter of fact, we are proud that on more than one occasion, young visitors have asked us to “step up” and proceed to the MR programming workshop.

5.3 Introducing Basic Concepts of Computer Science

There are good reasons for using the MR as an example in the teaching workshops: it is a piece of history (the first Italian computer, see Section 4.1), it is an interesting machine (see Section 4.2), and it is simple enough that it can be presented to a general audience. We could have chosen the second machine built in Pisa: it was not dismantled and is now on display at the museum. However, despite

being built only 4 years after the MR, it is a much more complicated machine: it requires a lot of time to be barely understood even for a computer scientist or an engineer. For example, it implemented innovative (at the time) solutions for subroutine management that are very different from the classical solutions that we are used to.

During a typical workshop session on the MR57, several programs are loaded and run. They are written in a machine language that is simple yet elegant, having only 32 instructions with the same one-word format—in modern terms, the MR57 is a pure reduced instruction set computing (RISC) machine. Presenting the MR57 programs and discussing their implementation, having a look at the code, is surprisingly possible even with people who have a basic computer science background. Some examples derive from original code or adopt a similar programming style. As part of the immersion in past software development techniques, examples are given of dirty tricks typical of programming in great memory shortage.

During its short life, the MR57 was intensively used to provide computation services to research projects ranging from physics to chemistry. However, these kinds of problems are too complex to be presented to the public. To keep the audience's attention, we had to make a compromise between appealing software examples and pertinence with respect to the actual use of the MR57.

One of these examples is a game that turns the MR57 in a classical three-reel, eight-symbol slot machine. The spinning reels are represented by the three middle bits of each of the three groups of five bits in which IM is divided. The program starts with the usual jump procedure, then the user can spin the reels by pressing PA and can stop them by lowering the first CAC bit, thus activating the breakpoint with code 001. As usual, the jackpot is represented by a triple 7 (in binary a triple 111). Although it may seems disrespectful of the much more relevant problems the MR57 was used for, this simple game easy introduces the user interface of the MR57 and is a good starting point to present some programming challenges of the time, and even to discuss deeper issues about foundations of computer science.

User interaction. The slot machine game shows the use of the QCM as an I/O device. Playing the game, it seems as though the MR57 is able to manage interactive I/O, but actually it is not. The MR57, like most computers of its time, was a pure batch machine. Activating the breakpoint actually stops the machine by inhibiting the generation of clock pulses, which are restarted when PA is pressed. In the meantime, the machine is deadly stopped. Although effective for the sake of the game, the interaction is not controlled by the program. The MR57 was able to perform active polling, but this was limited to input through the TM keyboard on the QCM. First improvements in I/O management appeared in those years in the most advanced projects, such as the MIT TX-2 computer that featured an early form of synchronization by interrupts [Smotherman 1989].

Software hot topics of the 1950s. Random numbers generation has been a research topic since the very beginning of the modern computing era [Von Neumann 1951]. The MR57 was actually used to experiment with software random number generation, mostly for application in Monte Carlo integration methods. The slot machine game uses a pseudorandom number generator to simulate the spinning reels. The game source code is exploited both to introduce the topic and to show several solutions to implement the generator subroutine.

Universal Turing machine. Among the different implementations of the generator subroutine, there are few that rely on "recent" results. Indeed, the underlying math that makes the difference between good and bad generators was not deeply investigated until the mid 1960s, and the debate lasted several years [Park and Miller 1988]. In practice, in these cases, the MR57 is running a program that belong to its future. This little paradox is used to point out the different timelines that characterize hardware and software evolution, as well as to meditate on the universality of computers and to recall fundamental concepts of computation such as Turing equivalence.

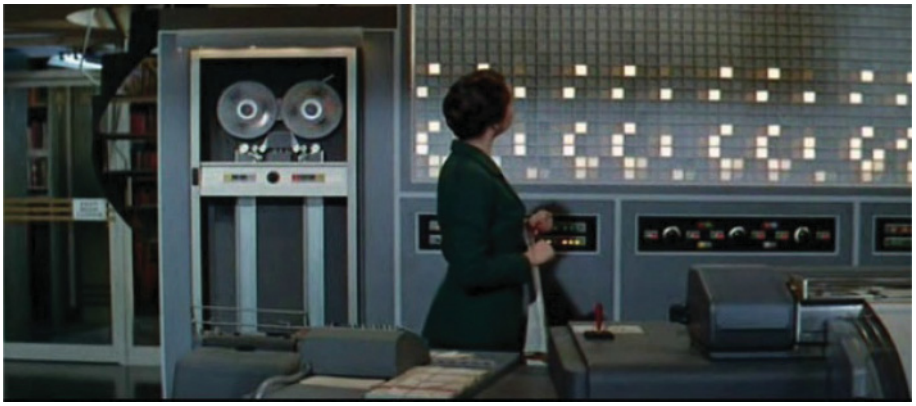

Fig. 5. A frame from *Desk Set* [Lang 1957].

5.4 Debunking Computer Representation in Pop Culture

For many years, the typical movie representation of a computer was based on blinking lights and spinning tape reels. In recent times, tape reels have been substituted with bundles of network cables and lot of monitors continuously scrolling text or plotting graphs.

In the mind of scriptwriters and set designers, these representations are aimed at depicting, for a less experienced audience, complex machines that are working on some important task, thus the need for moving parts (tape reels) or changing shapes (lights and monitors). Moreover, the task performed must appear complicated yet intelligible to a well-trained user (usually a character in the drama), thus the use of regular patterns of lights as well as graphs and text on monitors.

In Figure 5, a frame from the film *Desk Set* is shown—a 1957 classic Hollywood light comedy starring Katharine Hepburn, Spencer Tracy, and the *EMMARAC* computer, which was fictional yet largely inspired by machines of the period (IBM appears in the credits). The lady that operates Emma looks at the light patterns like she is able to read them. This kind of representation is very common and lasted even in masterpiece movies produced later and trying to imagine the computers in the future. One paradigmatic example is *Alien* [Scott 1979], where the *Mother* computer is represented as a room full of blinking lights. All of these representations are based on a realistic starting point: early computers do have lights and spinning tape reels. However, as absurd as it may seem, a genuine computer of the 1950s would not have passed the audition for *Desk Set*.

The computer lights were not supposed to be read in real time. Although it was still possible to grasp some information from the lights when the computer was running, their intended use was to display register and memory status when the machine stopped.

The MR57 simulator was developed to accurately represent the behaviour of the special triodes used for the QCM lights, and it can be used as a good example of the look and feel of a computer of the 1950s. The QCM triodes had times of ionization and deionization of, respectively, 50 and 200 μs: for their purpose, they were very fast and reactive lamps. However, having a clock cycle of 4 or 8 μs (depending on the microinstruction), the MR57 was faster than its lights. The bit values normally changed too fast for the triode to reach full ionization. When it happened (the bit stays on 1 for a while), the light remained on due to the longer deionization time even if the bit was changed to 0. Thus, the QCM lights were emitting an incomprehensible and flickering gleam: they were fully on or off only when the machine stopped and the bit values remained constant.

Besides being a curious anecdote and a good opportunity to speak about triodes, clock cycles, and computer speed, the use of an accurate simulation helps to discuss the differences between what is considered realistic because it is persuasive for common sense and what is actually true.

6. CONCLUSIONS AND FUTURE WORK

We witnessed two direct consequences of the MR rebuilding project so far. First of all, we can further confirm that the experience of programming or just looking at (the simulation of) an old computer can be a fruitful tool for teaching and popularizing computer science. Now improved with a graphical interface, the MR simulators are exploited to show the look and feel of a computer of the 1950s, explaining several fundamental concepts and mechanisms that still are at the base of the inner workings of computers. The schools participating in the teaching workshops held at the museum were provided with an immersive and fascinating experience in a past technology, which entices the kids to learn more and deepen their understanding of the topics at hand in each workshop.

Modelling techniques and simulation tools were pivotal to support the research phase and to validate our rebuilding hypotheses. Our experience also confirms that to rigorously approach the history of technological artefacts, sound technology expertise is needed. The MR remained underestimated for many years because the previous research did not attempt to recover and study the technical documents of the CEP project. On the contrary, the precise understanding of the technology of the time that was needed to rebuild a virtual replica of the MR led to full acknowledgement of its prominent role in the history of Italian computer science.

Current activities of the HMR project are aimed at extending the collection of programs used to show the features of the MR and the kind of applications for which it was used. Furthermore, we are working to improve the simulators of the MR I/O devices. Teletypewriters, tape readers, and tape punchers used by the MR were borrowed from telegraphy. We plan to build independent simulators of such devices to be connected to the virtual MR (thus acting as computer peripherals), as well as to be used to set up simple networks to show how worldwide communications worked prior to the Internet.

ACKNOWLEDGMENTS

The desk research for the rebuilding of the MR was carried out in collaboration with the Archives of the University of Pisa and the Library of ISTI-CNR of Pisa. A special thanks goes to Daniele Ronco.

The first simulator of the MR56 was developed with contributions from Claudio Imbrenda. Diego Ceccarelli helped in testing the simulator and in the restoration of the MR56 system software.

Finally, a special thanks goes to Elio Fabri, one of the protagonists of the MR who, with his personal memories and the documents he preserved, provided us with an invaluable contribution to the understanding of the technology of the very first Italian computer.

REFERENCES

Aconit. 2013. Association pour un Conservatoire de l'Informatique et de la Télématique. Retrieved March 1, 2014, from http://www.aconit.org.

A. Akera and W. Aspray (Eds.). 2004. *Using History to Teach Computer Science and Related Disciplines*. Computing Research Association, Washington, DC.

G. Berry, J. Sheard, and M. Quartly. 2011. A virtual museum of computing history: An educational resource bringing the relationship between people and computers to life. In *Proceedings of the 13th Australasian Computing Education Conference (ACE'11), Vol. 114*. Australian Computer Society, Darlinghurst, Australia, 79–86.

Bletchley Park. 2013. Bletchley Park Home Page. Retrieved March 1, 2014, from http://www.bletchleypark.org.uk/.

T. Blyth. 2013. Narratives in the history of computing: Constructing the information age gallery at the science museum. In *Making the History of Computing Relevant*, A. Tatnall, T. Blyth, and R. Johnson (Eds.). IFIP Advances in Information and Communication Technology, Vol. 416. Springer, 25–34.

C. P. Burton. 2005. Replicating the Manchester Baby: Motives, methods, and messages from the past. *IEEE Annals of the History of Computing* 27, 3, 44–60.

S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. 2007. USARSim: A robot simulator for research and education. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*. IEEE, Los Alamitos, CA, 1400–1405.

CCS. 2013. CCS Home Page. Retrieved March 1, 2014, from http://www.computerconservationsociety.org/.

CHM. 2013. Computer History Museum Home Page. Retrieved March 1, 2014, from http://www.computerhistory.org/.

G. A. Cignoni and F. Gadducci. 2012. Rediscovering the very first Italian digital computer. In *Proceedings of the IEEE 3rd History of Electro-Technology Conference*. IEEE, Los Alamitos, CA, 1–6.

G. A. Cignoni and S. Paci. 2012. UML modelling and code generation for agent-based, discrete events simulation. In *Proceedings of the International Workshop on Applied Modeling and Simulation*. 50–59.

G. A. Cignoni, D. Ceccarelli, and C. Imbrenda. 2009. Il restauro del software di sistema della Macchina Ridotta del 1956. In *Atti della Conferenza AICA 2009*.

J. Copeland. 2004. Colossus: Its origins and originators. *IEEE Annals of the History of Computing* 26, 4, 38–45.

CyArk. 2013. Cyark Home Page. Retrieved March 1, 2014, from http://archive.cyark.org/.

G. De Marco, G. Mainetto, S. Pisani, and P. Savino. 1999. The early computers of Italy. *IEEE Annals of the History of Computing* 21, 4, 28–36.

Debian. 2013. Debian Home Page. Retrieved March 1, 2014, from http://www.debian.org/.

Deutsches Museum. 2013. Deutsches Museum Home Page. Retrieved March 1, 2014, from http://www.deutsches-museum.de.

EDSAC. 2013. The Edsac Simulator. Retrieved March 1, 2014, from http://www.dcs.warwick.ac.uk/~edsac/.

G. A. Fierheller. 2006. *Do Not Fold, Spindle or Mutilate: The "Hole" Story of Punched Cards.* Stewart Publishing and Printing, Markham, ON, Canada.

T. H. Flowers. 1983. The design of Colossus. *IEEE Annals of the History of Computing* 5, 3, 239–252.

S. Greenblatt. 1991. Resonance and wonder. In *Exhibition Cultures: The Poetics and Politics of Museum Display*, I. Karp and S. Lavine (Eds.). Smithsonian Institution, New York, NY, 42–56.

P. V. Hansen. 1992. Experimental reconstruction of the medieval trebuchet. *Acta Archaeologica* 63, 189–208.

D. Harms and D. Berque. 2001. Using a PDP-11/10 to teach content and history in computer organization courses. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education*. ACM, New York, NY, 209–213.

HMR. 2013. HMR: Hackerando la Macchina Ridotta. Retrieved March 1, 2014, from http://hmr.di.unipi.it/.

IEEE1800. 2009. 1800–2009: IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language. *IEEE Standards Association*. Available at http://standards.ieee.org/findstds/standard/1800-2009.html.

IEEE610. 1990. *610.12–1990—IEEE Standard Glossary of Software Engineering Terminology*. DOI:10.1109/IEEESTD.1990.101064

J. Impagliazzo and M. Samaka. 2013. Bringing relevance to computing courses through history. In *Making the History of Computing Relevant*, A. Tatnall, T. Blyth, and R. Johnson (Eds.). IFIP Advances in Information and Communication Technology, Vol. 416. Springer, 135–143.

KZM. 2013. Konrad Zuse Museum Hünfeld mit Stadt- und Kreisgeschichte. Retrieved March 1, 2014, from http://www.zuse-museum-huenfeld.de/.

W. Lang (Director). 1957. *Desk Set* (movie). Twentieth Century Fox.

R. G. Laycock, D. Drinkwater, and A. M. Day. 2008. Exploring cultural heritage sites through space and time. *Journal of Computing and Cultural Heritage* 1, 2, 1–15.

MAME. 2013. Multiple Arcade Machine Emulator—MAMEDev Wiki. Retrieved March 1, 2014, from http://mamedev.org/devwiki.

M. Marcus and A. Akera. 1996. Exploring the architecture of an early machine: The historical relevance of the ENIAC machine architecture. *IEEE Annals of the History of Computing* 18, 1, 17–24.

MHCR. 2013. History of Computing Conference Home Page. Retrieved March 1, 2014, from http://historyofcomputingjune2013.wordpress.com/.

MOSI. 2013. Museum of Science and Industry | MOSI Home Page. Retrieved March 1, 2014, from http://www.mosi.org.uk/.

MR/S/2. 1957. Macchina Ridotta, Dis. 1. Retrieved March 1, 2014, from http://hmr.di.unipi.it/Disegni/ParteII/195703_MR_S_2.pdf.

B. Napper. 2013. Simulators: The Manchester Mark I Prototype. Retrieved March 1, 2014, from http://www.computer50.org/mark1/simulators.html.

NASA Office of Logic Design. 2013. Apollo Guidance Computer (ACG) Schematics. Retrieved March 1, 2014, from http://klabs.org/history/ech/agc_schematics.

S. K. Park and K. W. Miller. 1988. Random number generators: Good ones are hard to find. *Communications of the ACM* 31, 10, 1192–1201

G. Parolini. 2008. Olivetti Elea 9003: Between scientific research and computer business. In *History of Computing and Education 3 (HCE3)*. IFIP Advances in Information and Communication Technology, Vol. 269. Springer, 37–53.

Plan28. 2013. Plan 28: Building Charles Babbage's Analytical Engine. Retrieved March 1, 2014, from http://plan28.org/.

M. Reibold, P. Paufler, A. A. Levin, W. Kochmann, N. Pätzke, and D. C. Meyer. 2006. Materials: Carbon nanotubes in an ancient Damascus sabre. *Nature* 444, 286.

R. Rojas. 1997. Konrad Zuse's legacy: The architecture of Z1 and Z3. *IEEE Annals of the History of Computing* 19, 2, 5–16.

R. Rojas. 1998. How to make Zuse's Z3 a universal computer. *IEEE Annals of the History of Computing* 20, 3, 51–54.

R. Rojas and U. Hashagen (Eds.). 2000. *The First Computers—History and Architectures*. MIT Press, Cambridge, MA.

R. Rojas, F. Darius, C. Goktekin, and G. Heyne. 2005. The reconstruction of Konrad Zuse's Z3. *IEEE Annals of the History of Computing* 27, 3, 23–32.

Royal Society. 1985. *The Public Understanding of Science*. Royal Society of London, London, UK.

A. E. Sale. 1998. *Colossus 1943–1996*. M & M Baldwin. Cleobury Mortimer, Shropshire, UK.

R. Scott (Director). 1979. *Alien* (movie). Twentieth Century Fox.

SM. 2013. Science Museum Home Page. Retrieved March 1, 2014, from http://www.sciencemuseum.org.uk/.

M. Smotherman. 1989. A sequencing-based taxonomy of I/O systems and review of historical machines. *ACM SIGARCH Computer Architecture News* 17, 5, 5–15.

K. Squire, M. Barnett, J. M. Grant, and T. Higginbotham. 2004. Electromagnetism supercharged! Learning physics with digital simulation games. In *Proceedings of the 6th International Conference on Learning Sciences*. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 513–520.

D. D. Swade. 2005. The construction of Charles Babbage's difference engine no. 2. *IEEE Annals of the History of Computing* 27, 3, 70–78.

SysML. 2013. OMG Systems Modeling Language. Retrieved March 1, 2014, from http://www.omgsysml.org/.

A. Tatnall (Ed.). 2012. *Reflections on the History of Computing: Preserving Memories and Sharing Stories*. IFIP Advances in Information and Communication, Vol. 397, Springer.

TNMOC. 2013. The National Museum of Computing | Home to the First Electronic Computers Home Page. Retrieved March 1, 2014, from http://www.tnmoc.org/.

Ubuntu. 2013. Ubuntu Home Page. Retrieved March 1, 2014, from http://www.ubuntu.com/.

UML. 2013. Unified Modeling Language. Retrieved March 1, 2014, from http://www.uml.org/.

University of Cambridge. 2013. ECAD and Architecture Practical Classes: ECAD Labs. Retrieved March 1, 2014, from http://www.cl.cam.ac.uk/teaching/1011/ECAD±Arch/.

J. Von Neumann. 1951. Various techniques used in connection with random digits. *Applied Math Series* 12, 1, 36–38.

M. V. Wilkes. 1986. The genesis of microprogramming. *IEEE Annals of the History of Computing* 8, 2, 116–126.

M. V. Wilkes and J. B. Stringer. 1953. Micro-programming and the design of the control circuits in an electronic digital computer. *Mathematical Proceedings of the Cambridge Philosophical Society* 49, 2, 230–238.

WxWidgets. 2013. WxWidget Home page. Retrieved March 1, 2014, from http://www.wxwidgets.org.

T. Zoppke and R. Rojas. 2006. The virtual life of ENIAC: Simulating the operation of the first electronic computer. *IEEE Annals of the History of Computing* 28, 2, 18–25.

G. Zotti. 2008. Tangible heritage: Production of astrolabes on a laser engraver. *Computer Graphics Forum* 27, 8, 2169–2177.

H. Zuse. 2013. Reconstruction of Konrad Zuse's Z3. In *Making the History of Computing Relevant*, A. Tatnall, T. Blyth, and R. Johnson (Eds.). IFIP Advances in Information and Communication Technology, Vol. 416. Springer, 287–296.

Received July 2013; revised March 2014; accepted March 2014