Virtualization-based architecture for the operating systems history demonstration

Anatoliy V. Gusev, Vladimir Yu. Kovalenko, Dmitriy A. Kostiuk[1]

***Abstract*—**The universal virtualized infrastructure for embeddable demonstration of mobile and desktop operating systems is presented, with emphasis on live display of the graphical user interface history. Experience of using virtual machines instead of screenshots in a visual timeline of GUI is reviewed. Presented solution provides user control via the Internet or intranet web access, and QEMU-based nested virtualization for the unified interaction with heterogeneous simulated architecture of mobile devices. Availability of materials is considered, as far as approaches used to improve web-based user interaction with virtualized GUI. Internal architecture, deployment and scaling principles are discussed.

***Keywords*—**virtualization, history, computer interfaces, live demonstration.

I. INTRODUCTION

Although technically working with the new information technologies doesn't demand knowing the history of their development, however specialist who formulates or applies modern theory without knowledge of its history, runs the risk of repeating the mistakes of predecessors personally one by one [1].

Being in the context of spoken above statement we have developed the ostimeline project [2], intended to provide the user (either via network or on a stand-alone workstation) with a set of chronologically-related HTML documents, each one with a description of the specific graphical operating system and its live illustration in the form of built-in frame with the screen of a running virtual machine (thanks to the performance of today's notebooks and desktop PCs, this task is easy enough), as shown on Fig. 1.

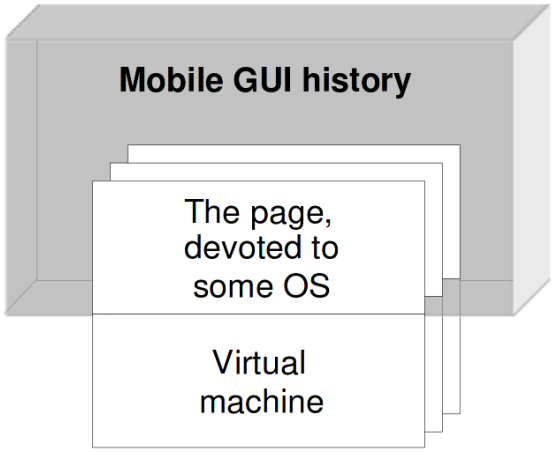

Fig. 1 Scheme of the virtualization-enchanced interactive document

Nowadays there is some amount of web based projects demonstrating historical operating systems with their 'screen' embedded into the Internet page. One of such projects is oldweb [3, 4] (dedicated to vintage web browsers), and there are several others (e.g. single demos of some old Microsoft or Apple OS, and Amiga). However we still didn't see web demonstrations dedicated to vintage mobile operating systems (perhaps due to some technical difficulties, discussed later). This fact increases the relevance of presented here project, based on the original university

1 Anatoliy V. Gusev, Brest State Technical University, Brest, Belarus (e-mail: avg.tolik@gmail.com)
Vladimir Yu. Kovalenko, Brest State Technical University, Brest, Belarus (e-mail: vl.kovalenko1989@ya.ru)
Dmitriy A. Kostiuk, Brest State Technical University, Brest, Belarus (e-mail: dmitriykostiuk@gmail.com)

Authors are grateful to CERES project for the financial support

course of the graphical user interface history, which itself includes 30 mobile operating systems and 40 desktop ones, user interfaces of which turned out to be a historical milestone.

II. TECHNICAL INFRASTRUCTURE OF THE HTML-BASED VIRTUAL MACHINES DEMONSTRATION

Client-server applications with web interface have gained today the widest usage, and are already replacing classic desktop applications in some areas. List of reasons backing such move includes the presence of a web browser on all platforms and architectures, as well as sufficient performance of a JavaScript client-side web applications are built on (at least in modern browsers). Universal accessibility (including platforms with a touchscreen interface and other so-called 'thin clients', i. e. on mobile and portable devices, targeted mainly at the use of cloud services) makes browser the most convenient entry point for the end user application / service. This is especially significant when the main computational load is placed on the server and / or other nodes in the network, and the client is used only for management and access to resources (the reverse case, unfortunately, at this moment is not effective as far as JavaScript implementation involves larger overhead costs compared to traditional languages on server platforms).

One of the tasks that can get significant benefit in case of the client-server implementation based on a private cloud, is virtualized applications exhibition, either in demonstration purposes or for some usability comparison & research, etc.

Problem of demonstrating software in case of strong fragmentation in target hardware platforms of correspondent devices is rather obvious. Therefore developers of such exhibitions have to use set of emulators with disk images containing a variety of mobile operating systems to run. In this case, even if the regular tools of desktop virtualization would provide a convenient interface to simultaneously run multiple instances of different emulators, low productivity of the emulation associated with different processor architectures would cause negative impact on the productivity and efficiency (especially noticeable during boot process of the emulated hardware). Also it turns out sometimes that local emulation do not fit the workflow at all: sometimes a shared access to a copy of the test application is required, as well as a demo access [5].

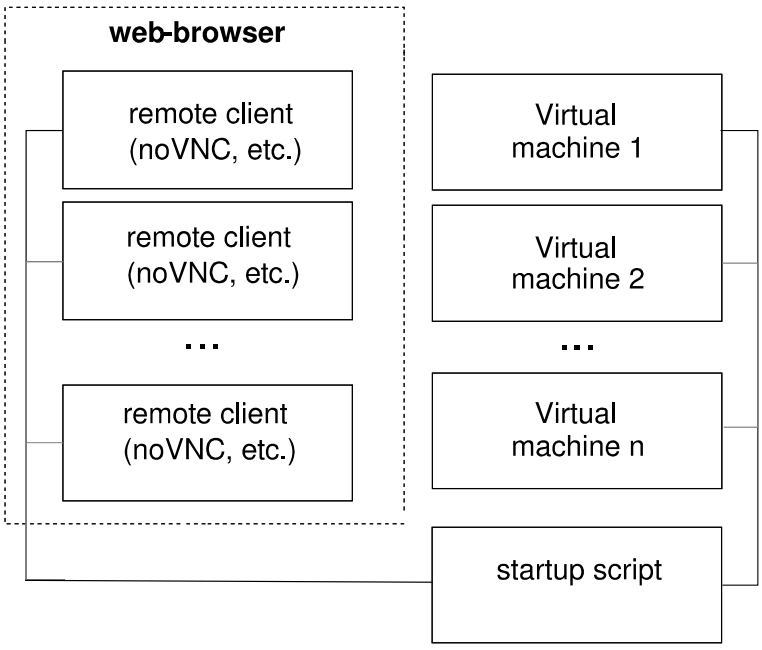

Fig. 2 General scheme of the virtual machines interaction

Technical infrastructure we have proposed to implement such demo materials, is briefly reviewed in [2] and includes the following components:

- QEMU virtual machine,
- some remote access client written in JavaScript and HTML5 (e.g. noVNC),
- JavaScript framework to display information in one interactive timeline, or sequence of slides, etc.

Running the scheme shown in Fig. 2 is done by a startup script. This script scans subdirectories in search for the information elements: pages with the content, virtual machine images and scripts to run them. Regardless of used remote access protocol (VNC or some other), script passes port numbers to discovered virtual machines and reconstructs HTML document to include pages with information materials into the demo timeline. This component-based approach allows to divide the information, making some parts of the exposition publicly available on the network, and taking away from public access those virtualized systems which can not be redistributed due to the terms of commercial licenses.

III. EMULATORS OF THE MOBILE AND DESKTOP HARDWARE

Virtual machine is used as a fully isolated container storing a snapshot of the already running OS. Choice of QEMU is caused by the extremely simple transfer of virtual machine images between computers, and also by its ability to emulate not only x86-compatible platforms, but also ones based on ARM, MIPS, and Motorola 68k processors, which are mandatory for some vintage operating systems.

However, currently multiplatform capabilities of QEMU are poorly used: support of motherboards and peripheral devices on alternative platforms at this emulator is not always sufficient, especially in case of really old operating systems.

At the same time, there are emulators available to support virtually all vintage Intel-incompatible systems – as community-developed emulators, so ones created as the part of some commercial SDK. The first variant is more common in desktop operating systems segment (they are used to emulate Xerox Alto, Apple Lisa and Mac computers, Amiga, etc.). For now ostimeline uses only one community-developed mobile OS emulator – Open Einstein project, which allows to run NewtonOS.

As opposed to them, emulators from proprietary mobile devices SDK are not typical for desktop segment but are dominating in mobile one, such as Psion EPOC16 and EPOC32, PalmOS, Magic Cap, Windows CE operating systems, and pre-release versions of Android SDK.

Unfortunately, all these specialized emulators don't support snapshots and remote access. Therefore a large part of demos is build along with the the nested virtualization scheme (see Fig. 3), where QEMU plays the role of an external container.

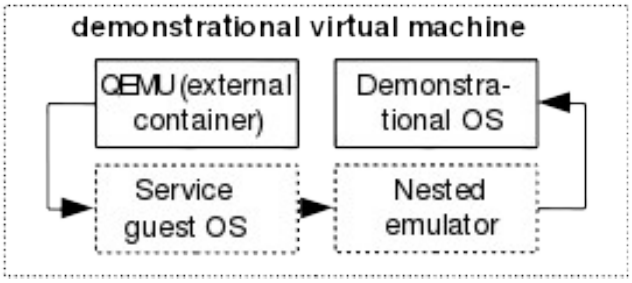

Fig. 3. Nested virtualization of the mobile OS

Of course substantial part of operating systems do not require nested virtualization and so internal emulator is not used for them. That's the case of different versions of Windows, OS/2, GNU/Linux Maemo, Android, WebOS (not least due to the fact that QEMU is often included in mobile SDK) and some others.

There is one more important component on Fig. 3, a service guest OS, which is used to start the nested emulator. Its choice is determined by the following three requirements: minimal memory consumption, usage of idle CPU cycles, and, preferable, support for USB bus emulation, which allows to emulate pointing devices in absolute coordinates (their usage will be discussed later). We have used FreeDOS and ReactOS as a service guest OS in addition to multiple versions of GNU/Linux. It should be further noted that ReactOS (a small Windows-compatible OS with an open source license) perfectly meets all three requirements, and thus in our own experience it is the first case of its successful practical usage.

IV. Availability and the legal aspects of running historical OS

Noticeable part of historically-significant operating systems are now a free/libre software (ones having such licenses from the very beginning, 'opensourced' because of the extreme aging or being a free clone of the abandoned commercial system). That's about different DOS- and Linux-based environments for desktop and mobile computers (such as GEM, CDE, Qtopia, Gnome Palmtop Environment, OpenMoko, Maemo, Android, etc.). So this part of the timeline is freely distributable (e.g. for educational and/or research purposes).

The timeline material is still at the stage of filling, and at this point the review has some missing objects, which have played an important role in the history of interfaces. But the number of historically important GUI shells without runnable versions appears to be surprisingly small: e.g., this row currently includes only two mobile systems: PenPoint OS and IBM Simon.

V. Multi-user infrastructure of the virtual machines management

Fig. 2 shows simplified architecture capable to provide user with the demonstration containing number of liveGUI demos. Such approach can be sufficient in case of relatively small amount of virtualized systems and local single-user access. But more complex solution is unavoidable in case of Internet or intranet resource simultaneously accessed with a number of users. General structure of such server is shown on Fig. 4.

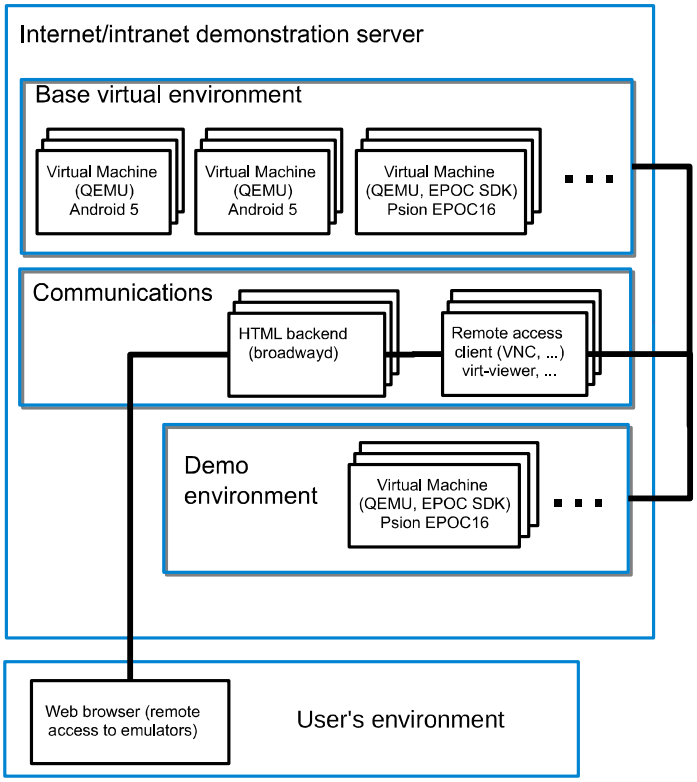

Fig. 4. The structure of the Demo server

As in case of the traditional embedded development tools stack, the demonstrated OS is on various types of emulated devices, which can be nested inside of an outer virtual machine for the convenience of the deployment and usage.

This approach significantly simplifies management of the heterogeneous set of the emulated systems, and retains all the benefits and convenience provided by server virtualization of desktop systems as far as possibilities of the automation, cloning and integration into the enterprise infrastructure, if needed.

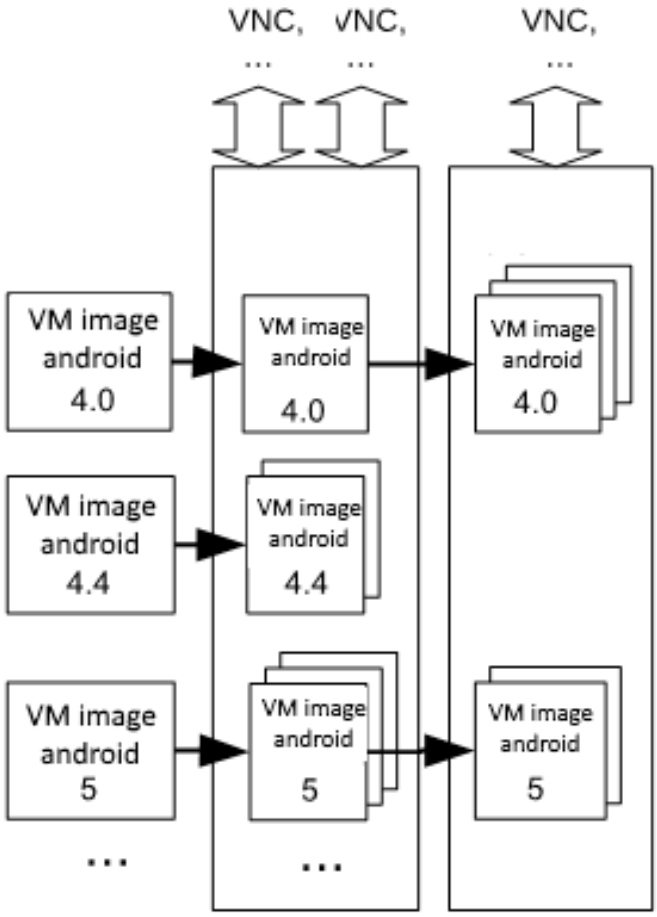

Fig. 5. Virtual machines images cloning example

Server contains reference images of a nested VMs with different OS (Android of various versions on Fig. 5). These images are configured and tuned for the fastest possible launch of a new instance. When user needs to interact with a personal exemplar of the virtualized environment, one of the reference images cloned to start the demo.

Technically image management is provided by a set of rather simple scripts that perform on-demand cloning of master images, as well as start and stop of the VM execution.

VI. More on WEB access

Users are accessing the desired VM through the web page. Initially we have used noVNC remote client based on JavaScript to access an emulator via the web environment, as described in [2]. However, in the process of testing some advantages where discovered for the access via the broadway GTK backend – a rendering subsystem in HTML 5 embedded in the latest version of the GTK library. The structure of this web-based access solution built into the server is shown on Fig. 6.

On the client-side access to the emulator is done via the web browser, which communicates with the broadway GTK backend, and the broadway draws the window of the chosen GTK3

applications on a HTML5 canvas object. The graphics program that uses GTK library runs in headless mode, i.e. only for network access. System daemons broadwayd (each on its own single network port) should be run for this, and the graphics program should be supplied with an additional set of variables at startup, which are read by the GTK library and thus determine the rendering mode. As a result, the startup code of the broadway system daemon inside of the starting script looks as follows:

```
broadwayd :1 &

GDK_BACKEND=broadway UBUNTU_MENUPROXY= LIBOVERLAY_SCROLLBAR=0
BROADWAY_DISPLAY=:1 VNCViewer
```

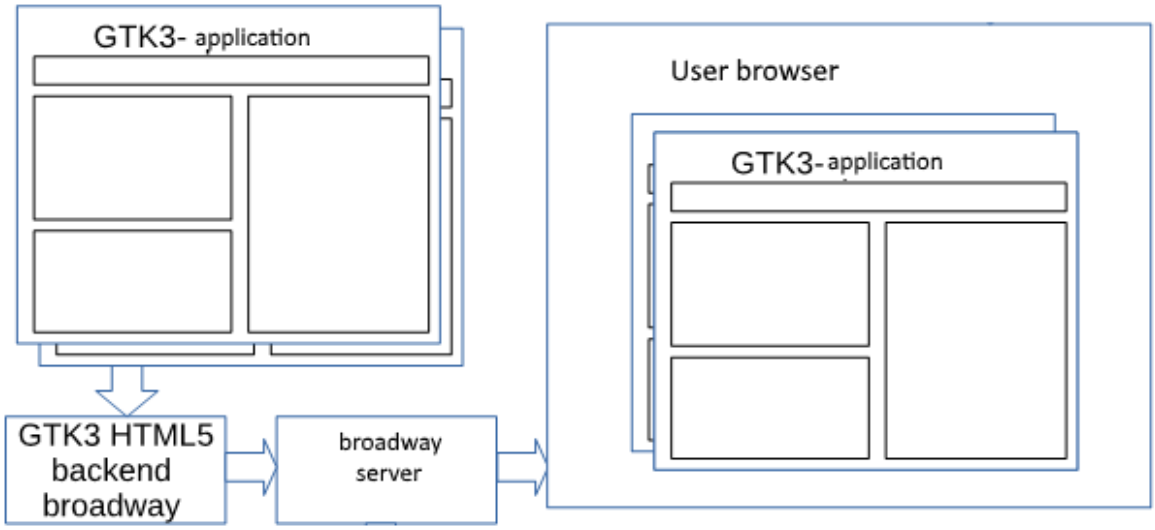

Fig. 6. The broadway operation princilpe

One of VNC or SPICE clients written using GTK3 plays the role of the graphics program running inside of the external VM and connected to the embedded VM (e.g. VNCViewer client can be used as far as QEMU graphical window); broadway technology, in its turn, translates the user session access to the browser outside of the server.

The choice of this web access architecture is influenced by two considerations: the best performance of the code (which will be discussed below), and no need for additional components of the JavaScript based VNC (broadcasting a TCP connection to the web sockets) [5]. As for additional components on the client side, which would be necessary when using the VNC-client in JavaScript, embedded in HTML page, their necessity is dictated by the fact that the code in JavaScript (executed by the web browser) does not have access to the TCP protocol used for remote access, and therefore requires the application proxy running on the client side and transmitting TCP traffic to a web sockets.

VII. noVNC AND BROADWAYD PERFORMANCE TESTING

Obviously, the reason for better performance of the selected web access solution in comparison with JavaScript VNC client is caused by a highly specialized web server that is a part of broadway, and the VNC-client running inside the external VM (both components are written in C). However, this code is executed on the server, while the use of VNC-client in JavaScript increases the processing load on the client machine. In turn, broadway also connects to the client code in JavaScript, executed by a Web browser for the reception, transmission, and rendering.

To assess the real broadway impact on the performance of the developed solution, performance testing was separately carried out for the web access (configured on the separate system with Ubuntu 15.04 GNU/Linux OS, the AMD FX-8320 processor and the RAM capacity of 16 GB). The OS choice is subject to the presence of the components required for broadway operation (GTK library version 3.8 and above, as well as some software using it). The client and server parts of the system were running simultaneously on the server, providing equal impact of the relatively high resource consumption of the selected OS GUI for both sides, so it did not affect the comparison.

The noVNC JavaScript client was used for comparison. Load was assessed by the «load average» parameter, produced with the help of htop tool, which is the most common and generally accepted indicator to assess the server workload.

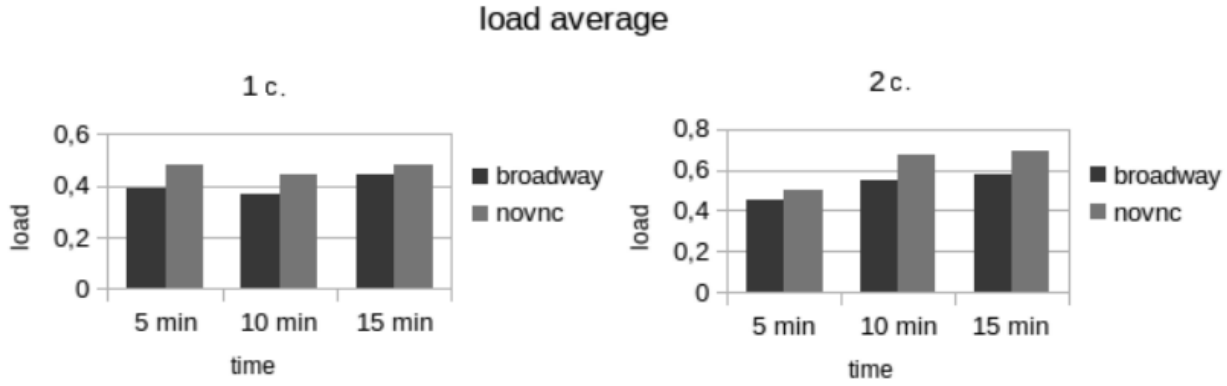

Fig. 7 noVNC and broadwayd comparison results

The comparison results are shown in Fig. 7. The analysis shows broadway advantage in terms of load, and this difference increases at growing number of connected computers (user's workplaces) to essential numbers when the number of virtual desktops reaches tens or hundreds.

Lesser server load of broadway, compared to VNC protocol, can be attributed to several factors: the lack of need for a separate VNC server, absence of operations associated with compression and stream encoding for VNC, more optimal and specialized server code. RAM consumption in the case of broadway allows to save 30-40 MB per virtual workplace.

As for the load on the client machine and the amount of generated traffic, both approaches are demonstrating parity (and both quantities are negligible). However, there broadway advantage is a complete lack of image artifacts that can occur when using VNC.

VIII. Solving the problem of non-coinciding cursors

Running vintage OS in emulator often comes with a problem of non-coinciding cursors. This problem arises from the availability of two different types of cursor positioning devices: ones with relative coordinates (computer mice) and ones with absolute coordinates (tablets and touchscreens). Relative positioning devices provide OS with a vector of the cursor movement instead of it's new coordinates, and as a result, different cursor acceleration formulas are breaking coincidence of host and guest systems cursors [6].

Mainstream virtualization systems (including QEMU) can emulate as relative pointing devices (PS/2 and serial mice), so absolute pointing ones (USB Wacom tablet in case of QEMU). Mode with absolute coordinates allows host cursor to control guest OS and is called «mouse integration mode» in desktop virtualization systems. Mouse integration is available only if guest OS has special driver from the virtual machine vendor, or supports USB tablet. In all other situations desktop virtual machines use mouse lock mode, when host cursor is hidden until some dedicated hotkey is pressed, and user can interact with the guest system only. This lock mode is not only less convenient for operation, but it is unusable for the web access to a virtualized system. So

projects providing web demonstration of old operating systems have to show two cursors moving with different speed (Fig. 8).

In our case guest systems with nested virtualization avoid this problem if ReactOS or GNU/Linux service OS works inside of QEMU (as far as these systems have driver for the USB tablet). Unfortunately, there are a lot of systems which need less convenient service OS (i.e. FreeDOS for the EPOC 16 emulator or Windows 9.x/MacOS 7.x of the emulator of Magic CAP). Also there are old mobile systems directly emulated by QEMU, such as Pen-driven version of Windows 3.x.

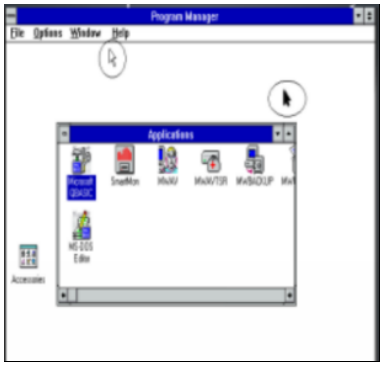

Fig. 8. Cursors non-coincidence caused by relative coordinates

Currently we have solved non-coinciding cursors problem for pre-USB service operating systems by creating a patch to QEMU, which implements emulation of the RS-232 Wacom tablet as an additional serial tablet backend. This sub-project was accepted for the Google Summer of Code program and have already landed in the mainstream QEMU version.

REFERENCES

[1] K. Thibodeau, "Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years" in The State of Digital Preservation: An International Perspective, Conference proceedings, Washington D.C., April 24-25, 2002. Washington, DC, 2002, https://www.clir.org/pubs/reports/pub107/thibodeau.html

[2] D. Kostiuk, P. Lutsiuk, S. Vlasenko, V. Zheludok, “Virtualization-based illustrated reviews of the software history.” in Proc. of LVEE 2014 conference, Brest, “Alternativa publ.”, 2014, P. 98-101.

[3] A.J. Dellinger, "Oldweb.today lets you browse the Internet like it’s 1999", The Daily Dot, Feb. 25, 2015.https://www.dailydot.com/debug/oldweb-today-legacy-browser-simulator [4] I. Kreymer, D. Espenschied, "oldweb.today: Web Archiving + Emulation", http://labs.rhizome.org/iipc2016/owt.html

[5] V.Yu. Kovalenko, D.A. Kostiuk, “Virtualized web based farm for testing and demonstration of android applications with advanced security and portability”, in Proc. of TIM-2016 conference, Grodno, 2016, http://goo.gl/4VKups

[6] Д. А. Костюк, “Особенности использования виртуализованных окружений, внедренных в презентационные материалы” in proc. of OSSEDUCONF-2012 conference, Alt Linux, Moscow, 2012, P. 83-86.

[7] А.В. Гусев, Д.А. Костюк, “Решение проблемы интеграции курсора при запуске старинных операционных систем в QEMU” in proc. of OSSDEVCONF-2016 conference, Alt Linux, Moscow, 2016, P. 28–33.