Evaluating an Emulation Environment: Automation and Significant Key Characteristics

Mark Guttenbrunner
guttenbrunner@ifs.tuwien.ac.at

University of Technology Vienna, Austria
http://www.ifs.tuwien.ac.at/dp

Secure Business Austria
http://www.sba-research.org

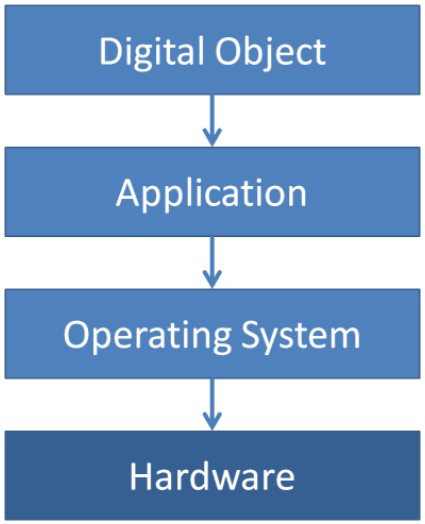
October 2012

Overview

- Introduction
- Business Process Preservation
- Validation Workflow / Comparing Renderings
- Rendering Environment: Recording Events
  - Controlling the Environment
  - Extraction of Data
  - Logging Events
- Automated Execution
- Key Characteristics of a Rendering Process
- Evaluation on an Emulator with two Case Studies
- Conclusions / Future Work

Introduction

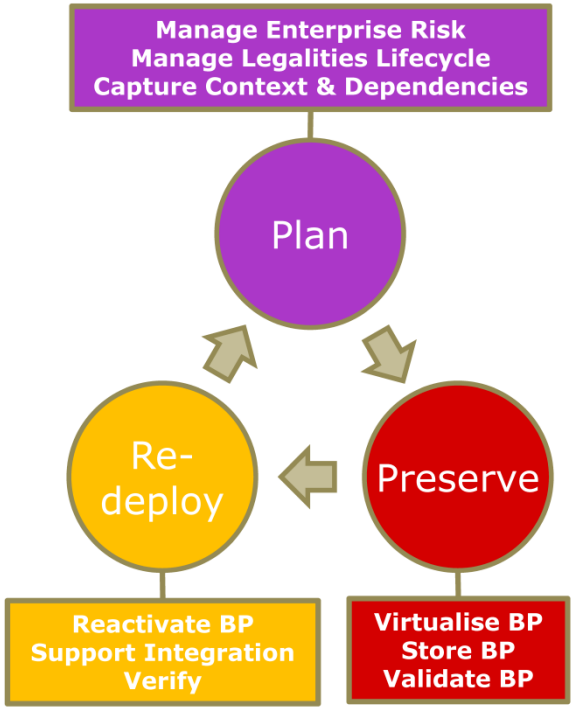
- Applying preservation action (migration/emulation) changes elements in the view-path: (object,) viewer, OS, hardware
- Strengthen trust in action by validating the rendered form of the object
- Rendering includes any form of output (screen, acoustic, physical actors, data carriers, ports etc.)
- Principle: Render an object in different environments, capture information and compare the results



Business Process Preservation

- Evaluation of Rendering in the TIMBUS process cycle for Business Processes

- Plan: Evaluation of digital preservation action results as part of preservation planning
- Preserve: Checking the validity of a business process after executing the preservation action
- Redeploy: Verifying the object when re-deployed for future execution in a new environment



Validation Workflow

1. Describe the original environment/context
2. Determine external events that influence the object's behavior (e.g., user input, web services)
3. Decide on what level to compare the digital object (memory dump, screenshot, output device)
4. Recreate the object's view path in a different environment (e.g., emulator)
5. Apply data captured from original environment to the new environment
6. Extract rendered data
7. Compare extracted data

Comparing Renderings

- Extract characteristics about the process (e.g. timing)
- Extract data rendered during the process from environment (e.g. visual, audible, actors)

- Avoid side-effects due to non-deterministic influences (e.g. manual input, external data from sensors, web services)
  - deterministic algorithms are “algorithms in which the actions of each process are uniquely determined by its local knowledge"
- Automation of process necessary

Rendering Environment

- Philips G7400
  - Basic computer system from 1983
  - Useable as a video game console or home computer (with extension cartridge)
  - Data Input to the system: Joysticks / Keyboard / Data loading from external device
  - Data Output: Screen / Audio / Data saving to external device

- Emulator O2EM
  - Open source C-project
  - Emulates all functions of the original system
  - Digital preservation functionalities implemented and shown on iPres2011

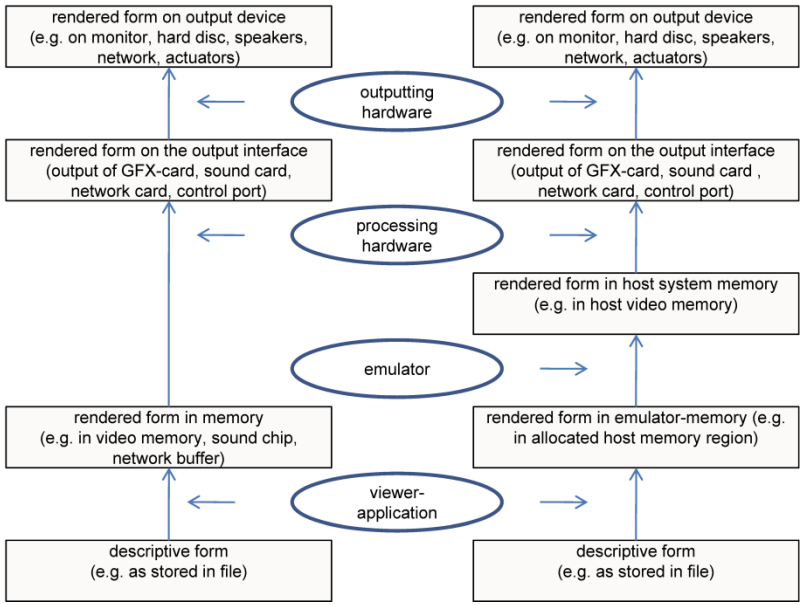
Recording of Events

- Controlled Environment
  - Deterministic execution
  - Rendering relying on external input (e.g. user input, network activity, access to files on the host system) -> must be reapplied on rerun

- In O2EM Emulator
  - Input as key presses, joystick input: rerouted input on host system to emulator -> record event and time
  - Files accessed in home computer emulation mode -> externally stored on host system, to be provided on rerun

Recording of Events

- Extraction of Data
  - Rendered data available on different levels inside the emulator: emulated memory / translated form on the host system (e.g. rendered screen)
- In O2EM Emulator
  - Save memory regions (main memory, video memory)
  - Save screenshots
  - Logged in event log to create memory / screen dumps at same point in time on rerun
  - Deterministic rendering -> same result files



Recording of Events

- Additional Events
  - Vertical Blank: period before drawing of a new frame is started. Used to synchronize events on the screen to a fixed timing. Allows conclusions about number of cycles executed, frames drawn related to the original system's timing
  - Emulation Start: Metadata about rendered file, name and version of emulator, date and time of execution
  - Emulation Stop: information that the rendering process was stopped, total number of cycles executed, number of frames drawn, total elapsed time

Recording of Events

- Execution Log – recording of the following information:
  - Executed CPU Cycles (number since emulation start)
  - Elapsed Time (in seconds since emulation start)
  - Drawn Frame (unique consecutive image produced by the video hardware of the system)
  - Recorded Event (code and full text)
  - Additional Information (e.g. key that has been pressed, direction of joystick, accessed external file)

1798	9267979	26,050	1276	254	Vertical Blank		
1799	9275237	26,070	1277	254	Vertical Blank		
1800	9276126	26,070	1277	1	Keypress	114	'r'
1801	9282496	26,090	1278	254	Vertical Blank		
1802	9289756	26,110	1279	254	Vertical Blank		
1803	9290645	26,110	1279	1	Keypress	107	'k'
1804	9297014	26,130	1280	254	Vertical Blank		

Automated Execution

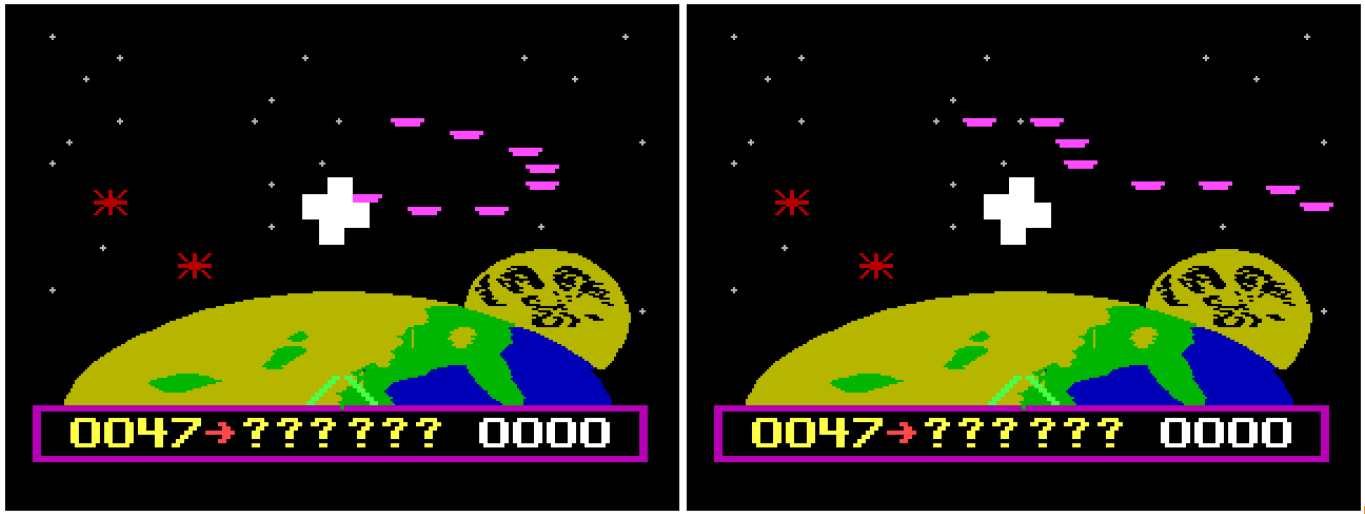
- Keyboard and Joystick Input
  - Manually recorded input events are applied at the same cycle count
  - Deterministic emulator -> cycle count is the same as original run
- Screenshot and Memory Data Extraction
  - Manually triggered memory/screen-dumps are executed at the same cycle count as the original run
- End Emulation
  - Emulation ended manually in the original run is stopped at the same moment using the automation script -> comparable total values of execution time/cycle count

Key Characteristics of Rendering Process

- Deterministic Rendering
  - Similar results for memory/screen dumps between different runs
- Cycles Executed vs. Emulation Time
  - Compared to original system using specified clock rate
- Executed Cycles per Frame
  - Correct timing per frame
- Time Needed to Draw a Frame/Frames per Second
  - Speed of emulator / slow downs with specific actions
  - Determine if emulation is fast enough / where slowdown occurs
- Accessed External Sources
  - Log implemented for all interfaces between host system/emulated system -> necessary external resources for digital object

Evaluation of Emulator O2EM

- Experiment: Videogame “Terrahawks”
- “Random” game events
- Interaction using joystick to play, keyboard to enter name for high-score
- Recording of game-play and replay using the log file
- Screenshot at different points in the rendering



Evaluation of Emulator O2EM

- Findings:
  - Difference in screenshots – emulator not working deterministic (problematic code: emulation of voice module)
  - Timing incorrect (no correct switch to PAL TV mode)
  - Timing not cycle accurate – timing sensitive code not working correctly

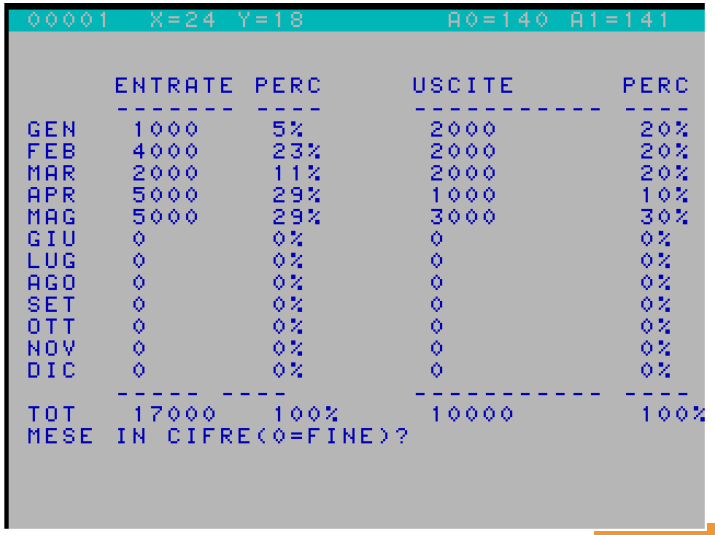
- Consequence:
  - Adjusting the timing
  - Correcting the errors
  - Make voice emulation independent of host system

Characteristic	Calculated	Measured
executed cycles per frame	7882	7259
executed cycles per second	394100	435540
frames per second	50	60
seconds per frame	0,02	0,0165

Table 1: Calculated versus measured key characteristics taken from the event-log of running Terrahawks in O2EM.

Evaluation of Emulator O2EM

- Experiment: Application “Cassa”
- Simple application: keep track of income/expenses per month over a year
- Loading of external data influences the rendering -> experiment: load program, load sample data and render the loaded data on screen



Evaluation of Emulator O2EM

- Comparison of memory dumps and screenshots
- Access to external files recorded in event log
  - Program file
  - Data file
  - Archival of files needed for rerun at a later point in time
- Execution without throttled speed
  - Simulation of a later rerun for validation of the emulation
  - Rerun performed in 7.7% of the original time

Characteristic	limited	no limit
total executed cycles	49201503	49201503
total frames drawn	6778	6778
total emulation time	136.426s	10.512s

Table 2: Characteristics for testing the application Cassa with original (=limited) and unlimited speed.

Conclusion

- Deterministic Emulation
  - Comparison possible for “random” objects
  - Errors in emulators detectable
- External Data
  - Data recorded for replay (user input and externally stored data)
  - Simulation of external resources (e.g. web services)
- Key Characteristics
  - Deviations in handling the timing can be detected
  - Characteristics dependent on the system

Conclusion / Future Work

- Automation of Evaluation
  - Application of external data for automation necessary
  - Speeding up the validation process is possible

- Application on more complex environments necessary in future work
- Definition of software architecture guidelines for emulation developers
  - Deterministic execution of all emulator parts
  - Input data capturing on interface host/emulated environment
  - Automation of execution with captured input data
  - Standardization of data extraction (what to extract and format)
  - Standardization of log formats to exchange between emulators

Thank you for your attention.

mguttenbrunner@sba-research.org

http://www.sba-research.org

www.ifs.tuwien.ac.at/dp