

Preserving the software in software based art

Brian Matthews,

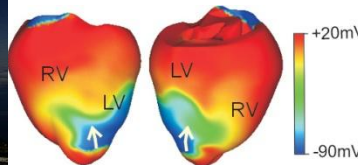
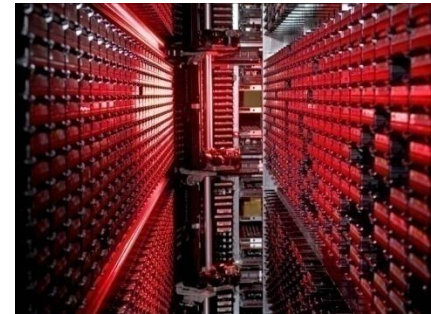
Leader, Scientific Information Group
STFC e-Science Centre

Arif Shaon, Juan Bicarregui, Catherine Jones, Esther
Conway, David Giaretta, Brian McIlwrath



Science and Technology Facilities Council

- Provide large-scale scientific facilities for UK Science
 - particularly in physics and astronomy
- E-Science Centre – at RAL and DL
 - Provides advanced IT development and services to the STFC Science Programme
 - Manage science data
 - Keep the results alive and available
- STFC Interest
 - Keeping science data usable for long periods
 - Specialised scientific analysis software
 - Needs to be kept along with the data



So, why am I here ?

- JISC projects (2007-09):
 - Report on the Significant Properties of Software
 - Tools & Guidelines for the preservation of software as a research output
 - Also SSI/Curtis Cartwright (2010-11)
- Software very large topic
 - Diversity in:
 - *application of software*
 - *software architecture*
 - *scale of software*
 - *provenance*
 - *user interaction*

Developed a vocabulary for talking about software preservation.

- Apply it to Digital Art



Role of Software in Digital Art

What might we preserve in each case?
(thought the artist might think differently)

- Digital Painting
 - James Faure Walker: Godwin (2007)
 - Preserve output (physical or digital)?
- Computer-Generated Art
 - Alex McLean: Forkbomb (2001)
 - Preserve the source code ?
- Digital Installation
 - Noah Wardrip-Fruin: Screen (2002)
 - Preserve whole environment ?
- Algorithmic art
 - Eno: 77 Million Paintings (2006)
 - Preserve algorithm + seeds ?
- Interactive applications
 - Maurizio Bolognini : Collective Intelligence Machine (2006)
 - Complex - Preserve Performance ?
 - Document Experience?



```
#!/usr/bin/perl -w
```

```
use strict;
```

```
die "Please do not run this script  
without reading the documentation"  
if not @ARGV;
```

```
my $strength = $ARGV[0] + 1;
```

```
while (not fork) {  
    exit unless --$strength;  
    print "0";  
    twist: while (fork) {  
        exit unless --$strength;  
        print "1";  
    }  
}
```

```
goto 'twist' if --$strength;
```



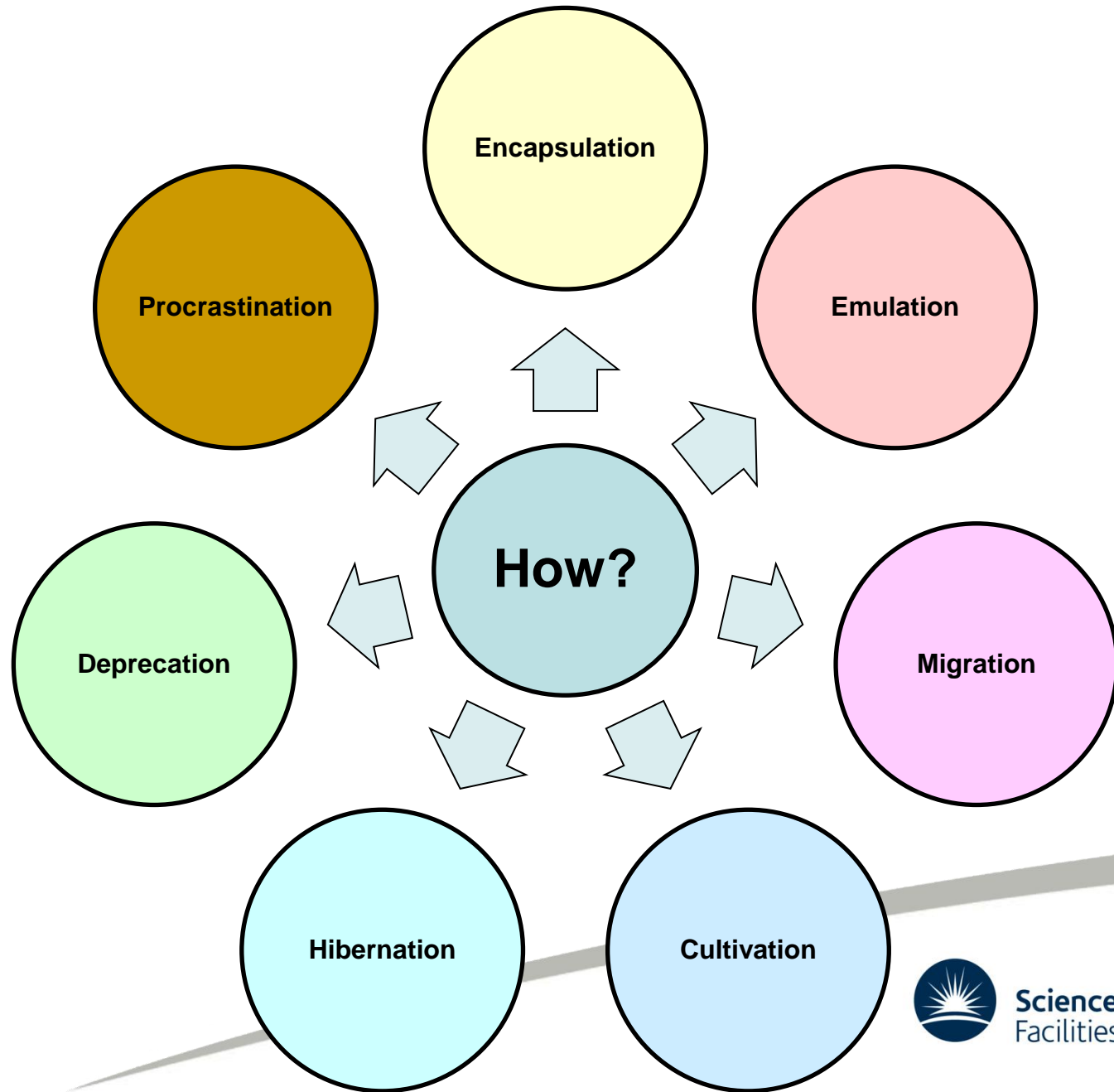
Issues

- Software varied and complex
 - May involve bespoke code
 - May involve specific hardware configuration
 - May involve an output (only)
 - May involve standard packages
 - May be “just” algorithm
- Lots of dependencies (as with any software)
 - Hardware
 - Software versions
 - Operating systems
 - Environment
- Licensing
- Subject to decay
- One size does not fit all.
- Depends where the key part of the artistic intention lies.

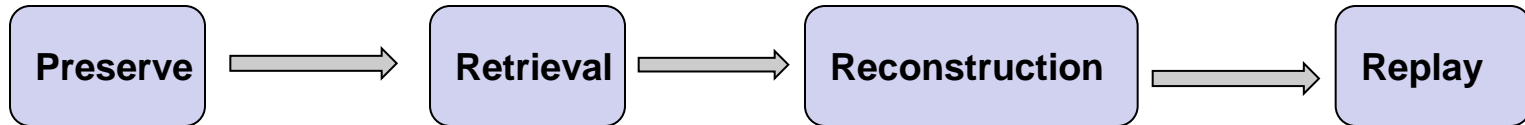
But need some vocabulary to discussing its management



Strategies



Software Preservation Steps



- What do we do when we preserve software?
 - Identify a number of related digital artefacts to preserve
- What do we do when we want to use it again ?
 - Find the right software artefacts to use
 - Reconstruct them into a executable system.
 - Replay the execution of the system



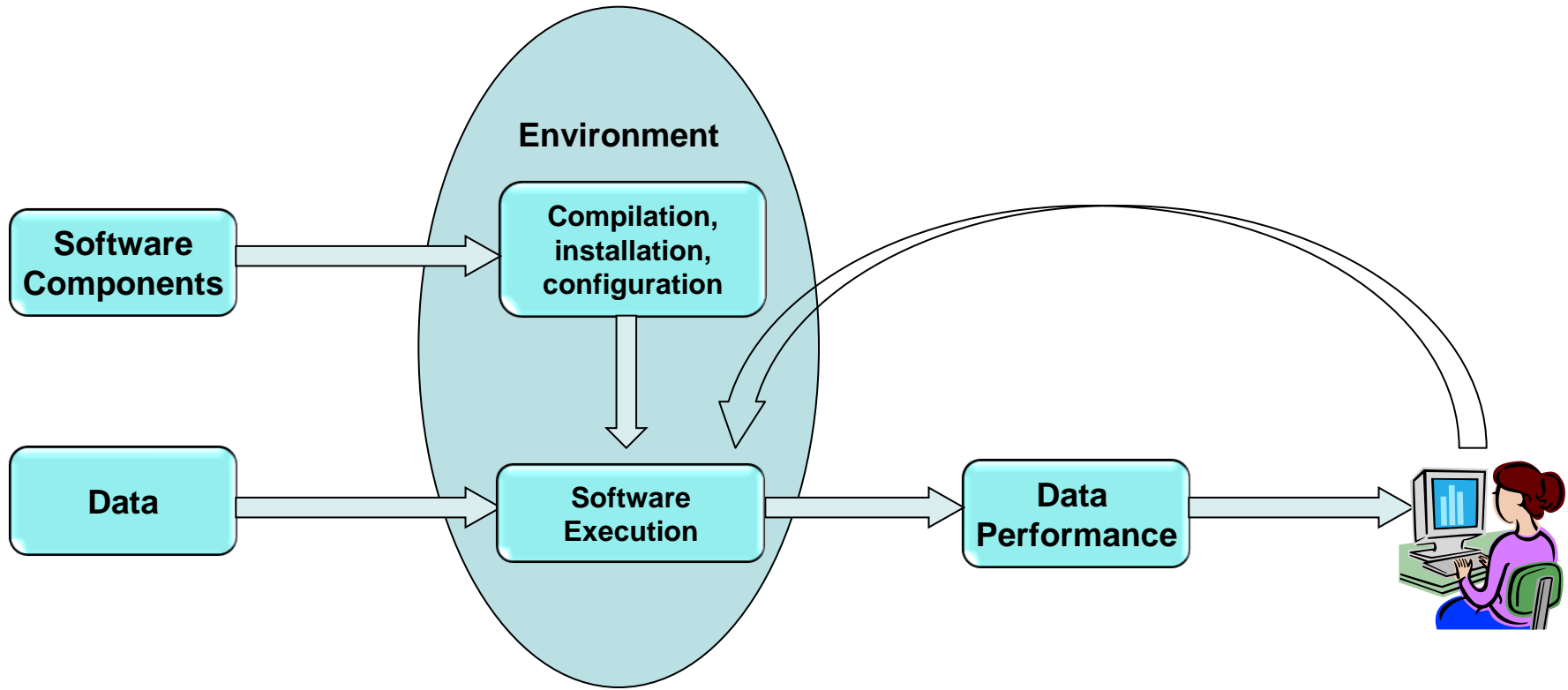
Preservation Properties of Software

What attributes of software do we need to take into account for long-term preservation?

- **Functionality**
 - what it does and what data it depends on
- **Environment**
 - platform, operating system, programming language
 - versions
- **Dependencies**
 - Compilation dependency graph
 - Standard libraries
 - Other software products
 - Specialised hardware
- **Software is a Composite**
 - Collection of modules
 - Specifications, Configuration scripts, test suites, documentation
- **Architecture**
 - Client/server, storage system, input / output
- **User interaction**
 - Command line, User Interface
 - User model

How do I judge now that what I have preserved is “enough” ?

Performance Model for Software



- Based on the NAA performance model for digital preservation
 - The test of the success of our preservation is the ***performance of the data for the audience***



How do I judge now that what I have preserved is “enough” ?

- Track the **Significant Properties**

“those characteristics of digital objects that must be preserved over time in order to ensure the continued accessibility, usability, and meaning of the objects”

- Testing the data performance
 - judge **adequacy** of the software performance.



Adequacy of Software Preservation

A software package can be said to perform adequately relative to a particular “significant property”, if in a particular performance it preserves those significant properties to an acceptable tolerance.

- Significant properties are evaluable features of the performance
- Supply **test cases** to evaluate the adequacy against properties
- Assess the value of the performance
- ***Can be generalised to any digital object***



So what does this mean for Digital Art ?

- Preserving software artefacts
 - Gather as much contextual information as possible
 - *Software (re-)engineering*
- May not need to preserve it all
 - Individual cases treated with different strategies - a nuanced approach
- Does it keep on satisfying the artist's intention?
- Does it keep on satisfying the audience's experience?
 - Identifying appropriate significant properties
 - Uniqueness of code
 - But decide them in advance
 - Testing algorithmic art
- But digital art can hard to sustain
 - Be prepared for reinterpretation.

With software, as with painting only the user can tell the difference- James Faure-Walker



Variable Media Network

<http://www.variablemedia.net/e/index.html>

An example of where some of these issues have already been considered in preserving artworks.

“For creators working in ephemeral formats who want posterity to experience their work more directly than through second-hand documentation or anecdote, the variable media paradigm encourages creators to define their work independently from medium so that the work can be translated once its current medium is obsolete.”

Storage: store a work physically, mothballing dedicated equipment or archiving digital files on disk.

Emulation: devise a way of imitating the original look of the piece by completely different means.

Migration: To migrate a work by upgrading equipment and source material.

Reinterpretation: reinterpret the work each time it is re-created.

Mark Napier: Net Flag, Java Applet 2001



<http://sigsoft.dcc.rl.ac.uk/twiki/bin/view>

<http://www.e-science.stfc.ac.uk/projects/software-preservation/preserving-software.html>

<http://www.software.ac.uk/>

Thank You

Questions?

brian.matthews@stfc.ac.uk

<http://www.e-science.stfc.ac.uk>



Science & Technology
Facilities Council