soundsoftware.ac.uk

SoundSoftware.ac.uk: Sustainable Software for Audio & Music Research

Mark Plumbley, Chris Cannam and Luis Figueira
Centre for Digital Music
Queen Mary, University of London

EPSRC
Engineering and Physical Sciences Research Council

Overview

- Audio & Music Research:
  - Researchers, issues, and opportunities
- Research Pipeline:
  - Reproducible research & Reusable research
- Opportunity: Software Sustainability
  - SoundSoftware.ac.uk project
- Activities & examples
- A software repository: code.soundsoftware.ac.uk
- Conclusions

Audio & Music Researchers

Wide range of backgrounds

- Signal processing, electronics, computer science, music, information sciences, dance & performance, data sonification, music therapy, biology, ...
- Common interest in the use of audio and music in research
- Not all want to write own code (or have skills to)
- Different platforms: Mac, Linux, Windows – but typ. desktop
- Many environments:
  Matlab, Python, Max/MSP, SuperCollider, VST plugins

Issues for the field

- Software designed for legacy platforms / not maintained (Sun SPARC, NeXTSTEP, Dec Alpha)
- PhD students graduate, staff move:
  -> web pages containing original software/data lost
- Code not released (not a priority, not “ready”)
- Software/datasets in slightly different versions (error corrections, enhancements)
- Copyright issues of datasets (e.g. audio of Beatles tracks cannot be placed on the web)

Example of issues: Beat Tracking

- Classic algorithm (Scheirer, 1998) in legacy C++ for DEC Alpha, original website lost, various modified versions "passed around"
- Another early algorithm (Goto, 1994-8) written for a parallel architecture, computer no longer exists, code never released
- UK algorithm (Hainsworth, 2005) in Matlab but with Windows-only DLL component (limits its portability)
- Another key algorithm (Klapuri et al, 2006) in Matlab, available from authors, requires contract preventing redistribution
- Several other algorithms never released, but researchers may help individually (Cemgil, 2000; Laroche 2003; Peeters 2005).

So; hard for new researchers to compare with those algortihms!

Example of potential: Beat Tracking

- Onset detection (Bello, 2003) in beat tracking (Davies, 2005)
- Davies (Matlab), ported to SuperCollider by Collins (2006)
- Davies alg ported into cross-platform C++ Vamp plugin for Sonic Visualiser / Sonic Annotator (on EPSRC OMRAS2 project)
- Inspired Max/MSP beat tracking system (Robertson, 2007)
- Used in beat-synchronous audio effects (Stark, 2008) - developed in Matlab, ported to real-time VST plugins.
- Rhythm morphing (Hockman & Davies, 2008) originally Matlab, ported into a C++ library (on EPSRC Platform Grant)

Helped by (a) personal continuity, (b) funding for "extra step"

Ideal Research Pipeline

Researcher A ("Producer")

- Read background papers
- Do own research
- Publish paper X

Researcher B ("Consumer-Producer")

- Read paper X
- Understand/reproduce results in paper X
- Do more research building on X
- Publish paper Y that cites X / produce product that uses X

... and so on.

Researchers write code (but badly)

Typical Research Skills:

- Maths
- Experiments
- Analysis
- Proofs
- Writing & presenting
- Matlab/Gnuplot/LaTeX

Typical Coding Skills:

- Design
- Documenting
- Version control
- Unit testing
- APIs
- C++/Java/Python

Some can do both well. But they are uncommon.

See e.g. (Hannay et al,, 2009)

So: Real Research Pipeline

Researcher A ("Producer")

- Read background papers
- Do own research (including lots of coding)
- Publish paper X (not enough space for all the code)

Researcher B ("Consumer-Producer")

- Read paper X
- Can't reproduce or use results in paper X
- Tear out hair
- Give up / do something else

NB: A and B may be in same group (or same person later!)

Reproducible Research

(Buckheit & Donoho, 1995; Vandewalle et al, 2009)

Idea: researchers should be able to reproduce the work of others.

Research used to be "reproducible" from the paper alone.

Computational research (including audio & music) is now very complex: algorithm, parameters, datasets, etc.

The paper alone is not enough to reproduce the research

So, we need

- The paper (ideally Open Access)
- The code (ideally Open Source)
- The data (ideally Open Data)

Well-known example: WaveLab (Buckheit & Donoho, 1995)

Why is Reproducible Research Hard?

Researchers might not release code because

- Copyright/IP – maybe they could sell/license it later?
- Badly written – would be embarrassing!
- No time to tidy up – not a priority (“It’s not research”)?

Researchers might not release the data because

- They don’t have the rights to (e.g. my CD collection)
- We spent ages collecting it, why give it away?

They might even be thinking:

- “Someone else might use it to do better research than me”
- “Someone might notice something wrong with my research”

Reusable research

Even "Reproducible" might not get to the people who need it:

- Signal processing people use Matlab –Musicologists don't
- Code no longer works when they come to use it

At Centre for Digital Music & Digital Music Research Network

- Transdiscipline – cross traditional discipline boundaries
- "User-researchers" (outside field) different to "peer-researchers" (in own field)
- Additional work needed to make research usable
- New generation (e.g. PhD students) can cross these boundaries

Opportunity: Software Sustainability

Funding call for EPSRC ("e-Science? What's that?")

Proposal - provide a Service to:

- *support the development and use of software and data*
- *to enable high quality research*
- *in the audio and music research community*

In other words:

- Help audio & music researchers to make sustainable and reusable research software
- Help other researchers use audio & music research through sustainable research software

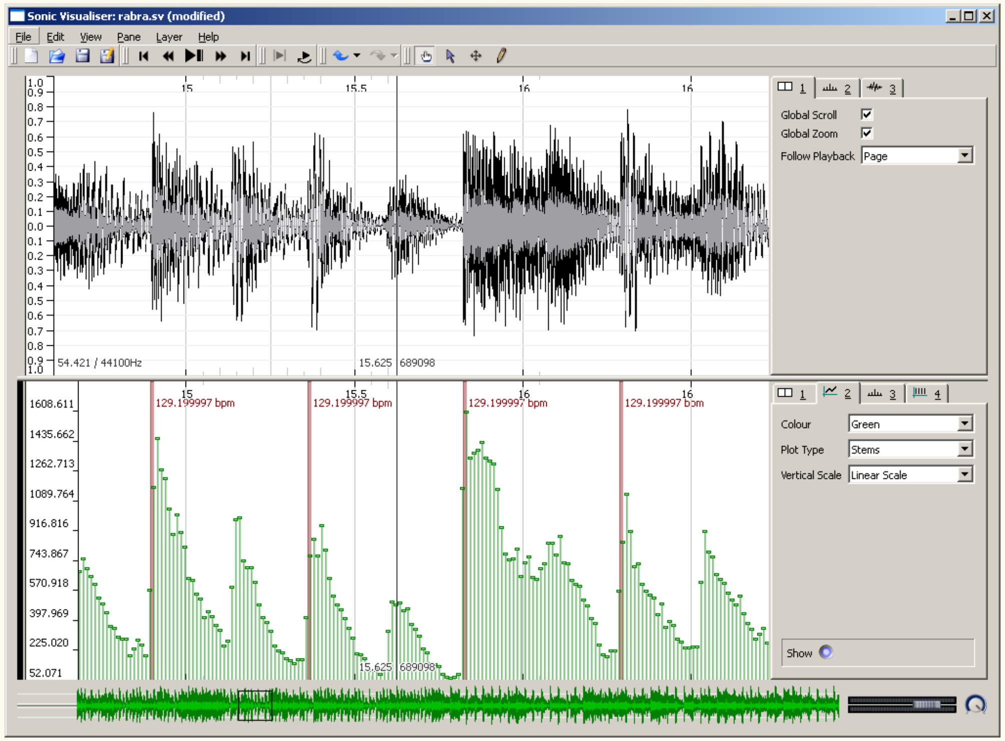
and so, make audio & music research have an **IMPACT.**

SoundSoftware.ac.uk: Planned Activities

- Employ software developers, to
  make existing research software robust & usable
- Training for researchers, to
  write robust & reliable research code
- Help for academics / research project managers, to
  build software development into research projects
- Curation of data and software, to
  help future researchers find what they need

Example software:
Long-term software reuse: Sonic Visualiser

- Multi-purpose visualiser for sound recordings
- Open source
- Built from modular libraries which can be used for other applications
- Introduced plugin architecture (Vamp plugins) for analysis tools based on research
- Used by audio researchers, musicologists, etc

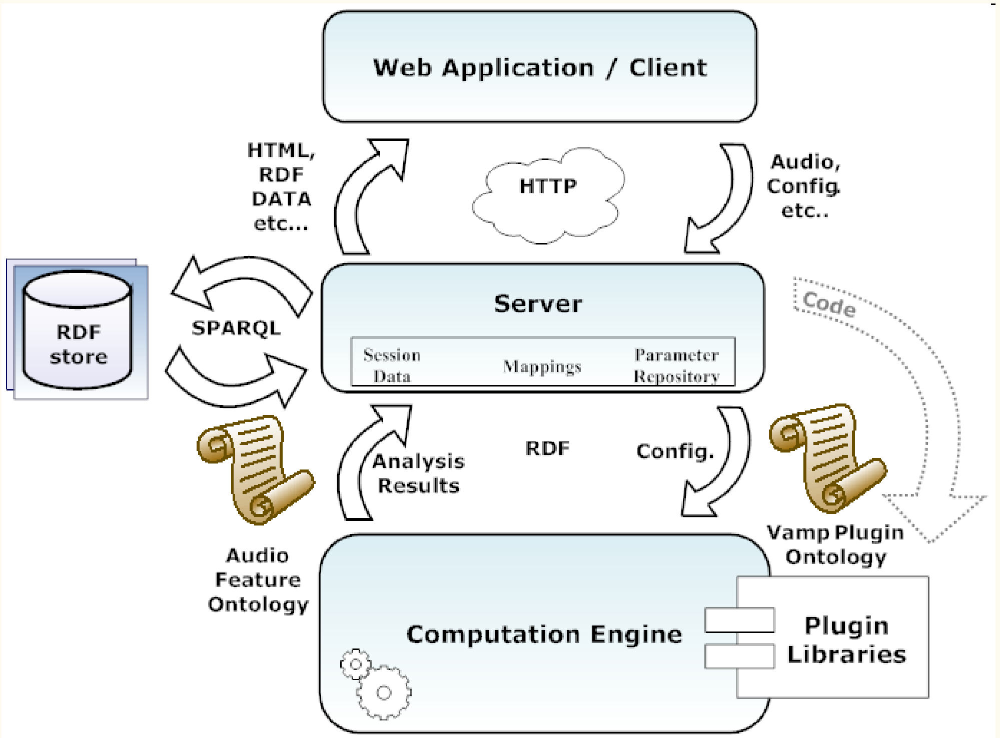


Global Scroll ✓
Global Zoom ✓
Follow Playback: Page

Colour: Green
Plot Type: Stems
Vertical Scale: Linear Scale

Example service: Maintaining systems and services: SAWA

- Allow access to rdf repositories with audio and music related meta-data (150000 audio tracks)
- Web-based audio features extraction and similarity search services
- Can be seen as a continuous service for researchers in music/audio similarity research



Example data:
Blind Audio Source Separation DB (BASS-DB)

- Used for Blind Audio Source Separation contest presented at ICA 2006 conference.
- Many researchers used this database for evaluating their algorithms in following years.
- The database is now superseded by later evaluation campaigns, may disappear in future.
- Aim: archive and maintain the database, so researchers can continue to compare their own algorithms against previously published results.

Training Researchers: November 2010 Autumn School

- PhD students etc. are typically not taught how to build robust software they need for their research in a systematic way
- Many self-taught – lead to big problems later
- Autumn School:
  Software Carpentry for Audio & Music Research
- Based on "Software Carpentry" course from Greg Wilson (U Toronto)
- Nov 2010: Ran School with ~20 selected researchers
- Future: Release videos; distance/self-paced learning materials

Software Repository for Audio & Music Research Software

What is a software repository?

- Provides version control for your research software
- Keeps track of changes as you develop
- Avoid risk of breaking working version
- Get back to previous versions
  - E.g. Version used for the figures in your journal paper
- Other features that might be useful:
  - Bug tracking, collaborative development, documentation, public version releases

Alternative repositories

Version control etc at own institution

- Good idea if you can do this yourself! But…
- Does your research group/IT support provide it?
- Who should you ask? (We might be able to help)
- Not so easy for collaborative work with other institutions

SourceForge, Google Code, etc

- Only for Open Source projects – perhaps not so good for pre-release research, or software to be licensed?
- Not designed for **research** software: not easy to associate with publications, data, etc that use the software.

Repository Launching Today: code.soundsoftware.ac.uk (beta)

- Available to UK audio & music research community
- Source code revision control (Mercurial)
- Bug/feature tracking; Wiki for documentation
- File uploads and sharing
- Aimed to start with small projects (“just working on”)
- Some things still under development: (Documentation, tutorial material, tools, links to published papers and data)

How do I start?

- Register Now! See **code.soundsoftware.ac.uk**
- Create projects as soon as your account is active

Conclusions

- Issues for Audio & Music Research
- Wide range of researchers, platforms, languages
- Reproducible research is hard, reusable research is harder
- Recognition of "Software Sustainability" as an issue
- SoundSoftware.ac.uk – help researchers make impact
- Examples of software & data
- Nov 2010: Autumn school for researchers
- Today: Repository launch: code.soundsoftware.ac.uk
- Register today! Start building **sustainable** research software!