A National Agenda for Research Software

DRAFT

Tom Honeyman (Software Program Manager), Andrew Treloar (Director of Platforms and Software)

Version 0.9

11th June 2021

DOI: 10.5281/zenodo.4940274

ARDC is enabled by NCRIS

NCRIS National Research Infrastructure for Australia
An Australian Government Initiative

CONTENTS

Executive summary **4**

The Challenge **7**

Research Software 8

Software Authors 10

Drivers for change 11

Summary 12

See, Shape and Sustain **13**

See Research Software 13

Shape Research Software 14

Sustain Research Software 15

Bringing About Change **17**

Infrastructure 17

Guidance 18

Community 19

Advocacy 21

Summary 22

A Research Software Agenda for Australia **23**

See Research Software: 23

Shape Research Software: 23

Sustain Research Software: 23

Stakeholders **24**

Authors of Software 24
Researchers 24
Nascent Research Software Engineers 24
Research Software Engineers 25
Platforms Developers 25
Professional Support Staff 26
Informatics Support Staff 26
Research Software Support Staff 26
Research Institutions 26
Research Software Engineering Services 27
Publishers 27
Funders 28
Data Generating Facilities 28
National Digital Infrastructure Providers 29
The Australian Research Data Commons (ARDC) 29
Other National Stakeholders 29
Summary 30
Next Steps **31**
References **32**

Executive summary

This agenda is seeking that "Research Software is recognised as a first-class output of Research". To achieve this we must be able to "See, Shape and Sustain Research Software". Software is pervasive in modern life. In the research domain a majority of researchers say that their work would simply not be possible without the use of software. Despite this, software is an often invisible part of research, produced quickly without the application of software engineering principles, and often struggles to be maintained beyond a funding window.

Making research software first class ensures that Australia will be able to fully utilise the value that research software represents. Making software creation and use more visible increases trust in research processes. Shaping cutting edge research software for broadest and easiest use and reuse at the point of creation means that research software can more immediately translate into benefits for the economy, environment and society. Sustaining valued research software means that it can become resilient and continue to provide those benefits.

To clarify and direct discussion, a framework with three layers ("See", "Shape" and "Sustain") is introduced. These cluster distinct sets of needs, drivers, and authors that lead to distinct types of research software:

	Outputs	***Core Concern***	***Authors***	***Primary Driver***
See	***Research data processes*** *in research software form*	***Sharing/Availability***	***Researchers***	***Research integrity***
Shape	***Novel methods and models*** *as research software*	***Best practice software engineering***	***Nascent research software engineers*** *(possibly with Research Software Engineers)*	***Research excellence and impact***
Sustain	***Accepted and broadly used methods and models*** *as research software*	***Maintenance***	***Research software engineers***	***Stable research infrastructure***

Other countries have shown that these concerns can be addressed at a national level through the coordinated action of identified stakeholders. This agenda describes existing areas of activity in Australia addressing "See", "Shape" and "Sustain", before further breaking them down using a concise version of the "Strategy for Culture Change" from the Centre for Open Science. Each layer is considered against what makes change possible (Infrastructure), easy (Guidance), normative (Community) and

codified (Advocacy). Combining the three layers against these four areas for change results in a grid of twelve actions for this agenda:

	Infrastructure	***Guidance***	***Community***	***Advocacy***
See	*(1) Ensure that research software publication, citation and other informatics infrastructure is fit-for-purpose, and is ready to meet newly emerging practices*	*(2) Connect researchers, through support professionals, to guidance on informatics infrastructure and emerging best practices*	*(3) Build and sustain a national community of practitioners interested in informatics infrastructure and in the behaviours that inform it*	*(4) Create the policy and incentives environment to encourage code availability*
Shape	*(5) Ensure that new software assets are created as infrastructure built for easiest and broadest meaningful reuse from inception*	*(6) Connect researchers and research software engineers to guidance to develop research software infrastructure for easiest and broadest reuse*	*(7) Build and sustain local, regional and national communities of practitioners who bridge the gap between research and software engineering practices*	*(8) Create the policy and incentives environment that recognises research software as a first class output of research*
Sustain	*(9) Ensure enduring research software infrastructure is available to researchers through a skilled, diverse, well distributed and sustainable workforce of research software engineers*	*(10) Connect research software engineers and their employers to guidance for career and professional concerns in research software engineering*	*(11) Build and sustain a national community of professional research software engineers*	*(12) Create the policy and incentives environment that supports the development and maintenance of critical research software infrastructure*

Relevant stakeholders in this change are described with reference to whole layers or specific actions. The identified stakeholders are software authors (researchers, (nascent) research software engineers and research platforms developers), support staff (informatics and research software skills), research institutions, research software engineering service units, publishers, funders, data generating facilities,

and national digital infrastructure providers. AOASG, RSE-AUNZ and AeRO are named specifically. Here, stakeholders with the strongest identified interest are mapped below to corresponding actions.

	Infrastructure	***Guidance***	***Community***	***Advocacy***
See	*(1) Research Institutions, National Digital Infra., Platforms Devs, Publishers*	*(2) National Digital Infra., Informatics Support, Research Institutions*	*(3) Informatics Support*	*(4) Research Institutions, Publishers, Funders, Data Generators, AOASG*
Shape	*(5) (Nascent) RSEs, Research Institutions, RSE Services*	*(6) Research Software Support, RSE Service, Research Institutions*	*(7) (Nascent) RSEs, Research Soft. Support, Research Institutions, RSE Services*	*(8) Research Institutions, Publishers, Funders, RSE Services, AOASG*
Sustain	*(9) RSEs, Research Institutions, RSE Services, Data Generators*	*(10) RSEs, Data Generators, Research Institutions*	*(11) RSEs, RSE-AUNZ, Research Institutions*	*(12) RSEs, Research Institutions, Data Generators, RSE Services*

This work is an expression of interest by the ARDC to extend from the actions across the See layer to consider initiating activities across actions in the Shape and Sustain layers.

The release of a draft agenda is to facilitate a consultation phase with the sector to confirm or adjust identified interest against a measure of actual interest in and ideas for activities addressing the actions of this agenda. The subsequent phase in this process will be a final form of the agenda with ordered priorities, and an implementation plan describing the ARDC's commitment to priorities arising from this process.

The Challenge

Software is pervasive in modern life, and research is no exception to this. In a survey conducted by the UK Software Sustainability Institute [1], around 90% of researchers acknowledged the importance of software, with about 70% saying that their work would not be possible without the use of software. In addition to a general dependence on software for conducting research, software is a common output of research, with a literature analysis showing around 25% of research projects result in new software [2]. But this is a measure of the authors of code that have made their code available. In an environment where code availability was more highly valued and recognised, this percentage may well be higher.

In contemporary research, generation, handling and analysis of data almost always involves software. Digital data requires software. Highly specialised digital research data almost always requires highly specialised research software that encodes specific research data processes. These processes are the lens through which research data become more valuable. However, in looking through this lens, the value is more commonly attributed to the research data rather than the combination of data plus software. Considering the appropriate handling of this combination is in line with the 2021 OECD recommendation concerning access to research data from public funding [3] (bold added):

> [We must recognise] that re-use and value of data can depend on the availability of relevant metadata, **algorithms, code, and software**, from public funding together with information on **workflows** and the computational environment used to generate published findings...

While the role of software engineer and value of software in the modern economy is broadly recognised, the value of the highly skilled coder-scientist [4], the impact of their research software outputs [4], and how much research depends on their continued labour [5], is less recognised.

Research software is a highly specialised output which has had great impact on society, the economy and industry, but despite being an output with the same, or even greater potential for impact, it is not treated as a first class output of research. Recognition is the critical change called for in this agenda.

Research Software is recognised as a first-class output of research

This will take some time to achieve, and will require a willingness to change by a broad range of stakeholders, including researchers, research institutions, policy makers, funders, publishers and more.

Traces of the scholarly impact of research software can be seen in some highly rated scholarship [6], but for software to be recognised as a scholarly output in its own right (i.e., a “first class output of research”), **what is needed is a systematic change**. This change must make the true extent of software’s role in research visible, shape valuable software for the broadest possible use in research, and make the software components that underpin a broad range of modern research sustainable.

Thankfully, there are several examples of activities that have successfully addressed these challenges around the world. Chief amongst them is a decade of impactful work from the Software Sustainability Institute in the UK (SSI) and the Society of Research Software Engineering. Their work has heavily informed this agenda, and provided evidence that a national approach is possible.

Similar or related work is also being conducted internationally by (amongst others):

- the eScience Centre in the Netherlands
- the Canadian Network for the Advancement of Research, Industry and Education (CANARIE)
- the planning for a US Research Software Sustainability Institute (URSSI) and establishment of the Better Scientific Software (BSSw)
- the software working group of the Alliance of Science Organisations in Germany
- the Nordic e-Infrastructure Collaboration (NEIC), including the work on the CodeRefinery
- the Carpentries community of instructors, trainers, and courseware authors around the world, and of the various governance bodies in the Carpentries

ARDC is now seeking to build on initial work undertaken in the Nectar project and guide a discussion on how Australia can rise to the challenge of recognition of research software as a first class output of research.

The remainder of this section defines research software and needs associated with different types of software. It identifies who writes this software, and considers what drives change in practice. A summary combining this detail is given at the end of this section.

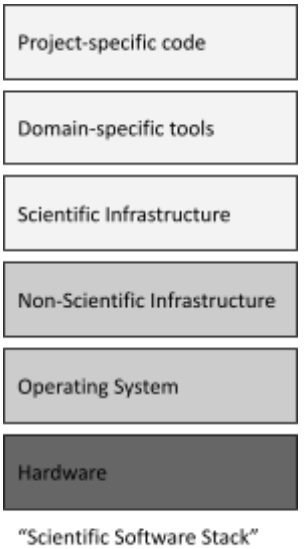
Research Software

A broadly used and accepted definition for research software [1] is:

> Software that is used to generate, process or analyse results that you intend to appear in a publication (either in a journal, conference paper, monograph, book or thesis). Research software can be anything from a few lines of code written by yourself, to a professionally developed software package. Software that does not generate, process or analyse results - such as word processing software, or the use of a web search - does not count as 'research software' for the purposes of this survey.

This section aims to narrow and reshape this very broad definition down to more specific types of software that help to define the scope of this agenda.

As indicated by the quote above, research software is most commonly framed in terms of its use rather than its creation. It is sometimes described as a tool, an output and an object of study [7] or as a part of research discovery, including model simulation and data analytics [8]. Hinsen [9] has described Scientific Software in terms of a stack of dependent layers (shown to the right). Some of these layers



"Scientific Software Stack"
Konrad Hinsen

originate in research, and some do not.[1] All combine to produce the end result, and that result depends on all these layers being available. In this agenda, the primary interest is in the layers originating in research or research adjacent domains.

The framework that drives this agenda is inspired by the top three layers of Hinsen's stack. Considered here are research software of the following kinds, based on the nature of what is captured by that software[2]:

1. An output capturing **research data processes** as software
2. **Novel methods and models** captured as software for others to use
3. **Accepted methods and models** captured as software for others to build on

The separate needs, authorship and drivers for these three types of research software form the basis of a framework for this agenda. As with Hinsen's stack, these three types of software are assumed to be built upon a layer of software infrastructure outside research and then operating systems and finally hardware.

Outputs capturing **research data processes** such as data manipulation, handling or analysis may be realised as software in the form of short scripts, macros, or even ephemeral command-line "one-liners", or may be more substantial pieces of code[3] that are all used within the life of a research project for a specific purpose. They may also occur in the form of computational workflows within a workflow environment or engine. This kind of software is not usually intended for reuse, but if it is then as characterised here it would be for repetitive tasks within the bounds of a project, towards a publication or similar. In many disciplines this includes what is often referred to as "(analysis) code" but it also includes any code developed in the steps prior to analysis including preparation and exploration.

Methods and models may be captured as utilities, tools, applications, libraries, modules or frameworks. These are developed with the intention that they should be used by others outside the research project within which they were originally developed. They may be computational workflows, but are more likely specific components of a larger workflow. They may also be made available within platforms or services, but this agenda is primarily concerned with the underlying software methods and models (as outputs), rather than the mode of presentation or ease of use and availability.

Methods and models are differentiated here as either accepted and broadly used, or novel ones representing the cutting edge of theory or discovery. **Accepted methods or models** are likely over time to have found a broader audience within or even beyond a domain or discipline. Accepted methods and models may be implemented more than once (in multiple programming languages, or presented in different ways). **Novel methods or models** have narrower audiences, will most likely only be implemented once (at least initially) and may have a short life after release (or otherwise go on to be accepted and broadly used).

[1] This is not to diminish the fact that all new technologies ultimately stem from research.
[2] Hinsen describes the top three layers of his stack differently to what is described here. In this document, the second ("novel methods and models") and third ("accepted methods and models") layers described here are a slight departure from "domain-specific tools" and "scientific infrastructure" in Hinsen's stack. The focus here is on the maturity of the software and ideas captured within it, rather than the specificity of the domain of application.
[3] This is most notably the case for code developed in the context of specialised resource consumption (HPC or HTC) where efficient use of limited resources is a major concern.

The distinctions between these three types of software are more obvious when considering three core concerns for research software: (1) the visibility of research software (encompassing sharing, availability, discoverability and credit), (2) the sufficient application of software engineering principles to best enable reuse, and (3) the maintenance of software. As shown in the table below, each of these issues align strongly (as indicated in bold) with one type of software, while the other issues are either irrelevant, partially or wholly addressed or at least less pressing.

	Visible	*Fit-for-purpose*	*Maintained*
Research Data Processes captured as software	**Not generally shared, but needs to be**	Minimally required, or limited to some domains	Rarely maintained, sometimes curated and less likely to be reused
Novel methods and models captured as software	Usually shared, visible to varying degrees, room for improvement	**Notably improved by application of best practice**	Not maintained (at least, not straight away), as it is new
Accepted methods and models captured as software	Usually shared, visible to varying degrees, room for improvement	Usually well structured for reuse, but room for improvement	**Struggles to be maintained, but needs to be done well**

Outside of these three types is the vast array of both common and specialist software where the primary audience is outside of research. There are many tools commonly used both inside and outside a research context, but developed or maintained outside of it. Because the focus is on outputs of research, these other types of software[4] are considered out of scope for this agenda.

All software interrelates in a fragile ecosystem of dependencies [9]. Common or novel methods for data manipulation and handling and analysis may well be expressed as the methods and models described above. When a script capturing a research process is used to analyse data after transforming it to a suitable shape, it is very likely that it will use other software (including what is described as methods and models here) to do that. And in turn, novel methods and models may well build upon accepted ones. When structured this way (as a hierarchy of dependencies), it is important to realise the two way relationship between these types of software. Research software often uses other research software and at the same time, when captured this way, it is a record of the use and impact of research software.

Software Authors

The three kinds of software described above are usually developed by different types of authors.[5] Defining different types of authors creates different points of focus for action later in this agenda.

[4] For further consideration, see [9], where “non-scientific software infrastructure”, “operating systems” and “hardware” are described as the layers that underpin the whole scientific stack. These layers are certainly important for the overall stability of research software.

[5] Daniel S. Katz made the connection in alignment between the layers of the “scientific software stack” described by Hinsen [9] and the three stakeholder types. See slide 16 in https://www.slideshare.net/danielskatz/research-software-susainability-226477374

Outputs capturing various research data processes as software are often written by **researchers**. They are most likely self taught programmers with a limited knowledge of software engineering practices. If they are not directly authoring software, they may instead be using systems like workflow engines which output re-executable workflows. While this stretches the notion of authorship, for the purposes of this agenda, this workflow output is also considered in scope. Most commonly though, researchers are (also) using software tools without capturing research data processes in a re-executable form.

Novel Methods and Models as software are usually written by researchers, possibly self-taught, or possibly formally trained in software engineering. As characterised here, they are likely to have a close relationship to the research area in which they are operating. For the purposes of differentiating this group, they are referred to in this agenda as **nascent research software engineers** (RSEs), but they may not recognise themselves under this title, nor wish for a career as a Research Software Engineer. Nascent RSEs are often PhD students or early career researchers, a career stage usually associated with stretches of fixed-term employment, and movement across institutions.

Accepted Methods and Models as software are usually written and maintained by a relatively small number of career **research software engineers**[6]. Research software engineers are either formally trained as software engineers, or have devoted considerable time to acquiring these skills on their own. They will ideally have a good understanding of the research areas they are authoring or maintaining software in, possibly to postgraduate level.

Drivers for change

When treating research software as a uniform whole it can be difficult to see what will drive recognition of research software as a first class output of research. In the previous sections, distinct needs and author profiles were presented, aligned with each type of research software. Understanding the different drivers across each will allow us to propose separate actions for advocacy later in the agenda.

By breaking them apart into the three types described above, and considering the needs of the corresponding types of authors, three clear and separate drivers for change in research software emerge:

1. research software that captures research data processes is driven by the need for greater **research integrity**
2. research software that captures novel methods and models is driven by the pursuit of **research excellence and impact**
3. research software that captures accepted methods and models is driven by a need for **stable research infrastructure**

[6] See also "Research Software Engineers" under Stakeholders for more detail on this role.

Summary

The table below shows different clusters of concerns. Each layer has a distinct cluster of outputs, needs, authors and drivers, all summarised in the table below.

	Outputs	Core Concern	Authors	Primary Driver
See	**Research data processes** in research software form	**Sharing/Availability**	**Researchers**	**Research integrity**
Shape	**Novel methods and models** as research software	**Best practice software engineering**	**Nascent research software engineers** (possibly with Research Software Engineers)	**Research excellence and impact**
Sustain	**Accepted and broadly used methods and models** as research software	**Maintenance**	**Research software engineers**	**Stable research infrastructure**

These three layers form the framework for this agenda. The labels *See, Shape,* and *Sustain* are used throughout the remainder of this agenda as a shorthand for these distinct clusters of characteristics.

See, Shape and *Sustain*

This agenda revolves around three layers of distinct action to *See*, *Shape* and *Sustain* research software. In this section the following are considered for each layer:

- the change sought
- the impact of getting this change right
- what activities both national and international have addressed or are addressing this change
- who has done or is doing consequential work in the area

This will allow us to identify areas for future action, which will be the focus of the section titled "Bringing About Change."

See Research Software

This layer is concerned with making visible the software already being created and used by researchers in the course of doing a wide variety of research projects. This is a very similar challenge to that of facilitating researchers to publish the data arising from their work. Given this, Australia is in a good position to build on the successes of the last decade that have occurred in the data space. Only by making software seen will the true impact that it has on research become measurable. Visible software is a realisation of the *Findable* and *Accessible* foundational principles of the FAIR principles [8], [10]–[13], and a necessary initial condition for recognition of research software as a first class output of research. This measurable impact flows onward: making software visible makes the use of dependencies visible and provides the means to measure the impact of underpinning software.

To date, making software visible has been approached internationally in several ways. Software citation is one way to increase the visibility of software; see the ongoing implementation work of the Force11 Software Citation Working Group [14] for activity addressing this. Standardised approaches to metadata facilitate software discovery; see the work of the CodeMeta project for example activity addressing this. Specialised software registries facilitate discovery and identification; see the work of the partners to the Force11 Best Practices for Software Registries Task Force [15]. In recent years, Software Heritage has provided archival services, and high granularity identification services in the overlapping domain of open source software (noting that research software isn't necessarily open source).

Right now, most researchers don't see research data processes that have been captured in the form of software (or code) as an output of research, although there is a growing recognition that such software is part of the constellation of inputs needed to address (computational) reproducibility [16]–[19]. Because researchers that program are commonly self-taught, sharing their code is often considered by them to be potentially embarrassing. A significant shift is occurring in research data sharing practices, but despite software being so intimately woven into data processes, a shift in practice around software sharing is lagging behind data. In order for software to be seen, it must be understood by all stakeholders to be part of the record of a researcher's work, and be appropriately captured and made visible.

Researchers are known to be responsive to shifts in policy and incentives. The 2018 edition of the Australian Code for the Responsible Conduct of Research [20] and subsequent associated better practice guides (as of writing) presently make little reference to research software. Australia is a signatory to the OECD Recommendation of the Council concerning Access to Research Data from Public Funding [3]. Originally released in 2006 but amended in 2021, this recommendation now calls for research software (in various forms) arising from public funding to be shared for greater public benefit. *See* is aligned with this component of the OECD recommendation.

The infrastructure needs of *See* are highly aligned with the work arising from Europe examining Scholarly Infrastructures for Research Software [21]. The baseline informatics infrastructure needs for software are similar to those for data.[7] The work of the ARDC predecessor organisation ANDS, the network of institutional data repositories connected to Research Data Australia, and the availability of suitable identifiers and vocabularies services all put Australia in a good position to now make software seen by using existing infrastructure. Internationally, software registries, and especially domain specific software registries have arisen to solve the twin issues of discoverability and identification. The ARDC is an active member in a consortium of software registries arising from the Best Practices for Software Registries Task Force ensuring that the way in which software is made discoverable and identified is consistent with developing norms.

Concurrent with the drafting of this agenda, a set of FAIR principles for Research Software are in development through a joint Research Data Alliance, Force11 and Research Software Alliance working group ("FAIR for Research Software" or FAIR4RS). There are several participants representing Australian needs in this group, including from the ARDC. This activity will help to clarify and drive change in research software practices with respect to making it Findable and Accessible (of particular interest to *See*), but also around Interoperability and Reuse (of particular interest to *Shape* below).

Shape Research Software

This layer is concerned with shaping existing research software for broadest use. This requires identifying and understanding new audiences for novel methods and models, and then employing software engineering techniques to ensure that new software intended for use by others is fit for purpose, well-documented, maintainable, robust and efficient in its utilisation of specialist resources (where applicable). This is a necessary second step in recognition of research software as a first class output of research: first class outputs must be more than visible, they must be easy to adopt and adapt and for this they must be well shaped. Getting this right will ensure that new research software is used as broadly as possible, maximising benefit.

Shape is concerned with forming a bridge between researcher practice and software engineering. Broadly speaking there are presently two models that seek to address this. The first is to upskill researchers, particularly the aforementioned **nascent research software engineers** that are self taught, so that they can apply more advanced techniques. The second is to make software engineering expertise available to research projects. This agenda suggests that both models are applicable in a highly heterogeneous research environment. Getting this right means a new generation of highly skilled research software engineers, and researchers who are supported to turn expert knowledge into an actionable form.

[7] While similar, Data and Software needs are not identical. The objects themselves are quite different [22], and the initial settings for change in Software practices are radically different [10], [11].

In the case of upskilling researchers with the basics of programming, there is considerable variation in how this is presently realised, and many instances where it is not being addressed in any systematic way. There are many options for the initial steps, from self-paced MOOCs to formal courses. In some undergraduate programs (outside computer science and information technology), basic programming skills are taught often in applied, statistical or theoretical domains, or at the postgraduate level across a broader range of domains. In some institutions, higher degree research students have access to basic training to jump-start what then becomes self-taught programming skills. There are many self paced options, but these are not directly targeted at researchers. A clear leader in the space of initial skills for researchers is the global Carpentries organisation. Training courses developed under this banner are currently delivered in a number of ways across Australia. Events like regional ResBaz or domain-oriented intensive 'summer schools' or workshops[8] have been quite successful in transferring these skills. The Carpentries model and materials are used by a number of institutions or training providers in Australia. These skills are more rudimentary than what is seeking to be addressed by *Shape*, but it must be noted that building on such a variety of pathways into entry level skills is a challenge for addressing the transfer of mid-tier and advanced software engineering skills.

For mid-tier software engineering concepts, it is currently difficult for a researcher to identify what areas of guidance they need, let alone source that guidance. In many cases the guidance exists in some form, but there are three barriers to adopting this guidance: (1) researchers might not know what specific guidance they need, (2) the guidance might be framed in a way (e.g., for industry applications) that the researcher struggles to translate to a research context and (3) the guidance might be spread over a number of providers and thus difficult to locate. In addition to this the assumed knowledge component for this guidance is very hard to signpost and scaffold. Software engineering has many branches of knowledge.

Many Australian research institutions have centralised units or host community events like Hacky Hours (or similar) to bridge these gaps in the delivery of guidance. At a national level, astronomy researchers have access to the Astronomy Data and Compute Services (ADACS) for domain aware assistance from research software engineers. Similarly, NCI and Pawsey provide access to guidance to ensure that code run on their supercomputing facilities is optimal so as to better utilise this limited resource. The CodeRefinery (an initiative of NEIC) has done an excellent job in producing and delivering "mid-tier" training materials that target researchers and research support staff. Many RSE associations around the world run events to raise awareness of relevant skills. The Software Sustainability Institute in the UK is the clear leader in this space, and has been running training and awareness raising events for a decade.

Finding new audiences for new software is a significant challenge for research software. We are seeking exemplar software projects or programs that have achieved this. Fresh ideas on how to approach this problem and partners in action are particularly welcome.

Sustain Research Software

Once abandoned by its creators, it is generally agreed that any piece of software is likely to continue to work for much less than 10 years [23]. This is usually shorter than the lifecycle for physical research infrastructure. *Sustain* is concerned with the maintenance and sustainability of existing research software. Maintenance and

[8] Examples include the Winter School in Mathematical and Computational Biology, Digital Humanities Downunder (DH Downunder), the Complex Human Data Summer School, and the Harley Wood School of Astronomy. These are all discipline or domain focussed events, and as such don't primarily focus on software engineering skills.

sustainability is certainly enabled by well-engineered software, but the primary concern here is with who will maintain the software, and how that activity can be sustained for the period during which a given piece of software is considered useful. This is the necessary final step for recognition of research software as a first class output of research. When useful software ceases to work, then the whole endeavor fails. Solving this problem leads to robust outputs, careers and an overall healthy ecosystem.

Whether many volunteer driven efforts to sustain open source software survive is often highly dependent on the number of single points of failure within the overall pool of people involved.[9] These projects often survive through the efforts of a surprisingly small number of individuals. Addressing the needs of research software maintainers is one of the concerns of *Sustain*.

Existing solutions to sustain a research software project include commercialisation, open source, or institutional sponsorship. It is quite common for ongoing research software projects to subsist on a string of research funding, going into hibernation some years, and branching off in odd directions in other years as dictated by a subsequent funded project's concerns. This is an unsurprising outcome, given that funding is usually intended for novel developments, rather than for maintenance. But the outcome is unstable infrastructure upon which research is dependent.

A shift in how research software is maintained and sustained will require careful evidence gathering and advocacy. The Research Software Alliance (ReSA) and its members, and many Research Software Engineering Associations have been working to gather the evidence, including our own local RSE association (RSE-AUNZ). The Society of Research Software Engineering (and previously in the guise of the UK RSE Association) and the UK Software Sustainability Institute have been the drivers behind a decade-long push to recognise research software engineering as a critical role in the research software space. Locally, the Federation for the Advancement of Victorian eResearch (FAVeR) conducted a project in 2018 [24] looking at the professionalisation of the eResearch workforce, which overlaps with the professionalisation of research software engineering.

Platforms are a solution of sorts, except they tend to focus on hosting or easing access to research software (as defined here) rather than acting as the primary maintainers of the tools deployed within them. This doesn't actually sustain research software – it shifts work into a different kind of maintenance. It also implies tools support at a certain scale. Any consideration of research software sustainability must also consider how soft-infrastructure interacts with software operating at different scales.

Some facilities, and especially NCRIS facilities, formally host informatics fellows who sometimes perform RSE functions. In many universities research software engineering is embedded within labs or research teams (usually not expressed as research software engineering though), or there may be a centralised, faculty- or centre-based service available to researchers under a variety of service models (from merit based to fee-for-service). In some states or nationally, a similar service is provided across member institutions. ADACS is a national domain based service that operates on a merit based scheme for Astronomy researchers. This model may well work in other domains. Publicly funded research agencies are a clear leader in sustaining research software development. CSIRO, Geoscience Australia, the Bureau of Meteorology and others all identify and support teams working on the development of research software, and also work to translate that software for broader public benefit on an ongoing basis.

[9] This is sometimes sardonically known as the "bus factor" for the project - the number of people that could get hit by a bus and yet the project would survive.

Bringing About Change

In the previous section, a framework was given for structuring the agenda around layers comprising distinct clusters of outputs, needs, authors and drivers. Existing activities attempting to address those needs were given alongside a description of the status quo. In this section, a structured set of areas for action are introduced. These are applied across all three layers of the framework to help identify priorities within each layer.

In many cases, the national gaps identified here have been addressed in some way by other countries. The work of several groups has provided concrete examples of successful activities to make change. This includes the activities of the UK Software Sustainability Institute (SSI) and the Society of Research Software Engineering, the eScience Centre in the Netherlands, the planning for a US Research Software Sustainability Institute (URSSI), the Nordic e-Infrastructure Collaboration (NEIC), including the work on the CodeRefinery, and the Carpentries community of instructors, trainers, and courseware authors around the world, and of the various governance bodies in the Carpentries. The activities conducted by these organisations have heavily informed this agenda, and provided proof that a national approach is possible.

Inspired by the Centre for Open Science strategy for culture change [25], each layer of the framework (*See, Shape* and *Sustain*) is further broken into four areas:

- Building, capturing or enhancing the **infrastructure** for research software to **make change possible**
- The development and subsequent socialisation of materials to support the layer, including **guidance** materials, training materials, reports and evidence gathering, presented as events, webinars, workshops, and training. This will **make adopting changes easier**
- The facilitation of national level **community** building to **make the change normative**
- **Advocacy** for change in policy and strategy will lead to change in incentives and requirements. This includes funder and research institution policy and strategic vision, capturing the ways in which research software is required or incentivised to support research software as an output of research. This will initially **create an environment conducive to change** and eventually **make the change codified**

This provides a useful diagnostic for identifying preconditions to activity at any level. Before attempting to make a change codified, it is useful to consider if it reflects normal practice, if it is easy to do, and certainly it should at least be possible.

Is it possible? > Is it easy? > Is it normative? > Is it codified?

Considering these questions allows us to consider if any infrastructure, guidance, community or advocacy work needs to be initiated prior to or in parallel with any other work addressing identified needs.

Infrastructure

Infrastructure is the baseline enabler of change. For this agenda, three kinds of infrastructure are under consideration. The first is technical infrastructure, the second is research software itself as a form of infrastructure, and the third is the human capital that develops and maintains enduring software infrastructure.

See Infrastructure includes informatics services (identifiers, registries, repositories), workbenches for software development (e.g., computational notebook infrastructure[10]), end-to-end "reproducibility infrastructure" or "reproducibility as a service" (e.g., BinderHub, containerisation services), and software preservation infrastructure. It potentially includes infrastructure to mine and link dependencies metadata in software to track impact, and automating the capture of software actors in provenance metadata against data outputs of platforms.

A necessary first action for this priority is to identify the gaps in this infrastructure that need to be filled. What infrastructure is missing?

ACTION 1: Ensure that research software publication, citation and other informatics infrastructure is fit-for-purpose, and is ready to meet newly emerging practices

What early interventions set up new research software tools for success? How can we ensure broadest meaningful uptake?

Shape Infrastructure refers to the development and tracking of emerging software assets and broadening the useability and user base of that software. This should be done in a way that assists tracking the impact of what would be exemplar software assets from multiple perspectives (researcher, funder, maintainer). In many ways this is about forming a better bridge from novel software to maintained and sustained software.

ACTION 2: Ensure that new software assets are created as infrastructure built for easiest and broadest meaningful reuse from inception

Sustain Infrastructure refers to the soft-infrastructure needed to develop and maintain research software in the mid- to long-term. Enduring software infrastructure simply cannot exist without the workforce to maintain it. The development of models of soft-infrastructure for the maintenance of software assets is needed as a first step to addressing Sustain, with reference to existing national and international efforts in this space.

How is RSE capability distributed nationally across domains and geographically? What do teams look like?

ACTION 3: Ensure enduring research software infrastructure is available to researchers through a skilled, diverse, well distributed and sustainable workforce of research software engineers

Guidance

Guidance is used here as a coverall term for guidance, training, materials development and evidence gathering as well as various modes of delivery. In many cases the guidance already exists and needs to be assembled and curated, in others it will need to be developed. Identifying gaps, monitoring new developments, and tying together disparate resources will be necessary across all areas here.

Suggestions for refinements, or additional areas of interest for guidance on making research software more visible and fit-for-purpose are welcome

See Guidance is best practice for the application of informatics techniques or concerns to research software. This includes the many aspects of making research software FAIR, software citation, registries, and software identifiers. It also extends to practices that go

[10] AARNET provides a national level service addressing this, the SWAN notebook environment

beyond software visibility including curating for reproducibility, awareness and appropriate use of reproducibility infrastructure, the role of workflow systems in research, software preservation and platforms that include software tools or provide a means for authoring software. For data providers, owners and consumers it includes the capture and handling of software actions in provenance metadata against data to expose the role of software in data generation, handling, and analysis.

ACTION 4: Connect researchers, through support professionals, to guidance on informatics infrastructure and emerging best practices

Shape Guidance is best practice in software engineering in a research context. The goal is to bridge research and software engineering practices, and to bridge research communities with common software needs. For software engineering skills, there are four high-level areas of interest:

- "Encapsulation techniques" includes understanding libraries and modules, submissions to package managers, interfaces (APIs but also GUIs), containerisation and any other technology or technique that wraps the core software up in a way that it can be more easily reused
- "Abstraction" includes understanding modularity, algorithms and data types and any other technology or technique to abstract away from low level concerns in order to smooth maintenance and ensure maximum reusability of the software
- "Performance" includes parallelization tools and techniques, code optimisation, low level programming languages, programming for specialist hardware, all to ensure that hardware resources are efficiently utilised
- "Integrity" includes testing, robustness, reliability, version control and maintainability

ACTION 5: Connect researchers and research software engineers to guidance to develop research software infrastructure for easiest and broadest reuse

What evidence will help managers make the case for sustained employment of RSEs? How and where should this capability be distributed?

Sustain Guidance is documenting the case for maintaining and sustaining research software, and the people that make it happen, as well as techniques to enable this. This would include a careful consideration of career paths, models of sustainability, guidance on complex ongoing issues for software projects, such as IP issues or software governance. These primarily relate to sociological rather than technological concerns (which are instead considered under *Shape*). It also includes gathering evidence of practices, careers and impact through surveys, and exemplar data on the impact of well supported software assets.

ACTION 6: Connect research software engineers and their employers to guidance for career and professional concerns in research software engineering

Community

The formation of communities leads to shared behaviours and development and utilisation of shared assets. As an agenda addressing national concerns, national level communities are the primary focus, but these

communities are always in a relationship to local, regional and international efforts, being both informed by and feeding into activities at these other levels.

See Community is a national community or communities around the informatics-focussed support. This includes software citation, curation, provenance, management plans and planning, and preservation concerns. Note that a software citation, curation and management community of practice has been facilitated by ANDS in the past, and is continuing with limited support from the ARDC. There are also domain- or discipline-oriented groups interested in specific areas such as software citation or computational reproducibility, which are important for communicating emerging standards, and developing specific guidance within a domain or discipline.

Would you or your staff be interested in participating in a community focussed on the informatics needs of research software?

ACTION 7: Build and sustain a national community of practitioners interested in informatics infrastructure and in the behaviours that inform it

Do you lead or would you be interested in leading communities or activities that bridge research and software engineering skills and techniques?

Shape Communities are local, regional, national and international efforts to bridge software engineering and research practices. This could be in several formats including (inter)national summer schools[11] with a domain focus or events with a broader technological skills and career stage focus[12], chapters of language oriented groups[13], institutional support groups[14] or groups oriented towards organisational units within an institution (such as faculty, division, school or research centre groups). There is potential here for the bridge to extend to Industry as well.

ACTION 8: Build and sustain local, regional and national communities of practitioners who bridge the gap between research and software engineering practices

Sustain Community is the national community around Research Software Engineering as a career and role, embedded and supported by the research sector. There is already an existing RSE Association covering Australia and New Zealand (RSE-AUNZ) that is concerned with these aims[15] (amongst others). This association is also a member of the International Council of RSE Associations, which coordinates activities with other RSE associations covering the same interests. The international Research Software Alliance also plays a coordinating and evidence gathering role in this space. All stakeholders identified in this agenda have a part to play in supporting the development, growth and sustainability of this trans-national association.

ACTION 9: Build and sustain a national community of professional research software engineers

[11] For instance, the Digital Humanities Downunder, or the Mathematical and Computational Biology Winter School.
[12] For instance, the various ResBaz events run annually around the nation.
[13] For instance, R-Ladies with chapters around the world
[14] For instance HackyHour events
[15] RSE-AUNZ already gather a critical piece of evidence on a 2-yearly basis in the form of an international survey on RSE recognition and value [26].

Advocacy

Advocacy is used here as an umbrella term for the authoring, implementation and advocacy for policy and strategy, depending on the stakeholder.

See Advocacy is for policy and strategy change with funders, publishers and research institutions, with a focus on capturing (software-based) research data processes as a component of research integrity. This is commonly referred to as code availability. A critical shift to enable change is the broadening of scope in policy to include software in its various forms as an output of research. This is inline with the 2021 revisions to the OECD recommendation on access to research data from public funding [3], to which Australia is a signatory.

What opportunities do you see at your institution for gathering evidence to support these areas of advocacy?

ACTION 10: Create the policy and incentives environment to encourage code availability

Shape Advocacy is policy and strategy change with funders, publishers and research domain groups and research institutions to recognise the value and impact of novel research software implemented in a form intended for others to reuse as a significant scholarly work in its own right.

ACTION 11: Create the policy and incentives environment that recognises research software as a first class output of research

Sustain Advocacy is policy and strategy change with research institutions and related employers to consider sustainable and impactful models of research software ownership and maintenance. Funders may have a role to play in this work.

ACTION 12: Create the policy and incentives environment that supports the development and maintenance of critical research software infrastructure

Fresh ideas and approaches to supporting software maintenance are particularly welcome. What would help you make the case?

Summary

The three layers of the agenda framework, *See, Shape* and *Sustain* are further broken down to identify concerns about necessary infrastructure, guidance, community and advocacy. The following areas are identified to address these concerns:

	Infrastructure	***Guidance***	***Community***	***Advocacy***
See	Research Software Informatics Infrastructure	Software Informatics Guidance	Software Informatics Support Community	Advocacy for Code availability
Shape	Research Software assets as Infrastructure	Software engineering Guidance	Communities that bridge research and software engineering practices	Advocacy for credit for new software
Sustain	Soft-infrastructure for Research Software	Professional RSE Guidance	Career RSEs Community (RSE-AUNZ)	Advocacy for Sustainable Research Software

Actions addressing these are summarised in the next section.

A Research Software Agenda for Australia

The primary aim of this agenda is recognition of research software as a first class output of research. To achieve this, the following 12 actions are proposed. Actions are grouped under three high level actions (to "See, Shape and Sustain Research Software) addressing infrastructure, guidance, community and advocacy concerns.

See Research Software:

1. Ensure that research software publication, citation and other informatics infrastructure is fit-for-purpose, and is ready to meet newly emerging practices
2. Connect researchers, through support professionals, to guidance on informatics infrastructure and emerging best practices
3. Build and sustain a national community of practitioners interested in informatics infrastructure and in the behaviours that inform it
4. Create the policy and incentives environment to encourage code availability

Shape Research Software:

5. Ensure that new software assets are created as infrastructure built for easiest and broadest meaningful reuse from inception
6. Connect researchers and research software engineers to guidance to develop research software infrastructure for easiest and broadest reuse
7. Build and sustain local, regional and national communities of practitioners who bridge the gap between research and software engineering practices
8. Create the policy and incentives environment that recognises research software as a first class output of research

Sustain Research Software:

9. Ensure enduring research software infrastructure is available to researchers through a skilled, diverse, well distributed and sustainable workforce of research software engineers
10. Connect research software engineers and their employers to guidance for career and professional concerns in research software engineering
11. Build and sustain a national community of professional research software engineers
12. Create the policy and incentives environment that supports the development and maintenance of critical research software infrastructure

Stakeholders

Delivering this agenda will require the input of and action from a large range of stakeholders. In this section, these stakeholders are elaborated upon as well as their suggested interests across the framework. Any given individual may identify with more than one stakeholder type below. In this case the consideration of relative interests should be the sum of the areas of interest across all roles.

A summary at the end of this section maps all stakeholders and their suggested interest against the specific agenda actions.

Authors of Software

Anyone writing research software is clearly a stakeholder in this agenda. *See*, *Shape* and *Sustain* are associated with three different types of author. In this section the author types defined under "The Challenge" are revisited as stakeholders, to consider their position across all layers of the agenda.

Researchers

As this agenda is concerned with the outputs of research, any change in this area cannot be achieved without the support of researchers. However, researchers are a large and heterogeneous group. Not all researchers directly produce or are involved in projects that result in software outputs. But many researchers do interact with software or infrastructure that utilises research software.

For the large number of researchers that produce small scale, often single use code or software for any step in a specific research project (for instance towards a publication), these researchers will be impacted by any improvements in *See*. Some researchers with a growing interest in software engineering skills may be interested in *Shape*.

Nascent Research Software Engineers

Nascent Research Software Engineers are those researchers who are likely self taught programmers producing software for the benefit of others, especially for those outside their project team.

In Australia, researchers may refer to themselves as research software engineers, but it is very likely that their job title may be quite different. For individuals who either primarily produce software (or want to), it can be a delicate issue as to whether they identify as a (research) software engineer or as a researcher, and further that some employers preferentially treat this either as an academic or professional role, which can have dramatic effects on career paths and options.

Nascent Research Software Engineers are often authors of research software in the form of novel methods and models intended for use by others. They need access to guidance and support for the appropriate application of

software engineering techniques. They may benefit from access to a broader network of like-minded peers. They would benefit greatly from advocacy for recognition of research software as a first class output of research. As such, outcomes from and participation in *Shape* are highly relevant to these stakeholders.

Research Software Engineers

This title is most commonly used in the UK to refer to the combination of professional software engineering expertise and an intimate knowledge of research. As an indicator of the maturity and recognition of this role, in 2019 the UKE RSE Association was transformed into the Society for Research Software Engineering, a charitable incorporated organisation. The UK RSE Association, and now the Society for Research Software Engineering has inspired the formation of a large number of local, regional and national level RSE Associations around the world, including a combined Australian and New Zealand Association.

Considered here are those researchers or professional staff that have a career in research software engineering. They may be embedded in projects, labs, schools, faculties or in centralised support services. To differentiate them from nascent research software engineers, they may or may not be self-taught, but are reasonably confident in their software engineering skills, and are mostly concerned with career pathways, guidance on professional matters relating to projects, might benefit from connecting to a community of like-minded peers and will likely benefit from advocacy for a considered approach to software maintenance. As such, outcomes and activities arising from *Sustain* are highly relevant to them. They may have knowledge and skills to apply towards *Shape* activities, and might either want to be aware of or consulted on elements of *See*.

Platforms Developers

Platforms are environments that provide ready access to specialist software, or additional resources such as data assets, (specialised) compute resources, or network capability.

Research software development, and research platforms development are considered largely distinct but occasionally overlapping endeavours in this agenda. Platforms often incorporate research software, but it is the locus of development that is being differentiated. Platforms developers can be Research Software Engineers and vice versa, but here the stakeholder is characterised as one not working directly on research software outputs.

The clear role that platform developers have in this agenda is within *See* to work towards the visibility of the software, and especially research software components that are deployed in the services they provide. By incorporating extensive provenance information that captures the software actors contributing to the final form of datasets provided by their platforms, platform developers have a role in making research software *seen*. Orchestration, containerisation and web API or service development are considered some of the relevant topics for *Shape Guidance* in “encapsulation” and “abstraction”. Platform developers tend to be highly skilled in these areas.

Professional Support Staff

Professional support staff are a critical stakeholder in this agenda. Support staff are involved in infrastructure development (including research software as infrastructure), and the development and delivery of guidance and policy through training, consultancy, community facilitation and other related roles.

Informatics Support Staff

Professional research support staff providing support services for software curation, citation practices, provenance and software management plans are here grouped under the single label of "Informatics Support Staff". These research support staff communicate relevant guidance, policy and other materials. They may be housed in the library services at a given institution, or possibly as part of another form of research services. Or they may be closer to the research team, but not as a software engineer. For instance, they may be data managers, data stewards, or employed as research assistants.

Outcomes from and participation in *See* is highly relevant to Informatics support staff. Awareness of outcomes in *Shape* may also be relevant. The action for *See Community* is primarily a community for Informatics Support Staff.

Research Software Support Staff

Those professional research support staff that assist with writing code (but are not the "authors" of that code), or are in training or similar roles are in a position to communicate or demonstrate good software practices as labelled "Research Software Support Staff" here. These staff help bridge researcher practice with software engineering practices, and as such *Shape* is particularly relevant to their work.

This characterisation assumes that they are not also working as research software engineers, but this is often the case. See that section for additional detail.

Research Institutions

In this agenda, research institutions refers to a range of institutions where research is conducted. In Australia, this could refer to research intensive universities, (medical) research institutes, or to publicly funded research agencies (PFRA) like the CSIRO, Geoscience Australia, or the Bureau of Meteorology. It can also refer to the research that is conducted in industry, for those industries that are sympathetic to the aims of this agenda. Note that the roles described here more readily map onto Universities, Research Institutes or PFRAs.

As employers of those that write or support the writing of research software, and as implementers of policy concerning software outputs, research institutions determine:

- how they support research software as an output
- how skills and capabilities are distributed, supported and nurtured
- how capabilities are crystalised in roles and structures in the organisation
- what strategic importance is given to research software as a research priority

These are all critical factors for the success of the agenda, with different actions relevant across all layers of the agenda. Responsibilities for these institutions are mostly across *Infrastructure* and *Advocacy* rather than confined to a specific layer of the agenda.

Research Software Engineering Services

"Research Software Engineering Services" refers to the units that employ professional research software engineers and platform developers for the benefit of researchers or research projects. This capability has several manifestations across the sector, from centralised units in research institutions, to units much closer to or directly related to a research initiative such as a funded, time-limited project, or an ongoing research enterprise. In some states, this is available as a service for multiple research institutions within that state (i.e., Intersect and QCIF). Common to all though, is that these service units are an institutional or organisational response that serves as an alternative model of support. These services sidestep the need to teach software engineering skills to researchers.

In some locations, research software engineering services come under the banner of eResearch services, but while eResearch services may house this capability, this capability is not viewed throughout the sector as synonymous with eResearch services. In addition to this, the use of the term eResearch is not generally recognised outside of Australia and New Zealand. However, for those that are put forward as eResearch services, the Association of eResearch Organisations (AeRO) might be able to play a coordinating role in this space.

The variety of service provision models is a strength in the overall ecosystem, and all these units or organisations have a part to play in:

- exploring models of support
- considering their position in the national ecosystem
- exploring greater optimisation of and coordination around the production of research software as an output of research

The great variety in delivery of services means that individuals in these support units (here considered a type of research software engineer) might be operating across all three layers of the agenda, but the organisational unit as a stakeholder has a clear role in *Shape* and *Sustain* activities.

Publishers

Journal publishers can be enablers of change in scholarly practice through the policies that they adopt and require. Within the framework of this agenda there is a difference between the kinds of software outputs in *See*, focussing on research data processes, and those in *Shape*, focussing on broadcasting novel methods and models as software for use by other researchers.

From a *See* perspective publishers have a role in broadening the scope in policy of expected materials that might be supplementary to a publication, and the expectations around citation of software components. That is, it is not just data that should accompany a published work, but also the software or record of software use that,

when combined with data, lead to the analyses represented in the publication. While not all publishers focus on research that might include this kind of detail, the applicability to publishers is broad. In addition to this, Publishers are often providers of infrastructure relevant to *See.*

With respect to *Shape*, the publisher's interest is narrower and more editorial. Journals that focus either exclusively on research software (for instance JOSS, JORS or Software Impact) or domain specific journals which accept papers broadcasting the importance, relevance and availability of research software in the form of novel methods and models have a role in directly recognising research software as a first class output of research. That is, the kinds of research software relevant to *Shape* should be considered a presentation of substantial scholarly work, and should be valued as such.

Funders

In setting the parameters for research funding, funder involvement is aligned with proposed *Advocacy* in this agenda.

From a research integrity perspective, policy and incentives to provide software code as part of the record of research data processes is the *Advocacy* focus for *See*. Recognising the benefits and impact of research software capturing novel methods and models is the *Advocacy* focus of *Shape*. Funders may have a role to play in how to maintain and sustain the research software that captures accepted methods and models for use by other researchers. This is the *Advocacy* focus of *Sustain*.

Data Generating Facilities

There is a role for data generating infrastructure providers, especially NCRIS facilities, to consider making more visible the role of software in the generation and processing of data. This can occur in the form of provenance metadata as called for in the FAIR data principles [13], or through making available the standard or bespoke workflows, scripts and other processing pipelines used in their facilities for research integrity purposes. This makes *See* especially relevant to these providers.

Most modern research instrumentation is enabled and heavily augmented by software, or access to the equipment is mediated through specialist roles that blend informatics, domain expertise and software engineering skills. This is a highly specialised form of research software engineer, and so data generating facilities have a strategic decision to make about the ongoing support and viability of these kinds of roles. As such, *Sustain* is highly relevant to these providers.

These providers are also sometimes owners of platforms to house data and provide analysis tools. The stakeholder interest for this is considered under Platforms Developers.

National Digital Infrastructure Providers

There is a role for National Digital Infrastructure Providers[16] (NCI, Pawsey, AARNET, AAF, and ARDC) in achieving the aims of this agenda. In particular, the proposed activities of this agenda within *See* explore the question of what (additional) informatics services might be needed for software identifiers, metadata and storage, and what infrastructure might help to address computational reproducibility. These providers can help in identifying and filling any gaps in the overall ecosystem.

HPC providers also provide specialist support to develop software that optimally utilises specialist storage and compute infrastructure. This is relevant to *Shape Guidance*, and the employees doing this may be characterised as Research Software Support Staff or Research Software Engineers.

The Australian Research Data Commons (ARDC)

The current work of the ARDC is in line with the areas considered above under National Digital Infrastructure Providers. The ARDC provides informatics infrastructure suitable for research software. We have provided guidance in informatics related issues for software, we have previously facilitated a Software Citation, Curation and Management Interest Group, and we have advocated for software in the context of data. These are all areas of work with *See*.

The release of this agenda is an expression of an interest in expanding into areas covered by *Shape* and *Sustain*.

Other National Stakeholders

RSE-AUNZ is the Australia-New Zealand Association of Research Software Engineering. The organisation itself (rather than its members) has a clear role in aspects of *Advocacy* and *Community*, particularly for *Sustain*, but also represents members with a body of knowledge relevant for *See* and *Shape.*

AOASG is the Australian Open Access Strategy Group. There is a clear role for this group in *Advocacy* for open source Research software outputs, of kinds relevant for both *See* and *Shape*: research data processes and novel methods and models in the form of research software.

AeRO is the industry association for Australian eResearch Organisations. There is a potential coordinating role for AeRO with eResearch units and their common activities as they relate to this agenda. See Research Software Engineering Support Service Units for these areas.

[16] Referred to as the Digital Data and eResearch Platforms (DDeRP) in the 2016 NCRIS roadmap [27]

Summary

All Stakeholders are listed below against a proposed relative order of interest in specific actions (as indicated by shading). Combined, it is clear that this agenda cannot be delivered by one organisation alone. Many stakeholders must come together for the success of this agenda.

	Infrastructure	*Guidance*	*Community*	*Advocacy*
See	(Action 1) Research Institutions, National Digital Infra., Platforms Devs, Publishers, Researchers, Nascent RSEs, Informatics Support, RSE Services, Data Generators, RSEs, Research Software Support	(Action 2) National Digital Infra., Informatics Support, Research Institutions, Researchers, Nascent RSEs, Platforms Devs, Research Software Support, RSE Services, Publishers, RSE-AUNZ, AeRO, RSEs, Data Generators	(Action 3) Informatics Support, Researchers, Nascent RSEs, RSEs, Platforms Devs, Research Institutions, Data Generators, National Digital Infra., Research Software Support, RSE Services, Publishers	(Action 4) Research Institutions, Publishers, Funders, Data Generators, AOASG, Researchers, Nascent RSEs, National Digital Infra., RSEs, Platforms Devs, Informatics Support, Research Software Support, RSE Services
Shape	(Action 5) Nascent RSEs, RSEs, Research Institutions, RSE Services, Research Software Support, AeRO, Researchers, Platforms Devs	(Action 6) Research Software Support, RSE Service, Research Institutions, Nascent RSEs, RSEs, Platforms Devs, Researchers, Informatics Support	(Action 7) Nascent RSEs, Research Software Support, RSEs, Research Institutions, RSE Services, Data Generators, Researchers, Informatics Support, Publishers	(Action 8) Research Institutions, Publishers, Funders, RSE Services, AOASG, Nascent RSEs, RSEs, Research Software Support, Data Generators, RSE-AUNZ, Researchers, Informatics Support
Sustain	(Action 9) RSEs, Research Institutions, RSE Services, Data Generators, AeRO, Nascent RSEs, Research Software Support	(Action 10) RSEs, Data Generators, Research Institutions, RSE Services, Nascent RSEs	(Action 11) RSEs, RSE-AUNZ, Research Institutions, Data Generators, Nascent RSEs, RSE Services	(Action 12) RSEs, Research Institutions, Data Generators, RSE Services, Nascent RSEs, Funders

Next Steps

This Research Software agenda will proceed through a number of phases.

Step	Description	Indicative Timing
Validation	This draft agenda is now available for community comment via the ARDC website. The Software Program Manager is also happy to discuss the agenda directly.	June 2021
Updating	The input received during the validation phase will be incorporated into a final version 1.0 of an agenda for research software.	July 2021
Planning	Once there is a final version 1.0 of the agenda, a detailed implementation plan will be developed for FY 2021/22, with an indicative plan for FY 2022/23 (the end of current ARDC funding). Identified stakeholders in the delivery of this program will continue to be consulted during this time. A high-level summary of this implementation plan will be made available to the community once resourcing has been secured.	August 2021
Implementation	This will commence with existing resources as soon as version 1.0 of the agenda has been finalised. Implementation will accelerate once additional resources are secured.	After August 2021

References

[1] S. Hettrick *et al.*, 'UK Research Software Survey 2014'. Zenodo, Dec. 04, 2014. doi: 10.5281/zenodo.14809.

[2] M. Bello and F. Galindo-Rueda, 'Charting the digital transformation of science: Findings from the 2018 OECD International Survey of Scientific Authors (ISSA2)', OECD Publishing, Paris, 2020/06, May 2020. doi: 10.1787/c7bdaa03-en.

[3] OECD, 'OECD Recommendation of the Council concerning Access to Research Data from Public Funding'. https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0347 (accessed Mar. 22, 2021).

[4] J. M. Perkel, 'Ten computer codes that transformed science', *Nature*, vol. 589, no. 7842, pp. 344–348, Jan. 2021, doi: 10.1038/d41586-021-00075-2.

[5] R. Knowles, B. A. Mateen, and Y. Yehudi, 'We need to talk about the lack of investment in digital research infrastructure', *Nature Computational Science*, vol. 1, no. 3, pp. 169–171, Mar. 2021, doi: 10.1038/s43588-021-00048-5.

[6] U. Nangia and D. S. Katz, 'Understanding Software in Research: Initial Results from Examining Nature and a Call for Collaboration', in *2017 IEEE 13th International Conference on e-Science (e-Science)*, Oct. 2017, pp. 486–487. doi: 10/ggfkvb.

[7] M. Clément-Fontaine, R. Di Cosmo, B. Guerry, P. MOREAU, and F. Pellegrini, 'Encouraging a wider usage of software derived from research', Committee for Open Science's Free Software and Open Source Project Group, Research Report, Nov. 2019. Accessed: Mar. 10, 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02545142

[8] W. Hasselbring, L. Carr, S. Hettrick, H. Packer, and T. Tiropanis, 'From FAIR research data toward FAIR and open research software', *it - Information Technology*, vol. 62, no. 1, pp. 39–47, Feb. 2020, doi: 10.1515/itit-2019-0040.

[9] K. Hinsen, 'Dealing With Software Collapse', *Comput. Sci. Eng.*, vol. 21, no. 3, pp. 104–108, May 2019, doi: 10.1109/MCSE.2019.2900945.

[10] A.-L. Lamprecht *et al.*, 'Towards FAIR principles for research software', *Data Science*, vol. 3, no. 1, pp. 37–59, Jan. 2020, doi: 10.3233/DS-190026.

[11] D. S. Katz *et al.*, *A Fresh Look at FAIR for Research Software*. 2021.

[12] M. Gruenpeter *et al.*, 'M2.15 Assessment report on "FAIRness of software"', Oct. 2020, doi: 10.5281/zenodo.4095092.

[13] M. D. Wilkinson *et al.*, 'The FAIR Guiding Principles for scientific data management and stewardship', *Sci Data*, vol. 3, p. 160018, Mar. 2016, doi: 10.1038/sdata.2016.18.

[14] A. M. Smith, D. S. Katz, and K. E. Niemeyer, 'Software citation principles', *PeerJ Comput. Sci.*, vol. 2, p. e86, Sep. 2016, doi: 10.7717/peerj-cs.86.

[15] T. F. on B. P. for S. Registries *et al.*, 'Nine Best Practices for Research Software Registries and Repositories: A Concise Guide', *arXiv:2012.13117 [cs]*, Dec. 2020, Accessed: Mar. 09, 2021. [Online]. Available: http://arxiv.org/abs/2012.13117

[16] V. Stodden, J. Seiler, and Z. Ma, 'An empirical analysis of journal policy effectiveness for computational reproducibility', *Proc Natl Acad Sci USA*, vol. 115, no. 11, pp. 2584–2589, Mar. 2018, doi: 10.1073/pnas.1708290115.

[17] Y. AlNoamany and J. A. Borghi, 'Towards computational reproducibility: researcher perspectives on the use and sharing of software', *PeerJ Comput. Sci.*, vol. 4, p. e163, Sep. 2018, doi: 10.7717/peerj-cs.163.

[18] A. Allen, P. J. Teuben, and P. W. Ryan, 'Schroedinger's Code: A Preliminary Study on Research Source Code Availability and Link Persistence in Astrophysics', *ApJS*, vol. 236, no. 1, p. 10, May 2018, doi: 10.3847/1538-4365/aab764.

[19] A. Culina, I. van den Berg, S. Evans, and A. Sánchez-Tójar, 'Low availability of code in ecology: A call for urgent action', *PLOS Biology*, vol. 18, no. 7, p. e3000763, Jul. 2020, doi: 10.1371/journal.pbio.3000763.
[20] 'Australian Code for the Responsible Conduct of Research, 2018 | NHMRC'. https://www.nhmrc.gov.au/about-us/publications/australian-code-responsible-conduct-research-2018 (accessed Apr. 12, 2021).
[21] European Commission. Directorate General for Research and Innovation., *Scholarly infrastructures for research software: report from the EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS.* LU: Publications Office, 2020. Accessed: Mar. 09, 2021. [Online]. Available: https://data.europa.eu/doi/10.2777/28598
[22] D. S. Katz *et al.*, 'Software vs. data in the context of citation', PeerJ Inc., e2630v1, Dec. 2016. doi: 10.7287/peerj.preprints.2630v1.
[23] 'Challenge to scientists: does your ten-year-old code still run?' https://www.nature.com/articles/d41586-020-02462-7 (accessed Apr. 12, 2021).
[24] N. May, 'Roles for eResearch: Results of a Role Modelling Session', Oct. 2019, doi: 10.6084/m9.figshare.c.4524317.v1.
[25] B. Nosek, 'Strategy for Culture Change', 2019. https://www.cos.io/blog/strategy-for-culture-change (accessed Apr. 16, 2021).
[26] 'Australian results of the 2018 International RSE Survey', *RSE Australia / New Zealand*, Aug. 09, 2019. /survey/2018/2019/08/09/Australian-Results-2018-Survey.html (accessed Nov. 20, 2020).
[27] A. Finkel, *2016 National Research Infrastructure Roadmap*. Government of Australia, 2018. Accessed: Apr. 14, 2021. [Online]. Available: https://apo.org.au/node/171966