e-IRG secretariat
P.O.Box 93575, NL-2509 AN The Hague, The Netherlands
Phone: +31 (0)70 344 0526
secretariat@e-irg.eu

Visiting address
Anna van Saksenlaan 51
NL-2593 HW The Hague
The Netherlands

978-90-817691-4-3

e-IRG

Policy Paper on Scientific Software

Table of Contents

e-IRG Policy Paper on Scientific Software

Report from the e-IRG Task Force on Scientific Software

25th September 2012

1. Introduction and Background

With the successful establishment of a European e-Infrastructure ecosystem, the question of how to maintain and improve the scientific software base has become an urgent issue. This ecosystem has been built through recent leadership projects such as EGI [1] and PRACE [2], combined with significant efforts on national levels. Many applications depend on legacy software that is difficult to maintain and difficult to run efficiently on current and future e-Infrastructures. There is a clear need for a coherent process and major efforts targeted towards enhancing the European software base for efficient use of European e-Infrastructures to increase the scientific output while ensuring the best value for money. The need for actions in this area has also been acknowledged globally and, e.g. in the US, significant funding programs targeting scientific software have been initiated.

The "software crisis" has been a topic of discussion for several years and various aspects have been addressed in several actions. A non-exhaustive list of examples includes:

- The EC funded software projects (such as ScalaLife [3] and MAPPER [4]) that look into specific community codes (e.g. ScalaLife is working on Life Science) or specific cross-cutting issues (e.g. MAPPER is working on multi-scale modelling);

- The International Exascale Software Project (IESP) [5], supported by the EC-funded European Exascale Software Initiative (EESI) [6], is proposing research directions essential for reaching exascale performance;

- The EC funded PRACE-2IP project has a dedicated work package on support for petascale community codes;

- Three exascale software projects (CRESTA [7], DEEP [8], and Mont-Blanc [9]) have recently been funded by the EC;

- The EC recently commissioned a software study by IDC advocating the establishment of European Software Centres of Excellence [10]

- Several European countries have recognized the need to invest in software and have instigated related programs, for instance the Swedish e-Science initiatives, SeRC [11] and eSSENCE [12], and the UK EPSRC [13] HPC Software Development and Software for the Future initiatives and the Software as an Infrastructure strategy [14].

- Several reports have been commissioned by the research community advocating changes in the investment in scientific software. For instance, the recent UK "Strategy for the UK Research Computing Ecosystem" [15] articulated the following recommendation: **"There needs to be long-term funding for ambitious software development projects, including the rewriting of legacy code, particularly targeting community codes, and for their on-going maintenance and support"** and the RCUK review of e-Science [16] led by Dan Atkins recommended **"creating and sustaining long-term centres for software development and support"**.

In view of the upcoming Horizon2020 program there is a clear need to develop a consistent framework and related policies for establishing a European Software Strategy building on the efforts mention above, and other related efforts, and taking a holistic view of the national, European, and international ecosystem. The e-IRG has taken a pro-active role in this process and started a task force to develop this e-IRG position paper laying the ground for a European software strategy.

Members of the e-IRG Task Force on Scientific Software

The task force started working in March 2012 with the following members:

- Neil Chue Hong, EPCC & Software Sustainability Institute
- Sverker Holmgren, UU [Chair]
- Dieter Kranzlmüller, LMU & LRZ
- Stéphane Lanteri, INRIA
- Erwin Laure, KTH
- Guy Lonsdale, Scapos
- Per Öster, CSC

Editorial Support has been provided by Christian Straube (MNM-Team, Munich) as part of e-IRGSP3 [17].

Scope

The scope of the task force is on “scientific software”, that is software that is primarily used in research and development, both within academic and industrial environments. Such software packages normally exist in an environment of a complex software stack comprising:

1. Operating System
2. Compiler
3. Libraries
4. Programming language/paradigm
5. Scientific simulation/analysis software
6. Frameworks (workflow systems, resource access/management frameworks/middleware)
7. Portals, GUIs

Although advances on all of these levels will eventually be needed, we will focus on the scientific simulation/analysis software (item 5 above) with some consideration of the underlying and high level layers. In particular this layer requires an interdisciplinary approach, combining competences in computer science, mathematics and numerical analysis, and the domain science in order to develop successful and efficient simulation/analysis software.

Current Software Maintenance Organization

One can identify different organizational schemes on how the development and maintenance of scientific software is organized. In some domains a set of canonical software packages exist that are widely used within the community. These packages are either supported by, sometimes large, long-lasting academic collaborations, often using an open-source model, or by commercial software organizations. In these domains it is relatively easy to identify the software packages whose improvement will make a significant

contribution to the efficiency of the scientific domain. The development of new packages is relatively rare in these domains, rather existing packages are extended or customized. On the other side of the spectrum are domains where a multitude of different software packages compete, some with a commercial background, the majority being developed in small academic settings, like within PhD programs. In these domains it is much more difficult to identify targets for concerted improvement actions as new software appears on the market very frequently, often at the expense of already existing software that ends up deprecated or abandoned. In addition, the user base of certain packages is much smaller as more packages compete in the market.

2. The Scientific Software Crisis

Since the early days of computing, the term “software crisis” (also often referred to as “software bottleneck”) has been used to denote the situation where the capabilities of hardware advance much faster than the capabilities of software and as a consequence software development lags behind hardware development (the 1968/69 NATO Software Engineering Reports [18]). In the past decades this crisis was less of a concern as much of the performance improvement of hardware was due to the ever increasing clock-speed and software was able to take a “free ride” on this development.

In recent years this hardware trend has slowed down and even inverted as CPUs were hitting the “power wall”. To further increase the hardware performance, chip designers are using different architectural features, particularly multi-core architectures, vector-units, as well as accelerators and many-core architectures (GPUs, Intel’s MIC, etc.), increasing the heterogeneity of hardware. Most scientific software is lagging behind those architectural trends and the **“software crisis”** is again having a very negative impact on application performance. In fact, most applications will experience performance losses when migrating to recent hardware, unless major changes in their software stack are applied [19].

For instance, benchmarking studies performed by the PRACE project [20] have shown that many applications exhibit very different efficiency and scaling behaviour on different architectures. It is thus for most applications not possible to simply migrate them from one architecture to another. The PRACE project also performed a user survey [21] asking, among other things, in which aspects applications would need improvement. The recurrent themes in the answers were the need for new algorithms; communication, memory bandwidth and I/O bottlenecks; and the need to exploit hybrid (MPI plus shared memory) programming.

To exploit modern hardware, advances on all levels of the software stack are required. This ranges from new programming models and languages, compilation and runtime technologies, to new algorithmic approaches. It is commonly accepted that only an interdisciplinary approach, involving hardware experts, numerical/algorithm developers, language/compiler/runtime system designers, and application programmers will be successful to tackle the current software crisis. This interdisciplinary approach is often being referred to as “co-design”.

3. Centres of Excellence for Scientific Software

To overcome the current software crisis we propose the establishment of **Centres of Excellence for Scientific Software (CESS)**. In the remainder of this section we discuss various aspects of such centres, particularly the competences needed, how requirements can be gathered, the role of software industry and e-Infrastructure providers, and finally organizational and networking aspects as well as funding and governance. A set of recommendations are being made and we hope that these recommendations are being followed-up in national and European programs to ensure European researchers from both industry and academia will have best of breed software available and for the continuing leadership of Europe in scientific software.

3.1. Competences needed

The prevalence of modern multicore technologies has made massively parallel computing ubiquitous and offers a huge theoretical potential for compute-intensive tasks. In theory, advances in this technology bring us closer to solving the scientific and technological challenges in modern computing through large-scale simulations. However, parallelism is no longer restricted to well-balanced systems built of homogeneous nodes. In modern computer systems, parallelism spreads over many architecture levels including nodes, processors, cores, threads, registers, SIMD-like and vector units, leading to several different levels of parallelism (from coarse to fine or very fine grain parallelism) that one has to harness in order to maximize computational efficiency and scalability. Moreover, heterogeneity of the memory is growing at the node as well as at the chip level. The resulting non-uniform memory penalty in data accesses is certainly one of the main critical issues for parallel performances and scalability on modern petascale and emerging exascale supercomputers such as those that are or will soon be at the heart of the European e-infrastructure ecosystem. In practice all these heterogeneous characteristics of hardware resources most often keep effective performance far from theoretical peak.

Indeed, most applications and algorithms are not yet ready to utilize the available processing capabilities and developing large-scale scientific computing tools that efficiently exploit this processing power is a very complicated task and will be an even more challenging one with future exascale systems. So a tremendous effort is required to close the gap and the heterogeneity characteristic and hierarchical organization of modern massively parallel computing systems are recognized as central features that impact at all the layers from the hardware to the software with issues related to computer science and numerical mathematics as well. At the current state of the art in technologies and methodologies, a multi-disciplinary approach is required to tackle the obstacles in many-core computing, with contributions from computer science, applied mathematics, high performance computing, and engineering disciplines. Compute and memory intensive applications can only benefit from the full hardware potential if all features on all system levels are taken into account in a global approach.

The CESS will define a collaborative work venue for maintaining and improving the scientific software base for the European e-infrastructure ecosystem. In order to achieve this goal, these scientific software centres will host engineers and researchers with different knowledge and expertise related to high performance scientific computing:

- Computer scientists with expertise in high performance computing, in particular on programming models, languages and environments for massively parallel computing.

- Numerical algorithms specialists who propose algorithms and data structures that contribute to such software in order to take benefit from all the parallelism levels with the main goal of optimal scaling on very large numbers of computing entities.

Numerical mathematicians who are studying mathematical models , numerical schemes and scalable algorithms for solving important scientific and technological problems related to the quality and the security of life in our society. This includes both simulation (including the solution of e.g. deterministic differential equations, stochastic differential equations, algebraic equations, etc.) and data analysis (including statistical methods and models and handling of large data sets). All together, these researchers will form a continuum of expertise on enabling methodologies and technologies required to address the aforementioned critical issues for harnessing the power of modern petascale and emerging exascale supercomputers. More precisely, skills and competences on the following topics are needed:

- General background on parallel computing. This topic covers all the aspects of parallel computing from hardware (MIMD and SIMD parallel architectures, distributed, shared, hybrid distributed-shared memory architectures, interconnection technologies, etc.) to software issues (parallel programming approaches for scientific computing, performance evaluation models, etc.).

- Programming models for petascale and exascale computing. This topic is concerned with the adaptation and harnessing of new massively parallel programming paradigms for numerical computing applications. Ideally, this covers low level to high level knowledge on programming models, languages and environments for massively parallel computing. At the lower level, a knowledge and an extensive practice of widely adopted standards for coarse grain MIMD parallel programming such as MPI [22] (distributed memory programming) and OpenMP [23] (shared memory programming) are mandatory. A knowledge and experience of the recently introduced OpenCL [24] standard for fine grain SIMD programming will definitely be an asset. At the higher level, advanced programming models such as a partitioned global address space (PGAS), Co-Array Fortran (CAF) and Unified Parallel C (UPC), have certainly to be considered. There are also new language proposals like Fortress (SUN), Chapel (Cray), and X-10 (IBM), which promise improved programmer productivity. However, although these languages attempt to address many challenges that are faced in moving to exascale, they have yet to establish themselves in terms of performance in complex applications and a variety of hardware architectures. Ideally, some of these candidates programming strategies should be concurrently considered in order to allow for a transparent and efficient combined exploitation of fine grain and coarse grain parallelisms.

- Tools and environments for debugging and optimizing the performances of parallel applications. Debugging a parallel application is a complicated and time-consuming task when conducted manually. Parallel debugging tools are often available as components of the software stack attached to a particular hardware system. A few generic tools are available as commercial software such as, for instance, DDT (Distributed Debugging Tool) [25] and TotalView [26]. The situation is somewhat more favorable for what concern parallel performances optimization and profiling since a larger number of software are available, some of them as free software, which are thus less specific to specific computer architecture. Examples are TAU (Tuning and Analysis Utilities) from University of Oregon, Vampir developed at the Centre for Applied Mathematics of Research Centre Jülich and the Centre for High Performance Computing of the Technische Universität Dresden, and Paraver developed at the Barcelona Supercomputing centre.

- Numerics oriented libraries and toolkits. The objective will not be about developing new mathematical methods but rather to take up and include modern numerical methods developed elsewhere (mostly in academic institutions) into complex scientific applications. A typical domain of interest to almost all differential equations-based applications or optimization problems is numerical linear algebra. This topic is concerned with core numerical linear algebra kernels adapted to modern high performance computing systems, as well as with scalable algorithms for the solution of the large linear systems of algebraic equations resulting from the numerical treatment of mathematical models underlying the complex scientific applications considered nowadays. This includes the solution of sparse linear systems which is one of the most critical and intensive computational kernels in terms of memory and time requirements, and which is very often at the heart of a numerical simulation tool.

Overall, this points to a requirement to ensure that the next generations of researchers are provided with the knowledge and skills to enable engagement with computer science, numerical algorithms and mathematical modelling specialists. This basic capability can be provided through international initiatives such as Software Carpentry [27] and through the development of appropriate curricula across Europe, which provides all researchers with a foundation in computational science and distributed computing.

3.2. Organization and Requirements

A CESS should have the capability of supporting the whole life cycle of software. This implies that its organizational focus, interfaces with stakeholders, and requirements gathering process must be able to create the mechanisms for transformation of innovative tactical development into strategic development for all sorts of scientific software.

3.2.1. Organizational Focus

Software development within academic and research organizations is typically done by motives that are either strategic (to fulfil long term vision) or tactical (to meet short term objectives). The different motives usually also indicate different organizational focuses:

- Strategic development: Organizations with a general software development capability as part of another larger activity. Typically computing centres, research centres, collaborations between such organizations, or similar non-profit organizations. Software development is conducted to support a specific user community or niche of users and fill a gap that is not handled by other commercial or community solutions. The programme of software development is well organized and of high strategic value for the organization. The development and result could be kept in-house, run as a community effort or whatever forms fulfilling the organizations strategic goals. Sustainability is achieved through creation of different revenue streams being anything from public funding to commercial sales.

- Tactical development: Groupings, organizations or projects dedicated to a specific research area or research topic perform software development primarily to fulfil their own needs. The whole range of maturity can be found when it comes to organization and sustainability of the software development. From ad-hoc short lived initiatives to well established community developments, a “food chain” of community benefit and effort. Eventually, also commercial opportunities can appear. Often one organization emerges for which the software becomes of strategic importance, and hence critical for its (commercial) survival (thus the development moves from being tactical to strategic).

3.2.2. Classes of stakeholders

There is a differentiation between the different stakeholders of applications and libraries spanning users, developers and providers. The Study of User Priorities for e-Infrastructure for e-Research [29] classified stakeholders into the following categories which are generally useful when considering scientific software:

Researchers

- Casual Users (Novice or Inexperienced)
- Intensive Users (Expert or Focused) [includes computing specialists]

Technologists

- Assemblers of domain components/services/tools
- Builders of domain components/services/tools
- Assemblers of generic components/services/tools
- Builders of generic components/services/tools

Infrastructure providers

- VO/Consortium Managers
- Resource Owners
- Helpdesk and Training Providers
- System Administrators

In addition, we can add funders and maintainers to this list.

3.2.3. Requirements for a CESS

Understanding the requirements for a CESS is a complex task. There are differing classes of stakeholders, and prioritization of requirements requires trade-offs. We describe the mechanisms for collection of requirements as well as the classes of stakeholders which must be consulted.

Forms of requirements gathering

There are several mechanisms for gathering requirements from the various stakeholders.

- Existing *requirements* can be understood via stakeholder surveys, user observation, user forums (e.g. at EGI or PRACE conferences), document review, and analysis of other requirements exercises;

- *Known new requirements* can be collected via feature requests, focus groups, stakeholder interviews, and undertaking joint application design;

- *Unknown requirements* can be defined by arranging brainstorming with stakeholders, by developing prototypes, which are then reviewed by stakeholders, or by usage analysis of existing software (c.f. nanoHUB [28]).

Previous requirements gathering exercises and mechanisms in the area of scientific software and infrastructure from Europe and the US include the Study of User Priorities for e-Research [29] which comprised face to face unstructured interviews, an online survey, and a workshop; the TeraGrid Evaluation Study [30] which featured telephone interviews, participant observations, a user workshop, document analysis and review, and two surveys; and the JISC Community Engagement projects (ENGAGE, e-IUS and e-Uptake) which engaged in a coordinated set of semi-structured interviews of stakeholders, refined by a literature review, prototyping, and example narratives.

The different methods provide different kinds of information and each of the existing exercises have shown that it is important to include more than current known users in the requirements gathering. However current methods, whilst effective, can be time-consuming and resource-intensive.

Trade-offs of requirements gathering

There are several trade-offs that must be made when assessing software requirements for a wide community such as exists in Europe. These include:

- Community vs. product: how much emphasis should be placed on satisfying known community needs versus product innovation?

- Top-down vs. bottom-up: how much should the process seek requirements from a few key figures versus large scale surveys?

- Specialised vs. generalised: how much should requirements be prioritised where the work is general versus satisfying concrete specialist requirements?

- Potential for transfer of functionality to new domain vs. forking of community

Prioritisation of requirements

Once requirements have been collected it is important that a process by which the information is able to be processed and reduced to a set of key findings is designed. This should not though damage or destroy the original data. One method that has been developed successfully for this (e.g. on the UK ENGAGE project) is the use of a Triage Evaluation Questionnaire which is used to formalize the review process and record its methods. The goal was simply to identify a core set of metrics that would identify the most promising, independent of cost. Further work by the Software Sustainability Institute [31] has generalized these metrics into:

- Importance: how well aligned is the requirement with stakeholders' strategic priorities?

- Value: what is the estimated impact that the satisfaction of the requirement will have on the stakeholders?

- Tractability: what is the likelihood that the requirement will be possible to implement? What is the enthusiasm and stability of the group working with the software?

- Opportunity: is this the right time in the cycle for investment? Will the satisfaction of the requirement lead to new opportunities?

- Stage: will the satisfaction of the requirement lead to an incremental improvement, or a revolutionary change?

Through this process, the potential work that might be carried out by a software centre for excellence can be assessed and either put into action immediately, put into the pipeline for commissioning when current projects are complete, or returned to the community either to seek joint funding from other sources if the scope of what is proposed is too large, too risky or out of scope, or to be reassessed by the community.

3.2.4. Organizational Models and Interfaces

A network of centres of excellence for scientific software development can be formed in many different ways. It could be anything from a loosely coupled network defined by at most a simple MoU, to a full consortium with a common legal entity, governance model and structure for coordination and collaboration.

A sustained software development needs to have a minimum of clear interfaces towards customers, stakeholders, and partners. The interface to customers or users is at least on two different levels: community influence on long-term goals and direct customer interaction such as support, maintenance and feature requests. The interface to stakeholders is formal and must ensure that the stakeholder's strategic interest is the overall directing factor. The interface to partners must be flexible as the possibilities of partnerships can be several and for different purposes: collaboration, reselling, marketing, and business development to mention a few.

For example, in the UK, the EPSRC-funded Software Sustainability Institute is a formal collaboration between four universities (Edinburgh, Manchester, Oxford and Southampton) each with existing scientific software development initiatives. The SSI makes use of the skills and experience of staff within the collaborating organizations to create an organization along the lines of a CESS which concentrates on strategic interventions to improve best practice and training amongst the researcher developer base, as well as tactical interventions to support specific projects achieve a transition to the next stage in the lifecycle of their software.

In the US, the organizations funded under the Software Infrastructure for Sustained Innovation (SI2) programme [32] are categorized into three sizes: Scientific Software Elements awards target small groups that create and deploy robust software elements for which there is a demonstrated need in a specific area; Scientific Software Integration awards target larger, interdisciplinary teams organized around the development and application of common software infrastructure aimed at solving common research problems and which should result in a sustainable community software framework serving a diverse community; and Scientific Software Innovation Institutes which focus on the establishment of long-term hubs of excellence in software infrastructure and technologies, which will serve a research community of substantial size and disciplinary breadth. A CESS as outlined in this section is envisaged to be similar in size and scope to this latter model of a Scientific Software Innovation Institute.

The organizational interface to the software industry and e-Infrastructure providers are discussed in more detail in the subsequent sections.

3.3. The role of Software Industry

3.3.1. Software vendors targeted by the report

Since the central focus of this report is on the scientific software layer, the primary software industry sector targeted comprises independent software vendors (ISVs) offering stand-alone codes for specific applications (for example, numerical simulation of specific phenomena) or offering general purpose software suites or technical computing environments that may be applicable for a range of applications (for example, general purpose Finite Element solver packages and supporting tools, including the integrated or complementary pre- and post-processing packages). Nevertheless, there will be a necessity for the CESS to enlarge the scope of activities to ensure that there is a match-up between the available computing infrastructures and the development of parallel applications software – meaning that there will also be a secondary target of industrial software providers in the parallel libraries and tools layer of the overall HPC technology stack. Software vendors in the latter include ISVs but also the IT (hardware and system) vendors.

3.3.2. Open Interfaces and industrially-maintained software

Taking into consideration the overall HPC software eco-system, there needs to be the possibility for academic / research community and industrial software developments not only to co-exist, but also actively complement each other. One key reason for this is that one of the major issues for users is the support and maintenance of software. Even when there is a growing interest in the use of open-source software in industry, it is important that the software systems created within the Scientific Software centres of excellence allow for commercial support and development for industrial use. Open standards and interfaces should be in the strategy for CESS [33].

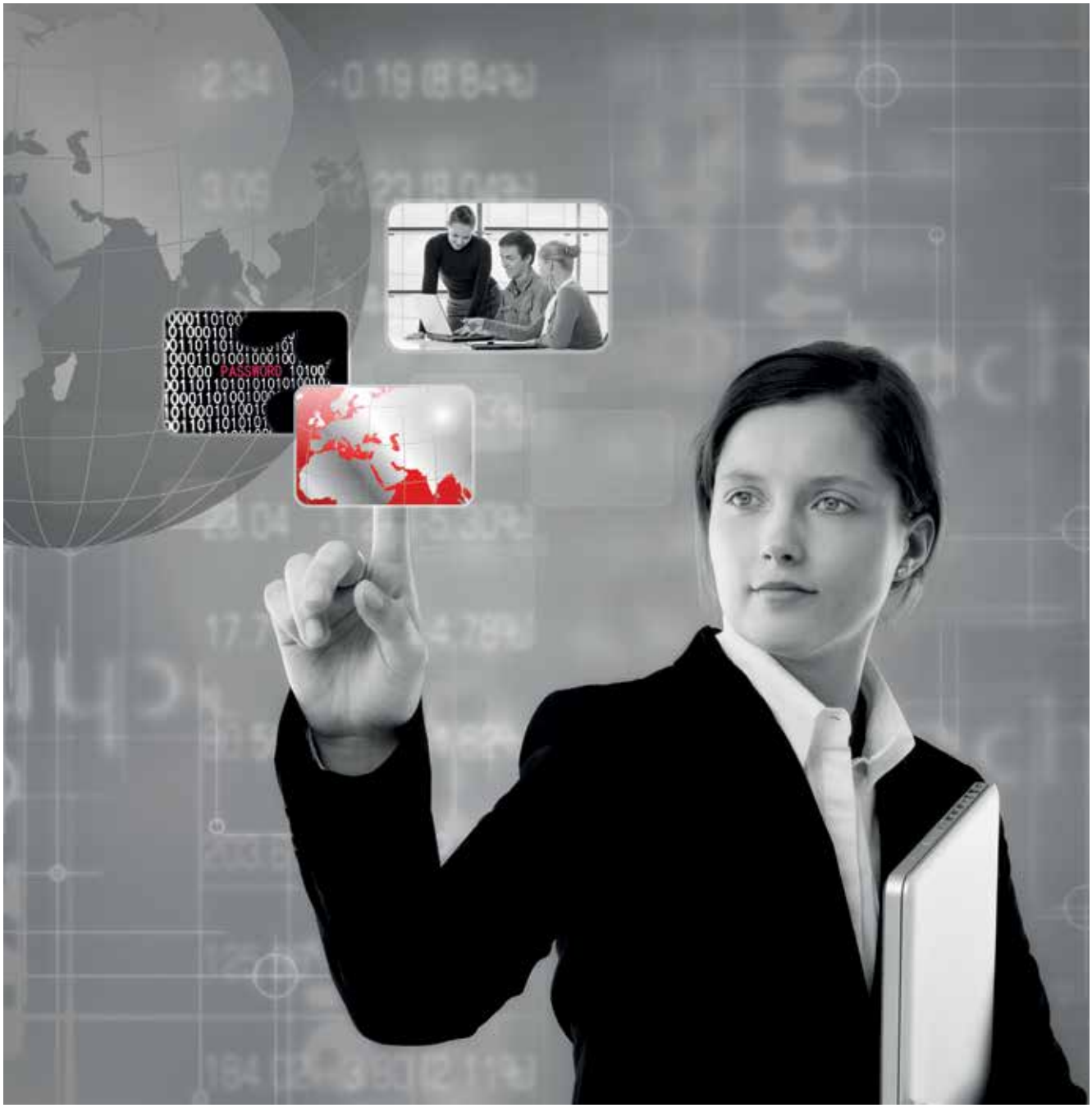
The software ecosystem developed by the academic community within the CESS should be designed and created such that it would be feasible for ISVs (and the target industrial software developers in general) to provide value-added software and services, which are consistent with that software ecosystem (whereby consistency is understood to mean: be interoperable with and/or build upon). For this, the application programming interfaces (“APIs”) and support library interfaces should be open and non-restrictive, allowing for the development of commercial (possibly proprietary/closed-source) complementary variants and supporting software tools. The targeted software industry should have the opportunity to collaborate with the CESS in the definition of the open interfaces.

3.3.3. Academic/Industrial Software “Co-Design”

Looking beyond the need for the complementarity of community and industrial/commercial software developments, the CESS can play an active and catalytic role in creating opportunities for collaboration between those groups. In a manner similar to that being discussed for the development of exascale computing systems (where the importance of alignment of software and hardware design has been recognized), the centres should create a collaborative environment facilitating the “co-design” of academic community and industrial applications software. The co-design would encompass the definition and/or adaptation of programming models and languages – to be used with scientific software – as well as the computing environments and middleware within which applications will be deployed.

3.4. The role of e-Infrastructure Providers

The e-Infrastructure providers in Europe are key stakeholders in using and deploying the scientific software stacks, even though they are usually not directly benefiting from the software themselves. As such, they are customers of the software producers and operate the hardware, install and maintain the software, which is then being used by scientists for their daily work. This raises the need for a well-defined interface to software centres of excellence, which provide a component of the e-Infrastructure services, while scientists depend on both, the services provided by the e-Infrastructure providers and the quality and functionality of the software developed by the software centres of excellence.



The relations between the e-Infrastructure providers and the CESS are manifold:

- e-Infrastructure providers rely on the software developed by the software centres of excellence to
  - Operate, maintain and manage their resources
  - Deploy and operate basic services for scientific users
  - Provide specific functionalities on higher levels of abstractions for scientists to perform their scientific tasks

- The quality of the software is therefore part of user satisfaction concerning the services provided by the e-Infrastructure providers

- The CESS require interaction with the e-Infrastructure providers specifically for:
  - Evaluation (feasibility) or large research collaborations' needs
  - Deployment and testing of software and updates

Obviously, these close connections between e-Infrastructure providers and software centres of excellence must be clearly defined and established. At the same time, e-Infrastructure providers need to maintain the option of multiple sources for software functionality, as a healthy market (of scientific software) will only work with multiple software providers.

The Universal Middleware Distribution (UMD) [34] of EGI is a good example on how to establish such a relationship, although it is clearly on the lower levels of the software stack. Within EGI, the components of UMD are defined based on the functionality provided to the users. The implementations of these functionalities can then be offered by multiple software providers, in the case of UMD for example from EMI [35] and IGE [36], therefore establishing an offer with multiple options. For some functions, EMI provides alternative software options. However, the options chosen by users are often determined by given factors such as the utilization of an earlier version of the software and thus a reduced update effort and learning cycle on behalf of the users. A range of different options providing the same functionality creates a setting where quality of software is a much more important selection criterion and as such. The software market could then steer the software landscape.

In the US, the nanoHUB is a resource for nano-science and nanotechnology with over 225,000 users. nanoHub brings together e-Infrastructure providers, research users, and students to provide simulation tools through common environments. One significant innovation is that the nanoHub portal captures information about the usage of tools, which enables new users to be categorized and provided with appropriate resources. It also enables new patterns of usage to be identified heuristically and the requirements of these new types of users to addressed more easily and addressed by the e-Infrastructure providers. This is done in conjunction with the more traditional approaches to requirements capture and stakeholder engagement.

For software centres of excellence to work, it is essential to establish a well-defined basis to the e-Infrastructure providers and vice versa, while at the same time establishing clear distinction between the operations part on the one hand and the development part on the other hand.

While industrial access to e-Infrastructures for production/commercial purposes is a topic on its own, access for evaluation purposes or in the framework of providing industrial requirements for software development needs to be in scope for CESS.

3.5. Funding and governance

A CESS for enhancing and maintaining scientific software must have a sustainable and well-defined organizational form, which in turn implies that both a sustainable governance model and a long-term funding scheme are needed.

It is imperative that the governance structure is fully based on caring for the needs (both long-term and short-term) of the users of the software, i.e. researchers in academia and industry. In short, these researchers should "be in the driving seat" and govern the CESSs - taking advice from other stakeholders and related entities. Here the interplay between organizational form and governance is also important. A CESS can be a legal entity on its own or hosted within a larger organization but important is that the organizational form should be chosen such that it will be possible to have a primarily user driven governance, avoid too strong governance influence from other stakeholders, like providers of computational resources or ISVs.

The structure and organizational strength of the user communities varies greatly between different fields and setting up a governance structure for user-driven CESSs for application software enhancement and maintenance will be a challenging task. A balance between "local", short-term goals and "global" long-term ventures must be found and guarantees that input from relevant stakeholders outside the user communities is heard, must be built in. In some fields, the research community is very diffuse and it may be hard to find persons that can represent the area (and not only their own group or sub-field) in a good way. In other cases the community may be very well organized but it may be hard to guarantee that a sufficient level of strategic innovation is brought into the work of the CESS. Also, if ISVs and other commercial actors are involved in the CESS this adds another level of complexity to the governance structure. It is important to fully acknowledge these challenges early on in the process of defining and setting up the CESS.

It is also important that the governing structures of the CESSs are strong enough to properly alter the direction of, or even close down a CESS, which is not fulfilling the user needs in an efficient way. This means that a sustainable and firm model for evaluating progress and efficiency of the CESS should be developed and integrated in the organizational and governance models. Here, the evaluation should go beyond a traditional review of accounts and deliverables and focus more on exploring the impact of the work by the CESS within the users, both in terms of short-term scientific and innovation output and in terms of bringing future research capabilities and competitive advantages to the researchers. As a result, CESS should set up advisory boards consisting of representative user communities, who support and provide input to strategic discussions of the CESS.

The user communities also play an essential role when it comes to establishing a sustainable funding scheme for application software enhancement and maintenance. Financial contributions from these communities are important both for motivating the governing role and for establishing the necessary level of commitment. However, it is clear that the ability of raising such funds among users communities is limited today. Also, in many cases the maturity of the user community has not reached the level where such funds can be assembled at the national or European level.

To ensure that sufficient funds can be secured for scientific software, CESSs should be recognized as entities providing versatile and widely used scientific instruments, in this case software, and hence become an

integral part of the research infrastructure landscape. Here, a change of attitude and strategies is needed among both funding agencies and user communities. It is also urgent that these chances are imposed both at the national and European levels, including policies for channeling national funding to CESSs located in different countries. The strategies of funding agencies need to be changed so that funding for hardware and provisioning of computational resources is complemented by comparable efforts on scientific software enhancement and maintenance. Within the user communities stronger organizational frameworks must be put in place and a wider understanding needs to be developed that enhancement and maintenance for major scientific software needs to done in a more organized, forward-looking and cross-disciplinary fashion than is often done today.

Funding for organizations that develop and support software typically go through different stages where the number of stakeholders increase and the funding diversifies. Typically most start off being grant funded and then explore other models such as foundations or commercialization. Many European research computing organizations (e.g. PRACE, EGI) seek to move from a "defined contribution" model where member countries pay a subscription to a true "pay-per-use" model where services are provided to a paying customer market. However there is still insufficient evidence to identify if these models are suitable and further studies are recommended to identify the best models for funding a CESS.

Nevertheless it is clear that the community as a whole needs to move away from a culture which treats software as being a disposable consumable and instead enables the channeling of funding from multiple European and national funding programmes into sustainable CESSs. This was well articulated in the RCUK Review of e-Science [16] led by Dan Atkins which said: *"We suggest creating and sustaining long-term centres for software development and support. They need consistent funding at a significant level, not just intermittent support from research projects […]. Software development is not basic research, but instead requires a critical mass of full-time engineering professionals, paid market wages and supported along a genuine career path. There needs to be a commitment to long-term funding and a recognition that software development and support is at least as important to modern science as massive accelerators and telescopes."*

Different means should be explored to quickly accomplish the changes needed to guarantee sustainable funding for the software enhancement and maintenance CESSs. One important tool here is funding at the European level for initiating structures like the proposed CESSs. However, such efforts should be carefully set-up to ensure that the efforts are governed by the user communities and that the sustainability of the activities is considered already at the start.

4. Summary of main recommendations

Based on the situation as described above, and the findings of the e-IRG Task Force on Scientific Software, we propose the following strong recommendation:

- To address the software crisis, the EC and the member states should provide support and funding for the establishment of Centres of Excellence for Scientific Software (CESS), currently focusing on scientific software on the application layer using a holistic approach and building up and retaining the necessary competence of future European software developers.

The CESS themselves are further described in the following recommendations, defining particular characteristics and attributes as revealed in this document:

- A CESS should support the entire software lifecycle, including all phases from design to maintenance, and on all levels, from tera- and petascale today to exascale tomorrow, within the European e-Infrastructure ecosystem.

- A CESS should create a collaborative environment facilitating the “co-design” of academic community and industrial scientific software, and allow ISVs to provide value-added software and services within this ecosystem.

- A CESS should provide a transparent process for requirements gathering and prioritization, using established and well-known methods as well as developing new ideas for requirements gathering for both, existing and future upcoming communities.

- A CESS should implement interfaces to the e-Infrastructure providers and vice versa, while establishing clear responsibilities for operations on the one side and development on the other side.

- A CESS should work with other organizations to develop and promote appropriate curricula across Europe that provides researchers with a foundation in computational science and distributed computing.

- A CESS should have a governance model, ensuring that user communities drive the scientific strategy. Experts on governance need to be involved in the setup of a CESS.

As an additional action beyond the scope of the e-IRG Task Force on Scientific Software, the following more in-depth investigation is recommended:

- The different potential organizational and funding schemes need to be discussed further to derive sustainable solutions for CESSs. It would be best to perform a design study project in order to investigate possible impacts and the advantages and disadvantages of different models.

References

[1] **European Grid Infrastructure**
www.egi.eu

[2] **Partnership for Advanced Computing in Europe**
www.prace-project.eu

[3] **Scalable Software Service for LifeScience**
www.scalalife.eu

[4] **Multiscale Applications on European e-Infrastructure**
www.mapper-project.eu

[5] **International Exascale Software Project**
www.exascale.org

[6] **European Exascale Software Initiative**
www.eesi-project.eu

[7] **Collaborative Research into Exascale Systemware, Tools & Applications**
cresta-project.eu

[8] **Dynamic Exascale Entry Platform**
Website is under construction (07/17/2012)

[9] **Mont-Blanc Project**
www.montblanc-project.eu

[10] **IDC Special Study - Financing a Software Infrastructure for Highly Parallelized Codes - IDC FINAL Report for the DG Information Society of the European Commission**
cordis.europa.eu/fp7/ict/e-infrastructure/docs/fisofi4hpc.pdf

[11] **Swedish e-Science Research Centre**
www.e-science.se

[12] **eSSENCE - The e-Science Collaboration**
essenceofescience.se

[13] **Engineering and Physical Sciences Research Council**
www.epsrc.ac.uk

[14] **EPSRC Software as an Infrastructure Strategy and Action Plan**
www.epsrc.ac.uk/ourportfolio/themes/researchinfrastructure/sub-themes/einfrastructure/software/Pages/default.aspx#strategy

[15] **Strategy for the UK Research Computing Ecosystem (V6.2 19/9/11)**
wiki.esi.ac.uk/w/files/7/7b/ResearchcomputingV62-final.pdf

[16] **RCUK Review of e-Science**
www.epsrc.ac.uk/SiteCollectionDocuments/Publications/reports/RCUKe-ScienceReviewReport.pdf

[17] **e-IRG Support Programme 3**
www.e-irg.eu/about-e-irg/e-irgsp3.html

[18] **"The 1968/69 NATO Software Engineering Reports"**
homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports

[19] **A view of the parallel computing landscape, Krste Asanovic et al.**
dl.acm.org/citation.cfm?id=1562783

[20] **PRACE – D5.4 Report on the Application Benchmarking Results of Prototype Systems**
www.prace-ri.eu/IMG/pdf/D5-4-extended.pdf

[21] **PRACE – D7.4.1 Applications and user requirements for Tier-0 systems**
www.prace-ri.eu/IMG/pdf/D7-4-1_1ip.pdf

[22] **The Message Passing Interface (MPI) Standard**
www.mcs.anl.gov/research/projects/mpi

[23] **The openMP API Specification for Parallel Programming**
openmp.org

[24] **openCL – The Open Standard for Parallel Programming Of Heterogeneous Systems**
www.khronos.org/opencl

[25] **Allinea DDT - the debugging tool for parallel computing**
www.allinea.com/products/ddt

[26] **TotalView - Dynamic source code and memory debugging for C, C++ and Fortran applications on UNIX, LINUX and MAC OS X**
www.roguewave.com/products/totalview.aspx

[27] **Software Carpentry**
software-carpentry.org

[28] **nanoHub portal for nanotechnology**
nanohub.org

[29] **Study of User Priorities for e-Infrastructure for e-Research**
www.nesc.ac.uk/technical_papers/UKeS-2007-01.pdf

[30] **Report from the TeraGrid Evaluation Study, Part 1: Project Findings and Part 2: Findings from the TeraGrid User Survey**
hdl.handle.net/2027.42/61838
hdl.handle.net/2027.42/61839

[31] **Software Sustainability Institute**
www.software.ac.uk

[32] **NSF Software Infrastructure for Sustained Innovation**
nsf.gov/si2

[33] **Roadmap for Open ICT Ecosystems**
cyber.law.harvard.edu/policy/

[34] **Unified Middleware Distribution**
repository.egi.eu/category/umd_releases/

[35] **European Middleware Initiative (EMI)**
www.eu-emi.eu

[36] **Initiative for Globus in Europe (IGE)**
www.ige-project.eu