2.4 First intermediate report on the state of the art of worldwide co-design centres

CONTRACT NO EESI2 312478
INSTRUMENT CSA (Support and Collaborative Action)
THEMATIC INFRASTRUCTURE

Due date of deliverable: 31-05-2014
Actual submission date: 31-05-2014
Publication date:

Start date of project: 1 September 2013 Duration: 30 months

Name of lead contractor for this deliverable: EPCC, Nick Brown
Name of reviewers for this deliverable:

Abstract: This is the first intermediate report of EESI2 WP2.3 state of the art of worldwide co-design centres. The document reports on the initial findings and presents initial recommendations.

Revision draft

Project co-funded by the European Commission within the Seventh Framework Programme (FP7/2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Contents

1. EXECUTIVE SUMMARY 3

2. INTRODUCTION 4

3. MAIN SECTION 6

3.1 QUESTIONNAIRE 6
3.2 TARGETED INTERVIEWING 12
3.2.1 What is their definition of co-design? 12
3.2.2 How often have they used co-design? 13
3.2.3 What parties are required to reach exa-scale? 13
3.2.4 What are the downsides of using co-design? 14
3.2.5 What needs to be changed to make adopting co-design more attractive? 15
3.2.6 How does co-design compare against centres of excellence? 16
3.3 EXISTING APPROACHES 16
3.3.1 Application proxies 17
3.3.2 US Department of Energy co-design centres 18
3.3.3 US Department of Energy co-design process 20
3.3.4 CRESTA 21
3.3.5 DEEP 22
3.3.6 MONT-BLANC 22

4. CONCLUSIONS 23

4.1.1 Recommendations 23

5. BIBLIOGRAPHY 24

Glossary

Abbreviation / acronym	Description
DoE	US Department of Energy
Mini app	Small self-contained program that embodies essential performance characteristics of key code
SIMD	Single Instruction Multiple Data
HPC	High Performance Computing
SSI	Software Sustainability Institute

1. Executive Summary

The HPC community have set themselves the target of reaching exa-scale performance by 2020. It is not a question of whether this target will be met, but instead whether or not an exa-scale machine can be programmed and used effectively to perform science at a scale never before attainable. There are many challenges in many different areas to overcome if the community are to scale their current scientific codes up to the exa-flop level and one of the approaches to solving this is through the use of co-design, where interdisciplinary teams work together to solve a specific problem.

However this development methodology is still relatively immature and in this report we gather the opinions of co-design from those in the HPC field, consider the positive and negative impact of such an approach, survey how co-design is currently being used and make some recommendations about how European projects might more effectively take advantage of this methodology to help solve the exa-scale challenge.

2. Introduction

The demands on HPC resources are ever increasing as the community wish to complete more science. In recent years hardware and software have evolved independently and if the continuing pace of development holds then it looks likely that we will move from hardware currently capable of the peta-scale (10^{15} flop/s) to exa-scale (10^{18} flop/s) by the early 2020s. On the other hand the pace of development on the software side has been much slower and if the community is to take full advantage of this relentless improvement in hardware then a great deal of work is needed to update many other aspects of the HPC life cycle including the mathematical methods, algorithms, languages and development tools. Many in the community advocate a revolutionary change, as opposed to evolutionary, where traditional development approaches and technologies are replaced by more modern practices. Numerous literature documents the technical challenges to reaching exa-scale [1] which impact all areas of HPC. In summary these are:

- **Power:** To continue designing supercomputers using existing technologies is not sustainable as the power requirements of such a machine rapidly become prohibitive as detailed in [2]. A goal has been set to achieve exa-scale performance within a power limit of 20MW and this has implications for all aspects of the HPC life cycle. It might very well be the case that a code's energy use might replace CPU time as the cost metric.
- **Extreme concurrency:** It is estimated that the parallelism of an exa-scale machine will be two or three orders of magnitude greater than current architectures. Existing programming models, tools and algorithms will not scale to this level and as such new technologies must be developed to take advantage of these future machines.
- **Limited memory:** Memory density is not expected to increase at the same rate as parallelism and therefore the amount of memory per core is likely to decrease. Combined with the fact that memory is a major draw on power means that it is likely that new methods will need to be developed which minimise memory usage.
- **Data locality:** Similarly to memory, the communication technology (whether it be inter or intra node) is not expected to develop as fast as processor technology. Therefore codes will need to be aware of data locality and optimised to minimise data movement. This is already happening in some small way, where developers are choosing to recompute values rather than store them due to the performance and increasingly power implications.
- **Fault tolerance:** Due to the large number of components in an exa-scale system, it is likely that there will be some failure during the lifetime of a non-trivial simulation. Existing methods are not expected to be sufficient to answer this problem and it is believed that mathematical properties of algorithms will need to be leveraged to solve it.

Co-design is a process where partnerships are created involving parties with distinct expertise who then collaborate to solve a specific problem. Central to applying this to the HPC field is the idea that the HPC life cycle has now become so complex that traditional approaches of domain experts developing scientific codes in isolation to their field of expertise will not be enough to reach exa-scale. Instead the co-design development methodology is oriented around a feedback loop where a specific application or science drives interdisciplinary teams and the underlying facets, such as architecture and language design, interact closely with the scientific codes to ensure that these are fit for the task in hand.

The use of co-design in HPC development teams is currently at an early stage. The past few years have seen a number of projects and centres trialling the methodology and a variety of successes relating to these have been reported. However, due to the immature nature of this approach its implementation has been quite different by different organisations and there is still no overall answer as to whether co-design, and in what form, should be adopted more wholesale by development teams to meet the exa-scale challenge. The aim of this work is to investigate the current state of co-design and aims to answer:

- **What do people in the HPC community consider co-design to be and from this can an accepted definition be deduced?** Because of the relative immaturity of using co-design in

HPC it is important to understand how members of the community might see this approach differently. Whilst one team may believe that they have adopted co-design effectively, in fact they might be missing a critical component which impacts adversely.

- **What projects are currently using co-design and how has it been implemented?** It is important to understand what projects are currently using co-design and what they hope to achieve from adopting this methodology. It is interesting to understand how each project has implemented co-design which may depend heavily on the nature of the work and parties involved.
- **What are the pros and cons of adopting co-design?** Both in terms of the projects currently using co-design and the perceived pros and cons by those in the HPC community. By considering the existing uses of co-design one will be able to understand how some of these negatives have been mitigated.
- **What changes are required to support the adoption of co-design?** What support from the community and institutions is required and does one need to change their ways of working to encourage the adoption of co-design and support interdisciplinary teams.
- **Based upon the above, is the co-design methodology realistic going to help the HPC community reach exa-scale?**

To answer these questions a variety of approaches have been adopted; A general questionnaire was produced and open to all with the aim of canvasing a wide selection of opinion. Secondly, with the assistance of other EESI-2 partners, a list of influential and relevant individuals in the HPC community has been assembled who we have targeted for an in-depth interview about their experiences and opinions of co-design. Thirdly a through literature review as to the current co-design methodology, uses and results have been completed along with informal communication made at a variety of HPC conferences and workshops. In this report we detail the results of our investigations, consider the points raised with respect to current practice and conclude by making some suggestions based upon the information gathered.

3. Main section

3.1 Questionnaire

As a first step we created a questionnaire [3] which enquired about co-design and people's experiences of it. This questionnaire was designed to be a light weight approach to collecting as many opinions as possible and Figure 1 illustrates the content where each question is optional and the exact flow depends upon answers previously provided to better extract relevant information. This survey starts with some inquiry about the individual and then considers what co-design means to them. Asking for a very general definition of co-design not only helps frame the remainder of their answers, but was also was to inform whether the community generally agreed on a definition or if there are substantial variations. A fork is encountered and, depending upon whether the subject has used co-design before or not, they are either asked what other parties were involved and how positive their experiences of co-design were or why they did not use the methodology and what might be done differently to encourage their future use of co-design. If the subject had previously used co-design and if co-design had had a positive impact upon their project then we ask for an example of this, if it had a negative impact then they were asked what the issues were and how they might be mitigated. Regardless of answers, the forks then re-join to ask about the challenges exa-scale will present to them and how co-design might address these. It is expected that the survey would take no longer than five minutes to complete and it was advertised on fliers and available at the EPCC both of Super Computing 2013, amongst the EESI-2 partners, around users of the UK national supercomputing service and other HPC contacts.

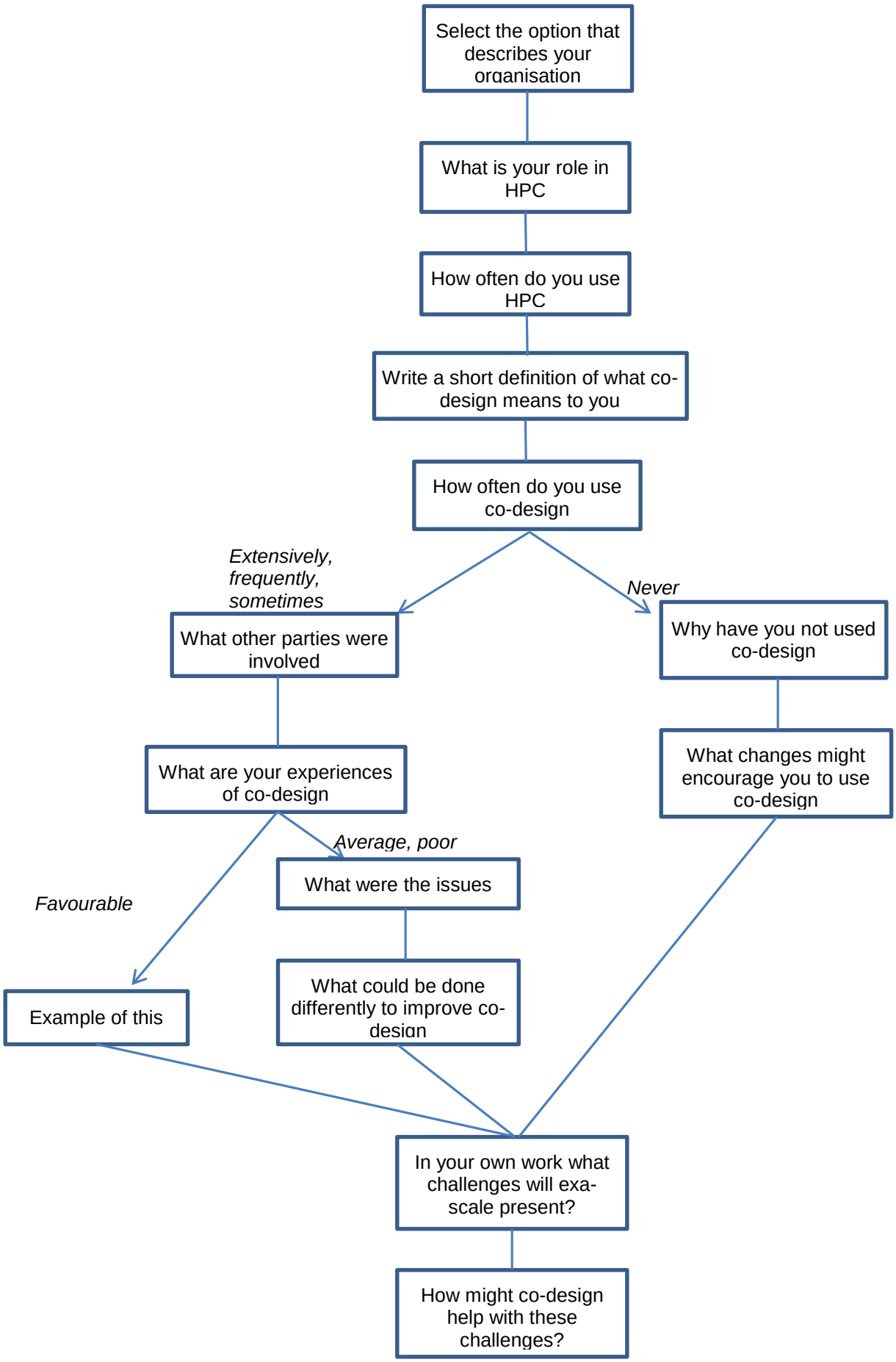

Figure 1 - Questionnaire flow

Erreur ! Source du renvoi introuvable. illustrates the type of usage that our responders make of PC and from this it is clear that a variety of different users have been surveyed each with their own backgrounds, experiences and opinions. Figure 2 depicts how often the responders use HPC and one can see that over half consider themselves to be frequent HPC users. Because the questionnaire was advertised mainly within the HPC community it is not surprising that there is a high usage. Eighty percent of the participants were from academic institutions and the rest classed their organisation as commercial.

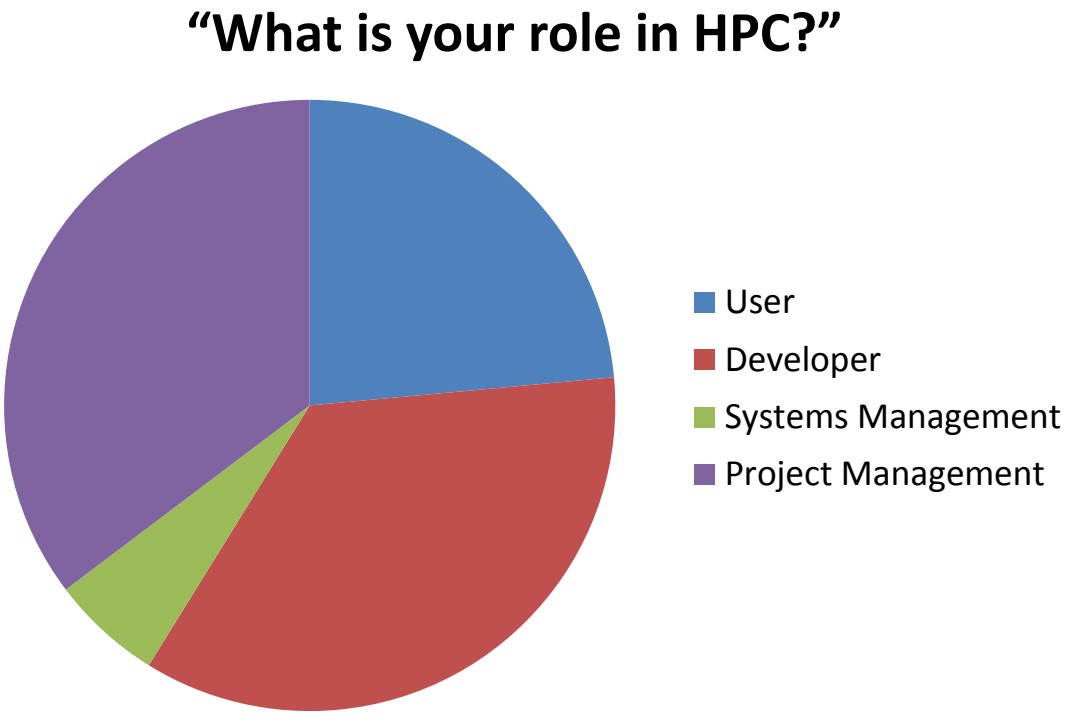

Figure 2 – “What is your role in HPC?”

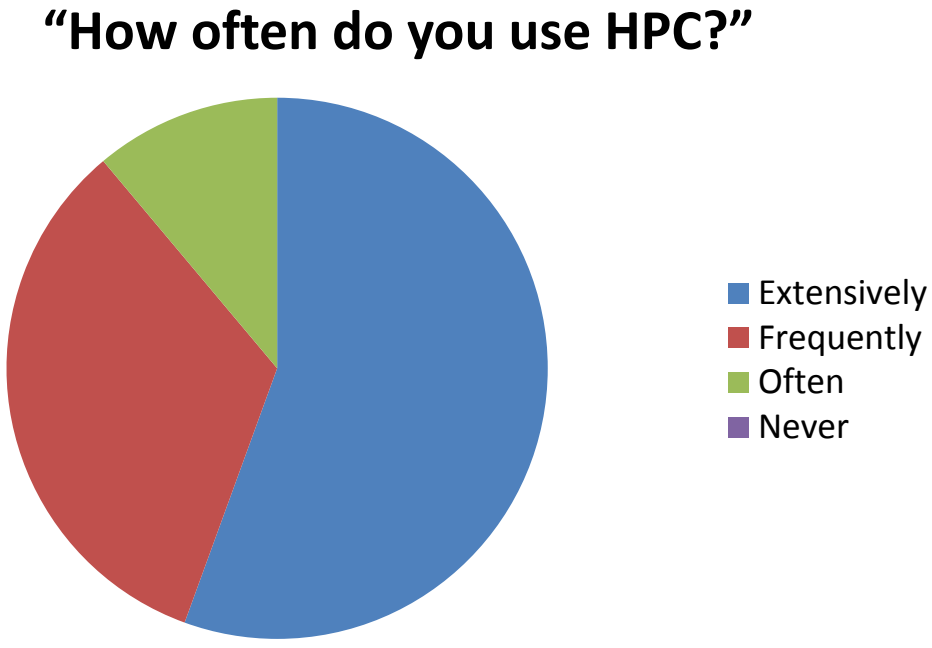

Figure 3 - "How often do you use HPC?"

Figure 4 illustrates the answers to the question ``How often do you use co-design.'' It can be seen that nobody replied extensively, the most common response was sometimes but some people considered that they used co-design frequently and others never. It is interesting to see that a reasonable percentage of participants do consider themselves to use co-design to some extent but nobody that completed the survey has adopted the methodology extensively in their work. Responses to the next question ``What other parties were involved in co-design'', where multiple answers were allowed, are

shown in Figure 5. The most common ``other parties'' were domain experts and mathematicians, with hardware engineers and end users still prominent but less so. It is interesting that nobody selected the ``Software'' option, and that might illustrate that either they consider themselves to be software engineers and take this as red or the option was not well defined. Some participants selected all of the boxes which would illustrate that they are considering large projects with many different partners.

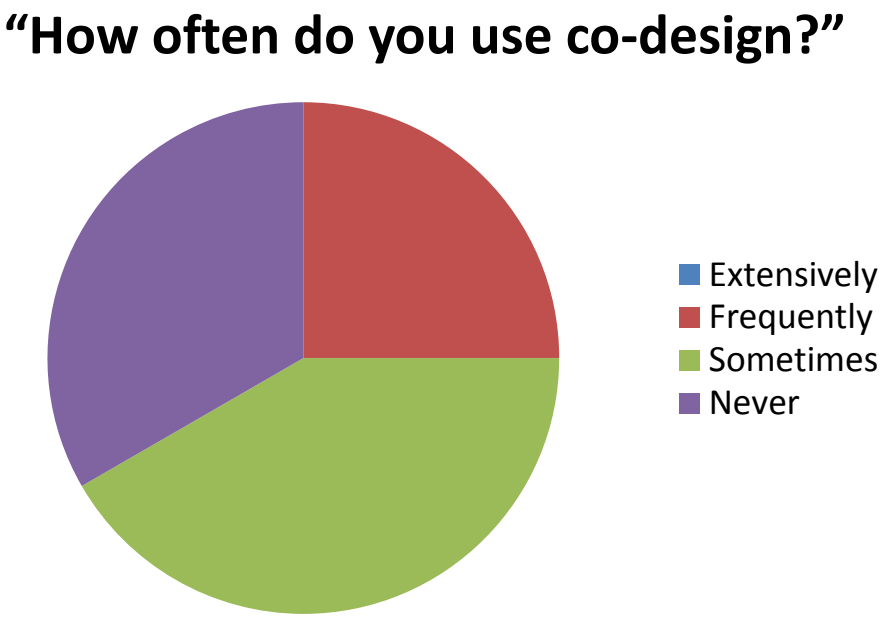

Figure 4 - "How often do you use co-design?"

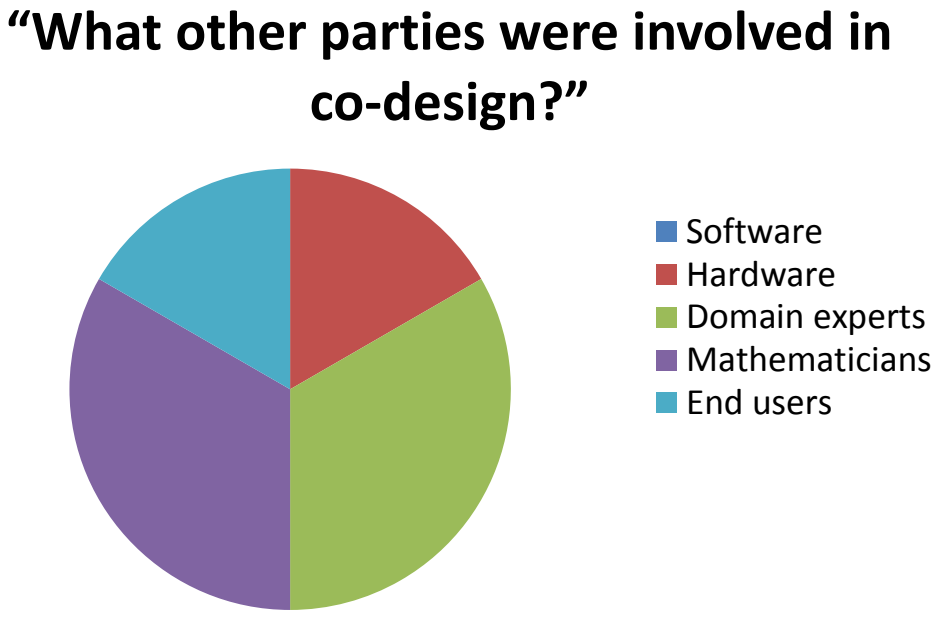

Figure 5 - "What other parties were involved?"

The results of a key question are depicted in Figure 6 where we asked how the participants rated their experiences of using co-design. The majority voted favourably indicating that using co-design was a better option than not using it, although a couple indicated that in their opinion there was no difference

and it had not made an impact upon the project. Nobody said that they felt adopting co-design was unfavourable and this might mean that either the HPC community as a whole have bought into the co-design methodology or those motivated the complete the survey are passionate about co-design. As illustrated in Figure 1 the questionnaire consisted of a number of questions which were not straightforward multiple choice, but instead invited the participant to write more detailed text. The responses from question about defining what co-design were very interesting, because each response had a different definition and some responses were vastly different. Many people talked about hardware, software and domain experts working together to solve challenges, but one person believed that co-design was all about driving the hardware development and other parties such as the software and domain experts feeding into the hardware design. Whilst many did mention hardware in their definitions, not all did and some talked about developing software by combining different stakeholders. Interestingly nobody mentioned science, the definitions were all hardware or software driven rather than science driven.

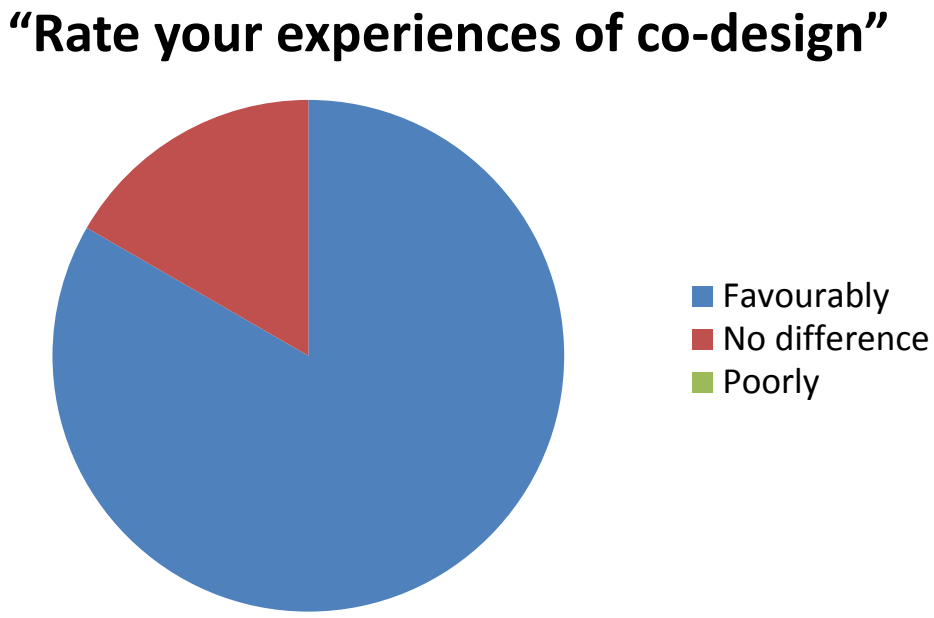

Figure 6 - "Rate your experiences of co-design"

In answering the question about why one doesn't use co-design, one of the answers explained that the individual was both a user and software developer, but doesn't design hardware and-so cannot use co-design. This person is the same one who defined co-design as being entirely hardware oriented and said that if they undertook collaboration with hardware vendors in the future then they would adopt the methodology. Interestingly all responses to the question about the challenges that exa-scale might pose in the future mention hardware, the power challenge is prominent in the responses as are other challenges such as fault tolerance and the limiting of communication. There are some software oriented challenges mentioned, such as the scaling of applications but these are in the minority. The follow-on question which enquires about how co-design might help with the challenges identified saw answers which all discussed collaborations between the hardware and software side. Some talked in detail about who should be involved, such as considering programming models and applications, whilst others just mentioned system software. There was also the point made that one needs foresight of upcoming hardware which will enable the redesigning of applications before this hardware arrives.

The questionnaire has generated some interesting points and is a useful foundation to build upon although it should be noted that there could be a bias because those people who are interested in and use co-design might be more willing to fill in a co-design related survey which could explain the answer to question four where the majority of people said that they frequently or sometimes used co-design. One of the most interesting results of the survey was that the definitions of co-design differed and that at least one participant believed that they could not use co-design because they were not

involved in hardware. From the answers to this question there is the clear indication whilst the term co-design has become well known in the community, and there is no universal definition of exactly what this means, how it might impact upon a project or the correct way of implementing the methodology in practice.

3.2 Targeted Interviewing

Based upon the results gained in the questionnaire a targeted interview was designed to take place either face-to-face or verbally where we could talk to specific individuals who have knowledge and experience of HPC and/or co-design. Each interview followed a similar approach, where there was a broad sketch of questions to ask but the majority of the time was taken up exploring the answers and the individuals' experiences and opinions. An interview lasted approximately twenty minutes to half an hour. The interview started off by asking the participant for a definition of co-design, then enquired how often they use co-design and to provide an example project along with the parties involved. They were then asked about the successes that came out of this and what parties should be involved to reach exa-scale. The interview then started to focus on any perceived negatives of using co-design; asking about the downsides, how these might be mitigated and generally how one might go about improving processes and/or culture to make adopting co-design more attractive. Lastly the interview focused on co-design centres compared to centres of excellence and what the interviewee considered to be the differences and benefits of one approach compared to the other.

This section of the report covers the responses to these questions in depth. A number of comments that were made have been addressed by existing implementations of co-design which are surveyed in section 3.3.

3.2.1 What is their definition of co-design?

There were some varied responses to this question. Many of the definitions centred around an informal definition and some differed to others. The point was made by one that we need to consider the whole pipeline and aspects low down, such as the system architectures, have a big impact upon how one designs facets higher up. Teams need to be integrated together, containing experts from a variety of different disciplines in this pipeline, because HPC has become too complex for the old model of a domain expert working in isolation to their field to be successful. The point was also made that co-design requires feedback and it should be seen as an improvement loop, rather than just one way. Whilst one participant mentioned the *pipeline* word, another stated that they disliked this term because it implies a uni-directional rather than bi-direction flow of information.

A number of people oriented their definition around hardware and talked about hardware working together with software to develop the two in tandem and optimise across that boundary with an example given of the cache; it is helpful for software developers to understand the behaviour of the cache when writing their codes but equally if hardware designers understand what will be run on their machines then they can better tailor the cache to suit. Interestingly one participant said that they believed co-design should only be oriented around software. This was because, in their opinion, hardware moves too slowly when compared to software and the vendors have a long to medium term road plan which means that they are locked into certain decisions. This comment came from a very software focused individual and when the point was discussed with a hardware expert they said that whilst it is true that the silicon is fixed once designed, nowadays hardware is quite soft so it is often trivial to further tune and modify once created. Others mentioned that they actually work with vendors and the feedback acts to drive their road maps.

One participant noted that co-design is sometimes subtle but can still produce important advancements. An example of this is those with similar technical backgrounds but in different communities collaborating together. The example of debugger developers and performance tool writers was given. Whilst there is much technical overlap between these fields, with them are looking at similar facets from different point of views, the communities traditionally do not work together. By

breaking down the barriers to this collaboration, then not only might the different experiences help to develop improved tools, but also the needs of one community might help inform the work of another.

3.2.2 How often have they used co-design?

There were a number of varied answers to this question. Some people never used co-design at all and cited that it was due to their project constraints but when asked each said that they would use it if possible in the future. Other participants were more active in their use of co-design and provided us with a variety of examples to its use, some of which are detailed in section 3.3.

SpiNNaker [4], a novel computer architecture inspired by the working of the human brain, is an illustration of co-design from the ground up. This project has built a massively parallel machine which is designed to represent how many neurons operate. This representation of neurons is not only used to investigate neuroscience and further our understanding of how the brain works, but the non-deterministic and unreliable communication behaviour associated with these raises interesting questions in other areas such as computer science. The SpiNNaker project has seen development in all areas, from hardware designers actually creating the physical machine to low level software engineers writing system drivers to domain writing the application software. The fact that this machine was built from the ground up results in a number of interesting attributes to the co-design process. If building from the silicon up then it can take a long time to have a development system that the software engineers can actually use and then that system is fixed so there is a lack of feedback from the software to hardware. Whilst this can be mitigated to some extent by building hardware in a soft fashion (such as with the use of FPGAs) it is still an issue and therefore funding is required over a long time scale to bring the project to fruition. At the start of this project around 80\% of the participants were hardware designers and 20\% software engineers; as the project progressed this ratio has reversed. Therefore staff and their skills need to be somewhat flexible, one of the ways around this has been the fact that some of the hardware designers also have expertise in writing low level system drivers so once their hardware tasks are completed they can switch over to that job with the added major benefit that hardware knowledge is kept within the project. Being a highly specialist machine built from the ground up, SpiNNaker is an extreme example of a project involving numerous parties with a variety of experiences working together and feeding back. In reality many HPC machines are much more general purpose - but it is still an interesting example as to reach exa-scale some centres with well-defined workloads might opt for specialist machines to solve these. It was the opinion of the interviewee that in many respects the specialist nature made designing SpiNNaker far simpler than a general purpose machine but some parties are interested in using this machine for other uses that the original designers did not envisage.

There were plenty of smaller scale uses that the participants also discussed. Examples of these included finite element and Lattice Boltzmann method where working with end users and domain experts, software engineers would write specialist codes aimed at a wide variety of uses from adaptive manufacturing to medical science.

3.2.3 What parties are required to reach exa-scale?

The simplest answer to this question, and one that was given a number of times, was ``everyone'' and ``as many as possible.'' What the participants were talking about are inter-disciplinary teams with a mix of hardware, software and domain experts. An important point was made that not everybody will be on the project full time, all the time. For instance with SpiNNaker the team began very hardware oriented and then once the hardware was somewhat mature the number of software and domain experts increased. A difficulty is, from an organisational point of view, how to manage this turn over. Academic institutions are in a good position because the average three year duration of a PhD student's study means that this turn over can be handled naturally but these people have still left and their knowledge can be lost with them which is especially disadvantageous when considering the time and effort required to train them up in the first place. Other ways of mitigating this are to use individuals with multiple skill sets and have a number of projects on going so that people move around the projects but are still on hand.

People with energy experience were explicitly mentioned, as per the questionnaire detailed in section 3.1, it was highlighted that hardware and (more increasingly) software must operate within specific energy constraints. The participant that focused on this area provided an example whereby algorithm experts were working with hardware experts to design more power efficient algorithms; this basically boils down to reducing the number of memory operations and replicating some calculations. They believed that this integration is just the start and a similar approach, potentially with more partners, will be required to develop fault tolerant codes.

When considering the parties required the other logical aspect is how they might work with each other. There are numerous collaboration models, for instance ranging from everybody being placed in the same physical location to all working separately and communicating when required. The point was made that, for relatively small teams who are working on the one long term project then it is quite easy to place everybody in the one physical location; this is true of projects such as SpiNNaker where the team are located in Manchester and the specialism of this project naturally attracts interest with a large pool of potential PhD students to pick from who are already there or willing to relocate. For many other projects the physical co-location of the team is not necessarily possible or ideal and the point was made that with modern technology it is very easy to remotely collaborate and share data. in many cases remote collaboration is almost as good as physically being in the same location. It was the general consensus that to reach exa-scale one single co-design centre for a specific domain or area might not have anywhere near the same impact as spreading these people and their expertise out. One of the participants responded with the fact that regardless of how the team physically interacts, for co-design to be successful then most importantly one must drop barriers to collaboration.

Another point made was that, much more important than the physical location, is the whole co-design culture which also includes initiatives such as workshops and boot camps. The Software Sustainability Institute (SSI) hold a number of these within Europe and it is a distributed way of sharing knowledge, contacts and experience. The SSI is a good model to further develop people's software engineering skills and might make the difference between requiring and not requiring the full time support of a dedicated software engineer on a project or whether domain scientists can fulfil this role.

3.2.4 What are the downsides of using co-design?

Whilst the vast majority of participants spoken to are enthusiastic about the positive impact that co-design can have upon a project, it was also felt important to understand any negative aspects of using this approach. The first downside raised was one of communication between parties. Bringing together many different people with diverse experiences and backgrounds can make it difficult for these people to sometimes find a common ground in discussions. A number of different approaches were suggested to help alleviate this; one participant suggested that he saw his team as the glue layer, with a more general knowledge than specific experts, which means that they can help facilitate discussions. Other people suggested an approach whereby the experts should also receive some elementary training in other areas, an example of this being hardware experts attending SSI boot camps to develop some understanding of the software development side.

The communication difficulties are not just limited to physical communication between parties. Additionally there are issues such as conflicting agendas and too many people feeding in too many project requirements. One participant said that it is important to incentivise the interdisciplinary teams and build trust amongst the members. There can often be a push to develop an institute's research funding or kudos and this can sometimes be at odds with the whole team from different organisations working together for the good of the team and the long term goals of the project. It is very important therefore to understand what each participant wants to gain from a project. For example the domain scientist is often a number of years, up to 10 being suggested, behind the computer scientist and this is also true of mathematicians where often the algorithms being implemented were derived many years previously. When involved together on a project it is important to understand that the computer scientists and mathematicians will be looking to develop techniques at the forefront of their field that the domain scientists on the project might not wish to use until they are mature. There needs to be a

drive to ensure that the computer scientist's and mathematician's work on a project is directly relevant to the scientific aims of the project.

Too many people feeding in project requirements can, not only arise from conflicting agendas, but also miss understanding and whilst a diverse set of people will produce more project requirements than a more traditional team there must be some control of these. It is also important to understand that requirements from one group of people can have an unforeseen impact upon others so all stakeholders must be involved. For instance hardware designers might wish to design the architecture in a specific manner that, unbeknown to them, makes it much more difficult for the software engineers to program. At the end of the day the project aims must be realistic.

Another potential worry some of the participants had is the cost implication. Setting up dedicated co-design centres is an expensive task and for instance the US DoE only funded a number of their initial centre proposals which are not co-located.

Another aspect that was mentioned a number of times is that, depending upon the exact nature of the project, not everybody will be required at the same stage. It can often be the case that as the project matures the requirement for specific skills does change and this staff turnover can lead to a loss of knowledge. Often people think of the idealised model of co-design whereby everybody is working concurrently and there is feedback between the groups, but in reality this is not always possible where, for instance, some hardware development must be done before the software side of things can gain momentum. It might be possible to develop tools and processes to alleviate some of this scheduling but that might be unattainable for smaller projects.

3.2.5 What needs to be changed to make adopting co-design more attractive?

During our discussions with the interviewees there were a number of strong feelings about how things could be done differently to improve collaboration. As discussed in section 3.2.4 communication is a major issue and it is important for the whole team to be working towards a common goal, therefore human factors need to be considered. The point was made that education is a critical enabler between different groups of people with an observation being that mathematics courses are increasingly becoming more pure with less of a focus on computation and in computer science courses the harder maths content is being removed. Without this overlap between the disciplines then it is far more difficult for one to work with another where, for instance, the mathematicians might have very little understanding of the underlying computation concerns and the computer scientist unable to broadly understand the requirements of the mathematicians.

There is a famous quote ``the whole is greater than the sum of its parts'' and this can be said about the co-design methodology. There was the worry that at an institutional level there is no appreciation of this and it can be difficult for interdisciplinary teams to exist within the rigid structure of an organisation. As an example of this current funding applications can often be targeted towards specific areas and even if a project is split up then the application will often be considered in terms of these individual areas rather than taking the high level view that it is the combination that results in something special. A comment was also made that PhD students working as part of co-design teams can sometimes be put at a disadvantage in some countries because their institutions and examiners might expect their work to be in one specific area but the whole co-design process has naturally spread it out and the student as not specialised as much in a specific area as might be expected.

Along the lines of breaking down rigid structures at an institutional level, another comment made was that there should be greater incentives for early career researchers to collaborate and form co-design teams rather than close in on the own discipline. From discussing this point with a number of different participants it was interesting to see that in some areas collaboration and interdisciplinary teams is very much encouraged, but in other fields and institutions is was far less so and one participant even

felt that it might harm the prospects of an early career researcher if they did not focus in on their own discipline. The point was also made that, sometimes European focus is very much application driven, with this driving all other aspects such as the underlying middleware and hardware. Many of the European funding calls reflect this focus and, whilst this works well in some cases, in others it might not be ideal. It was the opinion of the participant that the funding calls should be oriented around doing science and all the other areas, such as applications and hardware development, be driven primarily by that concern.

3.2.6 How does co-design compare against centres of excellence?

Especially if money is invested then a major concern is the best approach to adopt, in order to get the maximum benefit, when aiming for the exa-scale. Whilst co-design is often mentioned as an important enabler (and sometimes a silver bullet) others push the advantages of centres of excellence. It was therefore an important question to ask our participants about centres of excellence and the differences between centres of excellence and co-design centres. We had some interesting responses and many of the participants asked us to define the term centre of excellence as they did not really understand how one term was any different to the other. The definition we adopted was a team who are very specialist in a specific area and act as a central point of contact and dissemination for their knowledge, an example of this is the Cray centre of excellence at Edinburgh Parallel Computing Centre (EPCC.)

The most common response was that these terms are not mutually exclusive and co-design and centres of excellence need to work together if we are going to meet the exa-scale challenge. One participant stated that, in their opinion, co-design is an approach and a centre of excellence is a pool of people who can be used to provide some expertise as part of the co-design team. Co-design requires people to be flexible and the benefit of a using people from a centre of excellence is that it can keep team members busy on other tasks if the co-design project does not require them at a specific point. One participant voiced the opinion that an additional benefit of co-design is that all parties are exposed to a wide variety of advancements in their field and the worry with a centre of excellence is that the staff might get left behind as technology advances. With such rapid technological changes, if you close people in too much then they can lose sight of advancements happening in other but highly relevant areas. For instance software developers, if they are not careful, might not have sight of new and upcoming architectures which they are then expected to write code for all of a sudden when these mature. The comment was made that, to make the most efficient use of exa-scale resources it requires a team with fresh and dynamic ideas; as co-design brings together a diverse set of individuals this acts to encourage these traits.

One respondent clearly stated that, in their opinion co-design involves hardware and software interdisciplinary teams, whereas centres of excellence are more much oriented around the software side only. Another respondent stated that, in their opinion, these two things were basically the same in practice and it is often a political choice that determines which phrase is used. In their opinion, regardless of the specific phrase, the most important thing is that the role and aims of the centre are clearly defined and kept in focus. Another comment made was that if one wished to create a European Exa-scale centre then the best situation would be to find a middle ground between centres of excellence and co-design where one combines the benefits of both resulting in the best of both worlds.

3.3 Existing approaches

In section 3.2 we detailed the results of in depth interviewing members of the HPC community about co-design and exa-scale that have been gathered. An important are the concerns and considerations that our pool of interviewees believe a project must take into account when adopting the co-design process. Co-design has used effectively already by a number of high profile projects and it is important to consider not only the model that these have followed but also the mitigations put in place to alleviate any of the potential issues that might arise from using this development approach.

3.3.1 Application proxies

A modern HPC application can be a huge and complicated beast; over one million lines of code is not uncommon and this can make it very difficult for those with expertise in other areas, such as hardware developers, to then work with these applications to test and validate their own work. One approach would be to have a team of software engineers on hand to port these applications to a variety of different architectures, but this would still be hugely labour intensive and would require excellent communication and understanding between the hardware and software experts on a team. Instead an application can be cut down and it is often possible to remove much of the application complexity and third party library dependencies whilst maintaining the attributes which have the real impact upon an application's scalability and performance. When testing and developing systems there are two extremes of codes which might run; benchmarks at one end which are as simple and concise as possible to test specific attributes of a system and full applications at the other end. In the middle somewhere are these application proxies which contain both the performance intensive computations of a large-scale application and also contain the context of those computations.

[5] introduces a number of different levels of application proxies which can then be selected depending upon the exact work one wishes to do and these are summarised in Table 1. It can be seen that there a number of different types of application proxy and the ideal one to choose depends upon the project and team members.

Surrogate	**Description**
Compact application	Small application with fewer features and simplified boundary conditions
Mini application	Small self-contained program that embodies essential performance characteristics of key code
Skeleton application	Captures the control flow and communication pattern of an application - can only be run in a simulator
Mini driver	Small programs that act as drivers of performance impacting libraries
Kernel	Captures node level aspects of an algorithm

Table 1 - Categories of application proxy

Mini applications

Whilst a mini-app is only one level of application proxy, its popularity has meant that this term has become synonymous with simplifying an application to produce a clear and concise code which accurately maintains the performance characteristics of the original. The Mantevo [6] project are driving the development of important mini-apps which can be used to represent their full-fat counterparts. Mini-apps take advantage of the fact that often the performance of an application is dominated by a small subset of the code and whilst the physical models are mathematically distinct, they often share common performance characteristics. They do this by encapsulating only the most important computational operations and consolidating physics capabilities that have the same performance profiles to produce a lightweight independent executable that will run on a system. The large-scale application developer, who is tasked with developing the mini-app, guides the decisions, resulting in a code that is a small fraction of the original application size, yet still captures the primary performance behaviour.

As [6] details, using mini-apps results in a number of benefits:

- **Interaction with external research communities:** Mini-apps are open source software, in contrast to many production applications that have restricted access.
- **Simulators:** Mini-apps are the right size for use in simulated environments, supporting study of processor, memory and network architectures.

- **Early node architecture studies:** Scalable system performance is strongly influenced by the processor node architecture. Processor nodes are often available many months before the complete system. Mini-apps provide an opportunity to study node performance very early in the design process.
- **Network scaling studies:** Mini-apps are easily configured to run on any number of processors, providing a simple tool to test network scalability. Although not a replacement for production applications, mini-apps can again provide early insight into scaling issues.
- **New language and programming models:** Mini-apps can be re-factored or completely rewritten in new languages and programming models. Such working examples are a critical resource in determining if and how to rewrite production applications.
- **Compiler tuning:** Mini-apps provide a focused environment for compiler developers to improve compiled code

There are a number of approaches to writing mini-apps from starting with the existing application and cutting it down, to starting from scratch and independently writing a simple application which behaves like the complex one. Unsurprisingly the Mantevo project has found that the best person to develop a mini-app is the one who developed the main application and this project encourages creating mini-apps as part of the main application programming life cycle. The project emphasises keeping not only the code but also the build environment simple and for each mini-app to produce similarly formatted output that tools have been developed to visualise. This approach has been found to greatly help other people in the co-design team who can quickly get these simple standalone codes running and use them as required especially due to the standardisation between mini-apps.

[6] details the number of mini-apps that the Mantevo project has developed which are open source codes, and cover a wide range of application spaces. Using mini-apps has demonstrated to very much assist the US Department of Energy co-design teams as the software engineers can hand this simplified representation of their applications over to other members of the team and then these people, who many not be development or application experts, can then use them to develop, test and validate other areas. In section 3.2.4 it was mentioned that communication between interdisciplinary teams might be a difficulty, and in a way mini-apps are a great way of aiding this. By greatly simplifying all aspects of an application, from its code to its build system and output then non software or domain experts in the team can pick these up and use them with minimal effort.

3.3.2 US Department of Energy co-design centres

To ensure that future architectures are well-suited to the US Department of Energy's (DoE) goals and that their target applications and scientific problems can take advantage of future exa-scale systems, the DoE have funded three co-design centres. These have a broad remit to address the hardware, software, numerical methods, algorithms, and applications in a specific area that is seen as a grand challenge for the DoE. These three centres concentrate on solving domain specific problems, and they are the Exascale Co-Design Center for Materials in Extreme Environments (ExMatEx) [7], the Center for Exascale Simulation of Advanced Reactors (CESAR) [8] and the Center for Exascale Simulation of Combustion in Turbulence (ExaCT) [9].

ExMatEx

The objective of the Exascale Co-design Center for Materials in Extreme Environments (ExMatEx) is to establish the interrelationship among algorithms, system software, and hardware required to develop a multi-physics exa-scale simulation framework for modelling materials subjected to extreme mechanical and radiation environments. Such a simulation capability will play a key role in solving many of today’s most pressing nuclear problems such as the production of clean energy, extending nuclear reactor lifetimes, and certifying the aging nuclear stockpile.

To achieve their goals, ExMatEx concentrate their research and development effort on three key areas

- **Programming models:** The centre's target application is composed of various components that must interact with each other in a dynamic and adaptive fashion as the code runs. There

is the additional complexity that their code may cache previously computed results in a database to improve efficiency. No single programming language exists that supports all of the required functionality and as such multiple languages and systems must be combined to build the final application. This is a common issue faced in HPC, where one language or technology might suit one aspect of a large application but not other areas. Language designers have worked with other expects, such as compiler writers, to develop a domain specific language (DSL) infrastructure which can be used as the foundation for application programmers to build their own DSLs with minimal fuss.

- **Runtime systems:** Traditionally much of the emphasis has been placed upon the programmer to explicitly write code handling common HPC tasks such as load balancing, caching and fault tolerance. Not only is this duplication of code wasteful, it also requires the programmer to consider low level, uninteresting and often complex aspects of parallelism rather than to concentrate upon solving science. Computer Scientists are working with low level experts in the ExMatEx team to look at improving runtime support and libraries so that many of the current considerations that an HPC programmer must address are implicit.
- **Analysis, modelling and simulation:** No single analysis technique is powerful enough to provide the necessary insight into application code that is needed to drive the co-design process. Instead the techniques of analytical modelling, architectural simulation, system emulation and empirical measurements are combined to create a detailed view of the behaviour of an application. The team use existing toolkits and models to understand a variety of different factors such as power efficiency, memory usage and fault tolerance.

ExMatEx place emphasis on each of the three key areas being developed together and feedback from one improves another. For instance developments in the programming models can drive aspects of the runtime systems but lessons learnt here also feedback and determine what support is required in a programming model and how it might best interact with the underlying system. The analysis, modelling and simulation tools might be developed based upon the requirements from the other key areas and the lessons learnt feedback to improve these areas. Much of this work is based around mini-apps that the team have developed to accurately represent the facets of their target application.

CESAR

The ultimate goal of the CESAR co-design centre is the developing a next-generation nuclear reactor core simulation tool (TRIDENT) capable of efficient execution on exa-scale platforms which will help inform the design, licensing, and safety analysis of next-generation nuclear technology. To achieve this aim, the centre is placing a particular focus upon the algorithms that underlie the high-fidelity analysis of nuclear reactors and it is looking to drive architectural decisions and adapt algorithms to the next generation HPC computer architectures. The co-design process is focused around three classes of algorithms each of which represent a significant aspect of a future reactor simulator tool.

The co-design methodology that the CESAR centre has adopted is oriented around a cycle that couples algorithm development, performance modelling, hardware design and hardware simulation. The governing physics of a nuclear reactor can often be reduced down to more fundamental algorithms, each of each stress different aspects of the computer architecture. Due to the complexity the team have started working with mini-apps and are looking at the fundamental questions for each algorithm such as the optimal memory hierarchy, what instruction support is required to decompose the problem on SIMD architectures and the network performance required to run these codes at exa-scale. The team expect that significant redesign of the algorithms will be required to reach exa-scale and as the CESAR application matures then it will drive key decisions about the software and hardware support required.

Due to the complexity of modelling a nuclear reactor, the co-design process adopted has instead focused at the interaction between the algorithms and the machine architecture such that both are developed together.

ExaCT

The aim of the ExaCT centre is to obtain new understanding about combustion and as such they are looking to update their software models with changes necessarily for exa-scale and ensure that

hardware is developed to meet the DoE's requirements of performing real world combustion computations. ExaCT looks to iteratively co-design all aspects of combustion simulation including the mathematical algorithms for partial differential equations, programming models, scientific data management and analytics, and architectural simulation to explore hardware trade-offs with combustion mini-apps representing the workload of an exa-scale system.

One of the successes of the ExaCT centre is the Exascale Static Analysis Tool (ExaSAT) framework [10] which enables the team to rapidly explore the effects of code optimization upon the performance of a target application in the context of varying hardware details. This framework is based upon static analysis and generates performance models directly from the source code without requiring programmer intervention. The work has been able to demonstrate tuned hardware and software configurations that achieve up to a 90\% reduction in memory traffic through deep code analysis and performance modelling which have been fed back to vendors. In [10] the authors explicitly state that this work demonstrates the utility of a co-design approach.

3.3.3 US Department of Energy co-design process

Whilst the three US DoE co-design centres discussed in section 3.3.2 are aimed to solve different scientific challenges, it is clear that whilst the computational characteristics of their work might be different, they are adopting similar approaches and there is the potential for overlap. Other projects have grown around these centres to provide common components and help share the developments of one centre with that of another.

DoE mini-apps

Each of the three DoE co-design centres have developed their own mini-apps which relate to the area that the centre is investigating. Whilst these co-design centres are separate, it is still important that work is not duplicated and if one centre has developed a mini-app then it should be available to another and ideally the wider community. One of the jobs of the Computer Architecture Laboratory (CAL) [11] is to catalogue and identify the computational and communication characteristics of these mini-apps. Traces of the MPI communication for each mini-app have been collected and published along with performance and speed up data. From the work done by the CAL project, a co-design team member can in theory pick up a mini-app off the shelf and understand not only what it does but also its runtime characteristics.

CoDEx

To support their co-design laboratories the DoE have assembled a hardware and software co-design environment that they call Co-Design for Exascale (CoDEx) [12] to fully integrate the software development together with the hardware side. This CoDEx environment is a fundamental aspect of many of the US DoE co-design activities that have been discussed in this section. There are three main components that make up the environment:

- **The RAMP platform** [13] which provides a configurable and cycle-accurate, validated node design to understand the impact of architecture choices on application performance. This supports rapid synthesis of many hardware facets such as CPU designs, estimating power consumption and benchmarking of applications using hardware accelerated models of the resulting node design. This system uses FPGA based hardware emulation which is a fast and cost effective way of prototyping and running hardware implementations at near their real time speeds.
- **The ROSE compiler framework** [14] is used to enable rapid development of source code translators to facilitate detailed analysis of complex application codes and code transformations. This is critically important because the ROSE analysis tools provide architecturally relevant parameters to inform chip designers where to focus their attention, such as byte-flop ratios, on-chip cache-size requirements, maximum extractable parallelism for a kernel, and data flows required for inter-processor communication. This tool can automatically reduce the complexity of an application whilst maintaining important attributes such as the communication behaviour. Traditionally an expert would have to extract an application proxy from the main application, reducing the complexity but being careful to keep

the aspects which impact scalability and performance. The ability to automate this extraction is very important and enables far more applications to be tested and examined.

- **Sandia Laboratories' Structual Simulation Toolkit (SST)/macro** [15] allows one to model networks substantially larger than today's peta-scale machines. Construction of even medium sized network for experimentation is prohibitive due to cost therefore, instead architects have access to simulation tools that accurately model large-scale behavior. The SST/macro discrete event simulator enables the evaluation of large- scale interconnect designs that have millions of client endpoints. This simulator can be used to either replay traces of previously run MPI applications, such as the mini-app traces produced by the CAL project, so that the execution time can be estimated on new architectures and/or simulate skeleton applications (a type of application proxy) to understand their likely behaviour.

Uses of the CoDEx project have resulted in a number of significant results and developments in the DoE's work. Of most interest in this report is how it might be used to help mitigate any of the perceived negatives of co-design. In [5] they estimate that the conventional design cycle lasts around six years; two years to design a new system, two years to actually build the hardware and then two years to port and tune applications. These timescales highlight two major issues of combining co-design with other traditional development approaches. Firstly the fact that hardware engineers are required initially and then, once the architecture is built, software experts are required instead so the skill set shifts. Secondly the co-design process is meant to involve feedback but over such long timescales this is not possible especially when the design has been fixed some years before. As a revolutionary change, the CoDEx environment cuts the timescales down from years to one or two days and using these tools co-design teams can synthesize their design and emulate the hardware in a matter of hours with the application porting and tuning completed in a matter of hours too. Central to these greatly increased timescales are the hardware RAMP and SST tools and the ROSE compiler software to automate and simplify much of the process. In section 3.2.4 our interviewees worried about co-design timescales and how as the project progresses there is a shift of skills required and knowledge can be lost, from [5] it is obvious that CoDEx enabled DoE co-design teams to be working on hardware and software at the same time and actively feeding back, where changes to the hardware can be actioned immediately and the results available in hours to be tested against the software.

3.3.4 CRESTA

CRESTA [16] is a major European project bringing together leading supercomputing centres, equipment vendors, programming tools providers and six application and problem owners to explore how the exa-scale challenge can be met. It is very much application focused with a major aim being that their work will enable new science, which was previously unattainable, to be performed. The project has two integrated strands, the first focuses on enabling a key set of co-design applications for exa-scale, the other oriented around building and exploring appropriate system ware for exa-scale platforms and creating a foundation for future applications to make use of exa-scale systems. The six applications involved are known as co-design vehicles and represent a wide range of codes used by European academia and industry to solve critical grand challenge issues. These include bio molecular systems, fusion energy, the virtual physiological human, numerical weather prediction and engineering.

CRESTA is fundamentally different from some other co-design efforts, such as the DoE, in the respect that there is no hardware aspect to the project. The second integrated strand, about building and exploring appropriate system ware does involve a variety of individuals working on lower level technologies that underpin these applications such as experts in the fields of operating systems, middleware, programming languages, compilers and library development bit their remit is purely software based. Performance indicators are a critical part of any project, and as a showcase for co-design these can help give us some indication as to the success of using this approach. Amongst many improvements to the six software applications that have been demonstrated, the project has also very successfully explored and validated a number of middleware designs which have fed into vendor road maps.

One of the main reasons that the CRESTA project has been so successful is that all partners actively engage and have the desire to be involved. Whilst there is a robust management structure in place across the project, much of the interaction has been self-driven and this illustrates the good rapport that different partners have. Early on in the project it was understood that, for an interdisciplinary team to work successfully, there must be a common goal that all members are working towards. Fundamentally, people need to be signed up to this end goal and understand that, in the short term, they may be required to do tasks which have no specific benefit to them immediately but this is for the greater good. CRESTA also benefits from the fact that all partners have software development knowledge and experience, many of the domain scientist involved have written or are heavily involved in writing the applications that the project is focusing upon. This makes the understanding within inter-disciplinary teams much more straightforward than some project setups might encounter.

The use of mini-apps is popular in US DoE co-design centres. Whilst mini-apps are used in CRESTA, and have been of some benefit to middleware experts, the focus of the project is upon the applications themselves. As such mini-apps are viewed as a part of the process, and often a good starting point, but the main focus is crucially always upon the main application. This is a fundamental difference between the CRESTA approach and the US DoE approach which places great emphasis on the value and use of mini-apps as their focus is more on the development of other aspects driven by the application. In CRESTA co-design teams cross cut the work packages, such that if an expert in one area is working on a specific technology and an expert in another area on something different, then these teams can take the high level view and figure out how the work that one person is doing might be of benefit to another rather than people being confined to their own specific work package silo.

3.3.5 DEEP

The DEEP [17] project aims to develop a novel architecture that is a demonstrator for technology capable of reaching the exa-scale. This project combines the hardware development aspect with software, where the team are looking to develop a novel software stack and set of applications that are optimised for the architecture. On the hardware side, DEEP follows the approach of complementing conventional HPC systems with an accelerator cluster to increase the overall performance. The software stack comprises of adapted programming models, libraries and associated tools.

This is an excellent example of hardware and software co-design; where the architecture, middleware and target applications are being developed and optimised together.

3.3.6 MONT-BLANC

The MONT-BLANC [18] project is designing a new type of hardware architecture targeted towards energy efficiency. As noted in section 2, energy efficiency is a major challenge for, not only future exa-scale machines, but also at the current peta-scale. By addressing the entire HPC life cycle, from hardware to middleware to the applications themselves, MONT-BLANC aims to develop and test strategies for optimising power efficiency. As this project covers a wide spectrum it involves many parties such as hardware designers, vendors, industry partners and application developers.

4. Conclusions

In this report we have surveyed a number of existing co-design for exa-scale approaches, discussed aspects of using co-design with members of the community and analysed how people in the community view and use co-design currently. It is clear that, on the road to exa-scale there is no silver bullet and one methodology by itself will not be enough. Having said this, co-design has had a positive impact within the US DoE and other projects, such as the EU CRESTA project, have adopted similar approaches.

From the projects surveyed, it is clear that there are a wide variety of co-design implementations and different people have their own definitions. Regardless, it is critically important that if a project is to adopt co-design then it not only defines what its definition of the methodology is, but also ensure that all team members have a clear and common goal to work towards. It seems that the best co-design teams are comprised of individuals who appreciate that they need to work towards the *common good* and at times they might be doing tasks that have a long term benefit rather than a short term incentive.

4.1.1 Recommendations

Based upon the information assimilated, there are a number of recommendations to the use of co-design that would like to make in this report.

- Whilst there are a variety of co-design implementations, at its core the methodology is about different disciplines working together to solve a specific problem. A single definition of co-design should be adopted and agreed upon ideally at a European level. Exactly what parties should be involved depends entirely upon the specifics of the project and its goals; whilst some projects will combine hardware experts with software experts, others are far more subtle and involve collaborations between those with similar backgrounds but in different technical communities who do not traditionally work together.
- Crucially important to any co-design team is that the project goals are well defined and each member understands that long term goals differ from short term gains.
- As co-design is adopted by more and more European projects there should be some process to ensure that these do not reinvent the wheel either between themselves or compared with the US DoE centres. This report has considered some of the supporting DoE projects such as CoDEx that have grown up from the understanding that, whilst the DoE co-design centres concentrate on different scientific challenges there is still plenty of technical overlap. By sharing aspects such as environments and mini-apps between co-design teams will only help to accelerate development towards exa-scale.
- There should be some official standardisation published for European co-design teams to adopt in order to make collaboration between these teams and team members as natural as possible. An example of this standardisation is the common output format adopted by the Mantevo project for their mini-apps.
- Co-design can be adopted around the existing organisational structures and locations that people exist within although it might be useful to push some institutions to recognise that working as part of an interdisciplinary team is critically important for the HPC field.
- The results of the co-design process should be defined in terms of science rather than hardware or software oriented. Ideally science would be the driver to develop improvements at all stages of the HPC life cycle.

5. Bibliography

[1] Exascale Mathematics Working Group, “Applied Mathematics Research for Exascale Computing,” *DOE ASCR technical report,* 2014.

[2] P. Kogge, “Exa-scale computing study: Technology challenges in achieving exascale systems,” Technical report of the University of Notre Dame - Department of Computer Science and Engineering, 2008.

[3] N. Brown, “A questionnaire about co-design experiences and opinions,” [Online]. Available: http://www.surveymonkey.com/s/9KWPWKZ. [Accessed 30 May 2014].

[4] L. Planna, “SpiNNaker: Design and Implementation of a GALS Multi-Core System-on-Chip,” *ACM Journal on Emerging Technologies in Computing Systems,* vol. 7, no. 4, pp. 1-18, 2011.

[5] D. Q. C. J. J. Shalf, “Rethinking Hardware-Software Codesign for Exascale Systems,” *Computer,* vol. 44, no. 11, pp. 22-30, 2011.

[6] M. Heroux, “Improving Performance via Mini-applications,” Computer Science Research Institute, Sandia National Laboratories, 2009.

[7] T. Germann, “Co-Design Centers in the US: Exascale Co-design for Materials in Extreme Environments,” in *International Workshop on codesign*, 2013.

[8] US Department of Energy, “The CESAR Codesign Center: Early Results,” in *DOE Exascale Research Conference*, 2012.

[9] US Department of Energy, “ExaCT: Center for exascale simulation of combustion in turbulence,” [Online]. Available: http://exactcodesign.org. [Accessed 05 03 2014].

[10] D. U. M. L. W. Z. J. B. J. S. C. Chan, “Software Design Space Exploration for Exascale Combustion Co-design,” *Lecture Notes in Computer Science,* vol. 7905, pp. 196-212, 2013.

[11] US Department of Energy, “Characterization of the DOE Mini-apps,” [Online]. Available: http://portal.nersc.gov/project/CAL/doe-miniapps.htm. [Accessed 20 05 2014].

[12] D. D. C. J. H. A. D. Q. J. Shalf, “CoDEx: CoDesign for Exascale Architectural Simulation and Modeling for Exascale Platform Development,” [Online]. Available: http://www.nersc.gov/projects/CoDEx. [Accessed 20 05 2014].

[13] Z. Tan, “RAMP Gold: An FPGA-based architecture simulator for multiprocessors,” *47th ACM/IEEE Design Automation Conference,* p. 463–468, 2010.

[14] D. Q. M. Schordan, “A source-to-source architecture for user-defined optimizations,” *Lecture Notes in Computer Science,* vol. 2789, pp. 214-223, 2003.

[15] C. Janssen, “A simulator for large-scale parallel computer architectures,” *International Journal of Distributed Systems and Technologies,* vol. 1, no. 2, p. 57–73, 2010.

[16] The CRESTA project, “CRESTA: Developing techniques and solutions which address the most difficult challenges that computing at the exascale can provide,” [Online]. Available: http://cresta-project.eu. [Accessed 30 05 2014].

[17] The DEEP project, “DEEP: Dynamic Exa-scale Entry Platform,” [Online]. Available: http://www.deep-project.eu. [Accessed 25 05 2014].

[18] The MONT-BLANC project, “MONT-BLANC: European approach towards energy efficent high performance,” [Online]. Available: http://www.montblanc-project.eu/. [Accessed 22 05 2014].

