

# Software Engineering Support for UK Computational Science Community

Dr C Greenough and Dr MF Guest  
*CCLRC, CSE Department*

<b>1. INTRODUCTION .....</b>	<b>2</b>
<b>2. OBJECTIVE .....</b>	<b>2</b>
<b>3. DISSEMINATION &amp; DEPLOYMENT.....</b>	<b>2</b>
<b>4. COLLABORATIONS &amp; EXCHANGE.....</b>	<b>3</b>
<b>5. THE SES PROGRAMME .....</b>	<b>3</b>
5.1. SOFTWARE QUALITY ASSURANCE .....	5
5.2. PROCESSES FOR LEGACY SOFTWARE.....	5
5.3. EVALUATION OF METHODOLOGIES, TOOLS AND TECHNOLOGY .....	6
5.4. SYMBOLIC ALGEBRA SYSTEMS.....	8
5.5. PROBLEM SOLVING ENVIRONMENTS (PSE) .....	8
<b>6. COMMUNITY &amp; VENDOR PARTICIPATION .....</b>	<b>9</b>
<b>7. DISSEMINATION OF SOFTWARE AND RESULTS.....</b>	<b>9</b>
<b>8. RESOURCES .....</b>	<b>9</b>
<b>9. FUTURE PROGRAMME FOR SES .....</b>	<b>10</b>
<b>10. SOME RELATED UK &amp; US ACTIVITES.....</b>	<b>10</b>
10.1. UK GROUPS AND RESOURCES .....	10
10.2. US GROUPS AND RESOURCES .....	10
10.3. GENERAL/EUROPEAN .....	11

## **1. Introduction**

The Computational Science and Engineering Department (CSED) of CLRC has been involved in the collaborative development of scientific software for many years. The Collaborative Computational Projects (CCPs) and High End Computing (HEC/UKHEC) activities are driving the major strands of this activity.

The Department has maintained a watch of software engineering technology and encouraged the use of state-of-the-art techniques within its projects. It has also disseminated its experience through the activities of specific science projects.

Over the past few years the importance of good software design and implementation practices in science is beginning to be recognised by the UK scientific community and a number of small initiatives are being taken. CSED has been growing this type of activity during the recent past and believes it is now the time to make a major commitment to promoting advanced software engineering within the computational science community.

Because of its history in computational science through the CCP and HEC activities, CSED is well placed to act as a focus for such a Software Engineering Support Programme (SESP).

## **2. Objective**

The main objective of this activity within the CSED Core Programme is to investigate the effectiveness of software engineering technologies when applied to the development of applications software in computational science projects. The programme will provide initially software-engineering tools and expertise to the CCP programmes and in the future more generally to the UK computational science community. In addition SESP will provide software engineering assistance to the computational science projects as well as conducting educational and awareness workshops.

The main goals of this proposed SES Programme are:

- accelerate the introduction and widespread use of high-payoff software engineering practices and technology by identifying, evaluating, and maturing promising or underused technology and practices;
- maintain a long-term competency in software engineering and technology transition;
- enable the UK academic community to make measured improvements in their software engineering practices by working with them directly;
- encourage the adoption and sustained use of standards of excellence for software engineering practice;
- foster collaborations with other groups, in the UK, Europe and the US, that have an interest in the applications of advanced software engineering techniques in computational science.

These goals will improve the level of software engineering practice within UK computational science research groups. As a result, the software they develop will be of a higher quality; more easily developed and maintained; more easily re-used within the community and be computationally more efficient.

## **3. Dissemination & Deployment**

General dissemination of the projects of SESP is covered in Section 7. Here we note how the tools and processes gathered by SESP will be introduced into CSED's scientific software development programmes.

There are a number of threads in SESP ranging from tools to verify language conformance to tools to aid full scale re-engineering. The impact on the development project is clearly dependent on which tool or process is being adopted. Although much of the information flow between SESP and the science projects will be through web information pages and seminars/workshops, SESP will identify two or three *Flagship* projects in collaboration with the science groups.

For example, as part of CCP5 (Computer Simulation of Condensed Phases), we envisage a project to completely re-engineer DL\_POLY to use all the features of Fortran 90/95. The degree of software engineering tools to be used could range from just simple transformation tools to the adoption of a complete IDE with CASE tools. SESP and CCP5 will therefore establish a short collaborative project to introduce and use the SESP tools, thereby enabling CCP5 to continue using the tools in future developments.

CSED will review the SESP programme after its first year in operation. If this review is positive, CSED intends thenceforth to commit to use SESP methodologies in all its major software development projects. With this commitment as a platform, CSED will develop ways of offering SESP methodologies to scientific applications developers in the CCPs and the wider computational research community.

#### **4. Collaborations & Exchange**

Within the UK community there are a number of groups interested in software re-engineering and software quality. However the majority of this work is directed to C++ or Java. Fortran is not seen as a great priority. Section 10.1 gives some web links to activities ongoing in the UK related to the proposed activities of SESP. There is a large variety of software engineering research in the UK and Europe however much of the transformation and re-engineering activity is directed to complex business systems and not toward computational science applications and Fortran. SESP will develop links with the leading European software engineering groups working on methods for high performance computing. CSED already has some contacts to institutions such as CWI in Holland, INRIA in France and CNR in Italy through ERCIM.

Software Engineering is an area of great activity within the US and organisations, such as the NSF with the *Software Engineering Research Center* focused at Ball State University and NASA with the *Fortran Modernization Project* among many, have made significant investments. SESP will develop specific software engineering links with NSF laboratories with which CSED already has Memoranda of Understanding (NCSA, PSC and SDSC). SESP will also seek to develop links with the teams in NASA and other US programmes that have active work in this area. In particular links will be sought with the groups at PNNL and the Computational Sciences and Engineering Division at ORNL. Part of this process will include attending the NASA/IEEE Software Engineering Workshop.

Section 10.2 lists some of the current and recent US activities from which SESP will draw on and develop collaborations. CSED will aim to cement such collaborations initially by organising an “N+N” meeting to share software engineering practices among those developing the most challenging scientific applications in Europe and the USA. This, we hope, will nucleate an international “Scientific Software Engineering” community.

#### **5. The SES Programme**

In the following sections we describe the main themes of activity the programme will undertake. Each section gives some of the background to the theme and a list of proposed objectives. In some themes the EPSRC SLA is already funding some activity: for these the

current level of effort supported is indicated in Table 2. In general the scope of the current activity has been broadened with additional deliverables.

In the following paragraphs we emphasise the need to maintain a good awareness of the current trends in software engineering particularly as applied to software developments in computational science and engineering. The gathering of this information will enable CSED to take on broad practices and tools as appropriate. These initial steps have been characterised by the *Technology Watch, Assessment & Evaluations* process. Although the software engineering community has various very formally defined processes of *Software Assessment & Evaluation* we have defined below a rather more pragmatic approach for SESP.

*Technology Watch:* In each elements of the SESP information will be gathered on a regular basis and a rolling update made to a *Technology Report*. The primary sources of information will be:

- *The Internet:* the web sites of known groups and software vendors will be monitored and general searches performed.
- *Workshops/Conferences:* there are many workshop and conferences on the development of software engineering. Although many of these are aimed at blue-sky activities there are often associated software fairs where vendors present their latest tools. Example events that might be considered are the IEEE/NASA Software Engineering Workshop or IEEE International Conference on Software Maintenance - ICSM.
- *Commercial Vendor Conventions:* There are a large number of events organised by the commercial world on software tools. The TestExpo, organised by Q-Bit Limited, is a major UK venue for most of the major commercial tool vendors.
- *Direct Contacts:* SESP would gather information directly from the software vendors where thought appropriate. This would certainly happen semi-automatically with vendors whose tools are adopted.

All the information would contribute to a technology watch report that would be made available to the community through the SESP Web site.

*Assessments:* The starting point of selecting a tool for use in anger would be through paper assessment using a basic requirements document. The detail of the assessment would clearly depend on the area being addressed but there would be a collection of fundamental requirements such as operating systems, supported languages etc. developed by SESP. The applications Groups within CSED will be involved in developing and extending the requirements document for their own particular areas. These paper assessments would identify tools for practical evaluation. Much of the material developed in the paper assessments would be added to the technology watch reports.

*Evaluations:* Through the assessment, various tools will be selected for more direct evaluation. In discussion with the vendors, evaluation or some form of limited license would be agreed and the tools installed. They would be used in a realistic context either by SESP staff or those involved in the CCP and HEC programmes and their usefulness and effectiveness documented. Although in general the evaluations would not be placed on critical paths within the CCP or HEC activities, these programmes provide a considerable number of representative software packages that can be made the subject of an evaluation. The evaluations would lead to detailed reports and if successful the deployment of the tool or practice within the main stream.

### 5.1. *Software Quality Assurance*

Software Quality Assurance is the basic of software engineering processes that should be undertaken by all software developers. However the majority of codes in use and being developed within the community have not been through any process of QA save that of the compiler being satisfied.

The target language for most applications is now Fortran 95 or even Fortran 2000. Although the commercial world of Software QA is dominated by C, C++ and Java, there are good Fortran tools available. PlusFORT, ForCheck and the NAG Ware are but three examples. CSED has experience in using these tools and are developing web and GRID based interfaces to a collection of them.

The tasks for this activity are:

- Keep a technology watch on QA tool development
- Perform tools assessment on typical community applications
- Make tools available through suitable interfaces
- Provide web based documentation on tools

Part of this activity is already funded under the current Facilities Agreement (see Table 2).

### 5.2. *Processes for Legacy Software*

For many applications within the science and engineering community the root language has been Fortran 77 and for some - even Fortran 66. Software engineering has developed and languages have grown and now Fortran 95 and C provide the main modern vehicles for these applications. However we will make the assumption that the majority of the software of interest is in Fortran (of some form) and that the target language is Fortran 90/95.

To maintain and continue to develop the science encapsulated in these *legacy* codes a process of *transformation* and *re-engineering* must be formalised. This can be broken into three basic steps: standardisation, transformation and re-engineering.

*Standardisation:* As mention above often legacy codes are in Fortran 77 or 66 or even worst a mixture of standards and dialects. In general the standard of research programming is limited and most often not particularly portable. The codes often adopt mechanisms from other languages. The main example of this are the # directives from the Unix C pre-processor cpp - #include and #def are but two. To aid there transformation process these will need removing and documenting.

*Transformation:* Once the basic code is in a standard form automated transformation tools can be used to change the software format (e.g. fixed-form Fortran 77 to free-form Fortran 95). This is purely a source to source transformation and no structural changes are made. However it maybe thought necessary that the original legacy code be maintained for some compatibility reasons. If this is the case this code can be *wrapped* in some appropriate statements that will enable the legacy code to be called directly.

*Re-Engineering:* Within Fortran 90/95 there are many features that will improve the quality of an applications program in performance, maintenance and development potential. Examples are: modules, derived data types, dynamic memory management, pointers and allocatable arrays. This is the most difficult part of the process as it may require considerable re-writing of the software.

One of the major difficulties with this process is *author recognition* - the originator or current developer of the code no longer recognises the software and hence is reluctant to use the newer version as the basis of future development. The starting point of this process must

therefore be a body of code, documentation, test data and their resultant outputs. This material will be used to inform, control and provide checkpoints within the process as well as ensure that the author(s) of the software will have confidence in the software in its new form. Involvement of the authors is essential, as they need to develop a new image of their software. The tasks for this activity are:

- Survey approaches to the maintenance and renewal of legacy software
- Perform tools assessment on typical community applications
- Develop a legacy software process for community software
- Make tools available through suitable interfaces
- Provide web base documentation on tools

Part of this activity is already funded under the current Facilities Agreement (see Table 2).

### **5.3. Evaluation of Methodologies, Tools and Technology**

The computer science community has a long history of developing new methodologies, tools and technologies to aid the development of computing applications. These range from new languages, such as JAVA, to frameworks and environments that gather these tools and processes together in an integrated form, such as the Microsoft Visual studio. Along side these elements the computer science community are at the forefront in developing tools to assist the applications writer exploit the most advanced computing architectures such as the IBM Regatta and the SMP structure of the NEC SX6 or Silicon Graphics 3000 Series.

#### **5.3.1. Use of More Modern Languages**

There is a growth in the use of other languages and programming models other than the procedural style of Fortran. Languages such as C++ and Object Orientation are becoming more common in numerical software.

There is a great need within the UK community to assess the potential of these languages and programming paradigms in the context of computational science. There appears to be resistance to moving to languages such as C++ in many research projects. This is often due to the heavy investment in Fortran-base software over many years. However the dominance of Fortran in many computational science and engineering projects does not rule out the use of the most up to date software engineering techniques such as object orientation (OO).

The tasks for this activity are:

- Keep a technology watch on language and programming developments
- Investigate the use of Fortran 95 OO methods in software development
- Assess potential gains of using such approaches

Part of this activity is already funded under the current Facilities Agreement (see Table 2).

#### **5.3.2. Software Re-Engineering**

The final stage in upgrading legacy software is its re-engineering to use the current best practice in language and software design. There are a few tools dedicated to software re-engineering in general but only a very limited number addresses Fortran. In general these programs perform a structural analysis of the software and provide some transformation tools but full automation is not available. More of the re-engineering - moving the code around - has to be done by the programme. Some commercial tools are available for Fortran applications such as *FORESYS* from SIMULOG and *Understand for Fortran* from Scientific Toolworks Inc.

The tasks for this activity are:

- Survey available tools
- Assessment of some selected tools on *real* applications

- Performance tests

The output of this activity will be used in the later stages of the *Legacy Software* activities.

### **5.3.3. Integrated Design Environments (IDE)**

For most operating systems and languages *Integrated Design/Development Environments* are becoming available. Probably the best known of these is the Microsoft Visual Studio systems for windows. The visual systems provide a framework for software development and the Compaq Visual Fortran is the best known Fortran example. Similar tools are available for UNIX systems but these are in general less developed. The tasks of design, implementation, version control and maintenance are usually handled by these systems.

The tasks for this activity are:

- Survey of available environments
- Select two or three for detailed evaluation

### **5.3.4. Parallelisation Tools**

Automatic parallelisation has always been the ultimate goal of compiler developers. There are a few vendors that provide such compilers but their success is limited. Similarly there have been few tools to aid the parallelisation process; examples include ParaWise (CAPTools), WPIPS and POLARIS. These in general produce a source to source transformation using dependency analyses. Other aids to parallelisation are specially developed libraries or tool sets such as ScaLAPACK or PETSc. These libraries will provide parallel functionality at the cost of using a specific programming model and in-built data structures.

Although many applications are now designed to make use of MPI there are still many legacy programs that would benefit the community if they were available on machines such as HPCx.

This activity would survey the parallelisation tools available and would include compilers with automatic parallelisation. This task will look more critically at tools such as ParaWise in a number of test cases.

The tasks for this activity are:

- Review tools to assist parallelisation
- Evaluate a group of suitable tools
- Obtain and apply one of the source-to-source parallelisation tools to a group of selected examples
- Reports on the evaluation and test cases

### **5.3.5. Compilers**

The most basic software tool for computational science is the language compiler. There are many providers of compiler technology for both Unix and Windows environments.

While the primary goal of a compiler is often seen as generation of efficient code from any valid source code presented to it, the ability to detect programming errors is more important in the development of quality software. While all compilers can be expected to reject obvious syntax errors, good ones will provide additional warnings about possible mistakes, such as variables that are used before being set. The explicit interfaces and intent statements which are available in Fortran 90 means that many compilers are now able to make checks which previously were only possible using tools such as ftnchek.

A good comparison of many Linux and Windows Fortran compilers is given on the Polyhedron web site: <http://www.polyhedron.co.uk>. They include a set of diagnostic tests to compare how effective each compiler is at trapping common errors. However the technology is rapidly developing and it is important for CSED to keep abreast of these developments.

The tasks for this activity are:

- Maintain a technology watch on compilers
- Evaluate compilers with the aid of providers
- Ensure CSED has the best compiler technology both for software development and for high performance production.

#### **5.4. *Symbolic Algebra Systems***

Although computation is the bread and butter of the computational science there is often a need to develop or simplify the mathematical representation of a computational model. Modern symbolic algebra systems provide the scientist with an excellent tool to perform this. Although there are few well-developed systems information on their capabilities is not well known and there are few comparisons between such systems.

There is currently some use of these types of systems in the CCP programme. *Maple* is being used to compute derivatives of functions symbolically on a regular basis. From these computations Fortran source code is generated which is embedded into the applications. Versions of *MuPAD* and the symbolic algebra toolbox of *MATLAB* have also been used in the analysis of novel finite element methods.

The tasks for this activity are:

- Review of symbolic algebra software in the light of CSED needs
- Technical assessment of a few selected system

This work would lead to the purchase of a system.

#### **5.5. *Problem Solving Environments (PSE)***

There is an increasing use of *Problem Solving Environments* in the community: for example *Matlab* and *Scilab*.

The MathWorks, vendor and developer of *MATLAB*, offers a set of integrated products for data analysis, visualisation, application development, simulation, design, and code generation. *MATLAB* is the foundation for all the MathWorks products.

*Scilab* is a scientific software package for numerical computations providing a powerful open computing environment for engineering and scientific applications. It has been developed since 1990 by researchers from INRIA and ENPC. Distributed freely via the Internet since 1994, *Scilab* is currently being used in educational and industrial environments around the world.

These are two of the main *general-purpose* environments. There are many other application specific and prototype systems. For example *FEMLAB*: *FEMLAB* is an interactive environment to model and simulate scientific and engineering problems based on partial differential equations (PDEs) using the finite element method. *FEMLAB* is currently built on top of *MATLAB* but the next version will be a stand-alone system. *FEM2Dlib* is another example: it is a Fortran 90 module with a set of data structure definitions, functions and subroutines that can be used to solve simple problems with ordinary and partial differential equations using the Finite Elements Method (FEM).

The tasks for this activity are:

- Review and technology watch on these systems including developments in the well-established systems such as MATLAB.
- Assessment of new systems with potential in computational science.

## 6. Community & Vendor Participation

Although the scope of the SES Programme will be the SLA, CCP and HEC programmes within CSED it is clear that the results of the programme should be of use to the general computational science community.

The programme will seek the requirements of the CCP and HEC communities and review the practices being currently used.

Clearly much of the software in this area is commercial. Where possible and where thought useful collaborations will be sought with software vendors. Preliminary contacts have been made with Intel and IBM over CASE and optimisation tools and approaches will be made to a number of the other tool vendors.

## 7. Dissemination of Software and Results

All the results of the activity will be disseminated through a CSE Software Engineering Support Programme Web site and through seminars and workshops. The SESP web site will contain:

- An overview of the aims and objectives of SESP
- Detail of contacts in the programme
- Summary pages on the programmes activities and findings
- All technology watch and assessment reports (pdf, ps, html)
- Selected software
- Links to software and other software engineering pages of interest to computational scientists.

Seminars and workshops will be arranged to disseminate the results of the activity or to provide hands on experience with specific software tools. These may be arranged in association with the software vendors.

## 8. Resources

We view this as the development of a long-term support programme in the provision of software engineering technology and methodology to the computational science community. The SLA is the most appropriate framework in which this work should sit benefiting and participating in the SLA reviewing and assessment arrangements.

There are numerous threads to the SES programme. Table 1 below gives an initial broad estimate of the resources required by the programme and Table 2 gives an indication of the effort in FTE and time planned for each thread over the first two years.

	Year 1	Year 2 and on-going
Staff (FTE)	3	3
Software Licenses (£K)	10	5

**Table 1: Summary of Resources**

## 9. Future Programme for SES

The SES Programme is viewed as a bringing together of a number of software engineering strands important to ongoing CSED activities. It will become an ongoing activity along side those of the CCP and HEC programmes providing the software engineering input to these programmes. Outlined above is an initial two-year programme, the second year of which is intended to be an estimate of continuing year-on-year resource needs. Following, and subject to, the review mentioned in section 3, the programme will be developed in consultation with the CCP and HEC activities during the early part of the second year and a rolling programme of activities proposed.

CSED believes that an enhanced investment in software engineering practices now will pay significant dividends in the future. These include: higher quality scientific software; shorter development cycles; easier maintenance, porting and performance tuning. As the computational research community increasingly recognises the strategic importance of cost-effective software support as well as hardware facilities, we believe this is a timely and appropriate initiative for CSED.

## 10. Some Related UK & US Activities

### 10.1. UK Groups and Resources

- The Research Institute for Software Evolution (RISE), Durham:  
<http://www.dur.ac.uk/CSM/>
- De Montfort's Software Technology Research Laboratory  
<http://www.cms.dmu.ac.uk/STRL>
- Report on first SEBPC workshop on legacy systems:  
<http://www.dur.ac.uk/CSM/SABA/legacy-wksp1/report.html>
- RENAISSANCE ESPRIT Project on Re-engineering:  
<http://www.comp.lancs.ac.uk/computing/research/cseg/projects/renaissance/RenaissanceWeb/index.html>
- Lancaster University's Computer Department:  
<http://www.comp.lancs.ac.uk/computing/>
- Distributed Software Engineering Group, Imperial College:  
<http://www-dse.doc.ic.ac.uk/>
- Software Evolution and Reengineering Group, De Montfort University:  
<http://www.monkeydancer.co.uk/serg/>
- Department of Computing Science and Mathematics at the University of Stirling:  
<http://www.cs.stir.ac.uk>

### 10.2. US Groups and Resources

- The Center for Empirically-Based Software Engineering:  
<http://www.cebase.org/www/home/index.htm>
- The Experimental Software Engineering Group (ESEG) of the University of Maryland:  
<http://www.cs.umd.edu/projects/SoftEng/tame>
- The University of Southern California, Center for Software Engineering:  
<http://sunset.usc.edu/index.html>
- Fraunhofer Center for Experimental Software Engineering - Maryland:  
<http://fc-md.umd.edu/fcmd/index.html>
- The NASA Software Engineering Laboratory (SEL):  
<http://sel.gsfc.nasa.gov/>

- Laboratory for SEDS, University of Calgary  
<http://www.seng-decisionsupport.ucalgary.ca/>
- Software Engineering Research Center (SERC):  
<http://www.serc.net/>
- MIT Laboratory for Computer Science:  
<http://www.lcs.mit.edu/>
- The Software Engineering Institute (SEI):  
<http://www.sei.cmu.edu/>
- Software Technology Support Center:  
<http://www.stsc.hill.af.mil/>

### **10.3. General/European**

- International Software Engineering Research Network:  
<http://www.iese.fhg.de/ISERN>
- Fraunhofer Institute for Experimental Software Engineering, Germany:  
<http://www.iese.fhg.de/>
- Software Engineering Group, CWI:  
<http://db.cwi.nl/projecten/cluster.php4?clstnr=1>

**Table 2: Allocation of Effort to Tasks**

Activity	Current SLA Allocation <sup>1</sup>		Year 1				Year 2 and on-going			
	FTE per year	Staff involved	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
1. Technology Watch & Reviews	0.3	JA, MFG, MA	0.25	0.125	0.125	0.125	0.125	0.125	0.125	0.125
2. Quality Assurance	0.65	CG	0.25	0.125					0.125	
3. Legacy Software	0.1	CG		0.125	0.125	0.125	0.125	0.125		
4. Tools & Technology										
<i>4.1 Modern Languages &amp; Approaches</i>	0.2	CG	0.125	0.125		0.125	0.125			0.125
<i>4.2 Software Re-Engineering</i>					0.125	0.125	0.125	0.125		
<i>4.3 Design Environments (IDE)</i>								0.125	0.125	0.125
<i>4.4 Data Representation &amp; Management</i>					0.125				0.125	
<i>4.5 Parallelisation Tools</i>			0.125	0.125			0.125	0.125		
<i>4.6 Optimisation Tools</i>					0.125	0.125			0.125	0.125
5. Symbolic Algebra Systems				0.125	0.125			0.125		0.125
6. Problem Solving Environments						0.125	0.125		0.125	0.125
Totals	1.25		0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75

<sup>1</sup> See HEC Development section of SLA