Explaining StGermain: An aspect oriented environment for building extensible computational mechanics modeling software

Steve Quenette[1], Louis Moresi[2], P. D. Sunter[1], Bill F. Appelbe[1]

[1]VPAC
Computational Software Development
Melbourne, Victoria, 3053 Australia
{steve,pds,bill}@vpac.org

[2]Monash University
School of Mathematical Sciences
Melbourne, Victoria, 3800, Australia
louis.moresi@sci.monash.edu.au

Abstract

HPC scientific computational models are notoriously difficult to develop, debug, and maintain. The reasons for this are multifaceted — including difficulty of parallel programming, the lack of standard frameworks, and the lack of software engineering skills in scientific software developers.

In this paper we discuss the drivers, design and deployment of StGermain, a software framework that significantly simplifies the development of a spectrum of HPC computational mechanics models. The key distinction between StGermain and conventional approaches to developing computational models is that StGermain decomposes parallel scientific applications into a hierarchical architecture, supporting applications collectively built by a diverse community of scientists, modelers, computational scientists, and software engineers.

1. Introduction

StGermain [1] is a software infrastructure project for developing computational fluid and solid mechanics codes. It began in 2003, and is now the development environment for a broad community of computational geophysicists. Following an iterative development approach, the same environment is now being adapted to research of multiscale metal forming processes.

StGermain's organization is very different from existing libraries and frameworks for computational science. This paper aims to explain the history and the driving motivations of this environment. We then map the environment objectives to the programming constructs that are the foundations of StGermain.

2. Background: The need for a paradigm shift in computational science

Over the period of 2005 to 2006, the NSF cooperative Computational Infrastructure in Geodynamics (CIG) [2] hosted several workshops soliciting community direction on computational software infrastructure for several disciplines of geodynamics. In particular, we reference the NSF funded working group reports from the Mantle Convection Workshop [3], the Workshop on Tectonic Modeling [4], and the Magma Migration Workshop [5]. These disciplines are distinct in that computational mechanics computer simulation is used and accepted as a means of supporting theories on the processes of the Earth.

What is noteworthy from these reports is that several "codes" have prevailed, deemed proven over time, and accepted by the community for specifically scoped problems. Each code is essentially differentiated by the numerical schemes and constitutive models (equations) adopted. Such codes tend to have just one primary designer, and have evolved in an ad–hoc basis driven by individual academic research needs and resources.

However, the common reported objective of these communities is to broaden the scope of geological signals modeled, incorporating phenomena that cross temporal or spatial scales (including 3D parallel models). Yet, the existing codes presently do not have the numerical schemes or the multi–physics capabilities to enable such models. Hence, the working group recommendations are largely directed at what numerics and physics will enable these next generation models.

This problem of obsolete or hard to maintain codes is not unique to computational geophysics or the NSF. Equivalent problems are evident in programs within NASA, ASCI and DARPA [6].

1-4244-0910-1/07/$20.00 ©2007 IEEE

3. A case–study of research code evolution: The path to Underworld

Our observation is that the discrete set of numerical and constitutive capabilities of a code predominantly determines the scope of modeling opportunities of that code. This observation, predates the aforementioned CIG initiative, and began with a study into the evolution of the Ellipsis [8] code, as part of the development of its parallel and 3D ultimate successor Underworld [11, 12] largely within the Australian Computational Earth Systems Simulator (ACcESS) MNRF [9] Snark project.

Underworld's ancestry is illustrative. A code by the name of CONVECT [10] began in 1989 for a PhD dissertation. It is a 2D mixed finite element / finite difference code written in C developed for Stokes flow problems. Experiences from this led to the development of CITCOM [13] in 1994 as part of a post–doctorial position. CITCOM was a 2D and 3D finite element with Multigrid code also written in C. It was designed for a spectrum of mantle convection and Green's functions analysis problems. That is, effort was made into making the code extensible to new problem domains. Despite exhibiting some build and function pointer facilities to enable extensibility, by 1997 at least four distinct flavors of CITCOM existed. CitcomT [14] incorporated a representation for lithosphere faults by means of zero width elements and became 3D–spherical (enabling whole Earth not just regional models). Another variant was developed that incorporated sub–surfaces to represent faults [15]. CitcomS [16] introduced parallelism, full Multigrid, and then also became 3D–spherical. CitcomS is still in wide use today. Lastly, the CITCOM creator's own version [17] took the path of full Multigrid and further developed constitutive models, which in–turn targeted the realm of emergent faulting. However, it did not become 3D–spherical.

This was followed in 1998 by the development of Ellipsis. Largely inspired by the latter version of CITCOM, but with a significant numerical scheme addition: the Particle In Cell (PIC) Lagrangian integration point method (a form of material point method [18]). Ellipsis was hence better engineered for large deformation models. It was however only developed in 2D, but a 3D version [19] has since been made by another author.

These different code versions were each motivated by new scientific objectives that spanned both numerical and physical domains. The software differences between them are significant. A source code merge and patching by a "forest" of switch statements were deemed impractical. Furthermore, the explicit faulting versions have not shared the continued use and evolution of CITCOM and CitcomS — suggesting that sometimes code features are experimental and need to be an omit–able part of the software infrastructure, yet still remain readily maintained for potential future use.

In 2001 the ACcESS Snark project began, which was to promote the concepts of Ellipsis, to be parallel, 3D and ultimately spherical. Given the 12 years of legacy behind CITCOM and Ellipsis, a significant decision was made at that time to "start from scratch", to utilize existing software infrastructure from the computational sciences domains, and to carefully consider the software architecture for successful extensibility and scalable parallelism. An example was the immediate use of the MPICH [22] flavor of MPI and PETSc [21], enabling portability from laptops to large cluster and shared memory machines. The software named Snark [20] was our first evolution — driven primarily by the widely ranged collective group's need to learn the methods and technologies involved. It achieved moderate parallelism up to 16CPUs and 3D, but omitted spherical. It was also written in C, utilizing object oriented design for controlling code (similar to the Model View Controller style of general application development [40]), but the Fortran–paradigm of large arrays for efficient data allocation and access.

The present evolution is named Underworld. It is used to model a spectrum of long–term geophysics problems. It is 3D, parallel, PIC, and has Multigrid. It was created using the following approach.

4. Objectives and approach

The fundamental lesson learnt in the development of Snark was that statically typed object–orientation, in which all types are fixed at compile time, of the controlling code was not going to be enough to sustain extensibility beyond the implemented numerical and constitutive schemes. That is, these scientific numeric and physics concepts cross both controlling and data aspects, a quality that traditional object and component– oriented programming do not handle well [23, 39]. Clean enscapsulation of these concerns is non–trivial, yet our case study suggests that these concerns should be implemented as interchangeable software components (that is, manageable units of code with clear interfaces and high cohesion [24]).

Thus the measure of success of adaptability is to be able to change the implementation of either a numerical scheme or constitutive model without needing to change the phenomena model code.

Our experience was that the numerical schemes and constitutive models have a significant bearing on the internal structural boundaries of a code. In theoriz-

ing the mythical man–month Brooks [25] also suggests that internal structural boundaries are the lowest point of usable abstraction in a program, and that changing these structural boundaries is changing the very essence of the program. That is, it is an expensive operation.

The question then becomes how does one achieve such an extensible environment? In particular, how is it achieved in a scalable parallel HPC environment?

Our approach was to consider the expectations and interactions of the types of people involved in creating a code base to the prescribed capability from an aspect oriented perspective. This drove the creation of programming features that would enable such an environment. These are implemented in an open source package named StGermain.

5. A conceptual model of the nature of computational code development

The diversity of CITCOM illustrates that the development of scientific codes often occurs as a distributed community effort. That is, no one person created the collective CITCOM and descendants' capability. Rather, with respect to academic code development, a powerful base–line code is founded of which others build upon to suit their research purpose [26]. The establishment of the base–line code is usually through peer acceptance of a significant resultant publication. As others undertake research derived from this publication, it is natural that changes to the code are needed. Hence, a research code needs to be extensible.

CITCOM's case is also illustrative of the leveraged computational science disciplines, and how different researchers extended CITCOM with a different geological phenomena context in mind:

- CitcomT and CITCOM-with–faults added the two different forms of explicit faulting, within the scope of finite element method techniques (classifiable as a computational mathematician concern), to enable research into larger scale geophysical phenomena (the Tonga–Kermadec subduction zone [14] and the role of faults in generating tectonic plates from mantle flow models respectively [15]).

- CitcomS targeted efficient global scale mantle modeling through the provision of parallelism (classifiable as a computer science concern) and a spherical discretisation scheme (classifiable as a computational mathematician concern) well suited to this problem.

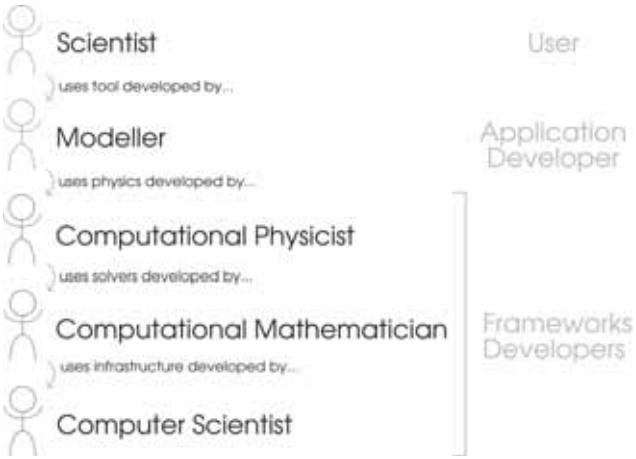

Figure 1. Competencies of a computational mechanics code — suggests that many skills are involved in developing such a code, and that these are research areas within themselves. However, as competencies, they are related to each other, and these competencies need to be aware they have "users".

- CITCOM's 2nd generation focused on the emergent generation of faults through macro–scale constitutive behavior [27] (classifiable as a computational physicist concern), which in turn are strongly non–linear behavior which produce systems of linear equations that are harder to solve [28], and hence the emphasis on improved numerical scheme technology (a computational mathematician concern).

In [26] we proposed a roles or competency based conceptual model of how researchers within these virtual communities of a computational mechanics code interact (see Figure 1). When contributing to or using the code, a member assumes the role of a particular discipline. Users may assume more than one role, based on their competencies.

5.1. Scientist

Our definition of a scientist, or "end scientist / engineer", is one who uses an established tool to undertake research or study of scientific or engineering relevance. Scientists can be modeling or even computer illiterate, but have strong scientific domain knowledge. They use a computational model developed by a "modeler", changing only well understood parameters. As scientists are not programmers, their interaction with the model is via "input decks", not programming.

5.2. Modeler

A modeler, or "phenomena modeler", has an understanding of the mathematical and physical abstractions used in modeling (for example the established partial differential equations and constitutive behavior for a class of problems). When creating a new model, they create a conceptual composition of these numerical and physical concerns. The modeler may switch roles, using a fabricated model as a scientist engaging in a research study. If a desired numerical or physical feature is not available, they switch to become a computational physicist or mathematician, and undertake some coding to add or modify that feature.

The natural language to this competency is mathematics (partial differential equations (PDEs) and constitutive behavior is described mathematically, e.g. [13], [14], [16]). Modellers use tools such as Matlab in geophysical modeling (e.g. for verification [29] and for analysis [30]). However, HPC performance of tools such as Matlab is often a limiting factor.

The numerical scheme evolution in the developments of CONVECT–to–Snark is suggestive of why highly expressive environments fail to perform and scale as well as purpose built C and Fortran codes. The numerical schemes, which improved with each evolution, had significant bearing on the speed and scalability of the code. That is, in computational mechanics users have to be concerned with how the problem is solved, not just what the problem is. Thus, there is a tradeoff between expressiveness and performance. In HPC this performance and respective developer productivity relationship is not well understood [31].

With this in mind, our approach is to moderately distance from the mathematical expressiveness of the phenomenological model. Instead we provide the modeler with both an environment to declaratively describe the model and how it is solved.

5.3. StGermain's Model Description Files

Declarative languages [32] are languages where the programmer describes what to do as opposed to how to do it (for example SQL). Declarative languages are considered safer and more productive than 3rd generation languages (such as C). In StGermain this language is a weakly typed XML schema (see Figure 2).

A modeler uses this language to compose a model application by describing the associations across numerical, constitutive, governing and computational concerns. Each aspect is implemented as a software component (refer to section 5.4) within a toolbox or plugin, which in–turn is dynamically loaded. Toolboxes are a collection of StGermain components. For example, Underworld is a toolbox of long–term geophysics boundary conditions and rheologies. It leverages upon three other toolboxes in a layered software approach [33]. StgDomain contains components such as parallel 3D meshes and swarms (our equivalent of meshes for a managed collection of particles). The StgFEM toolbox houses constructs for defining finite element systems, with several template PDEs and appropriate solvers for them. PICellerator is both a framework for developing Lagrangian integration schemes, and a toolbox that contains the PIC [8] implementation. Underworld is also an application with examples. Plugins differ from toolboxes predominantly by the order they are loaded: the input file(s) is read, toolboxes are loaded (populating the StGermain factories), the input file components are instantiated, then the plugins are loaded and subsequent components instantiated. Plugins hence allow a modeler to leverage an existing model, isolating their desired

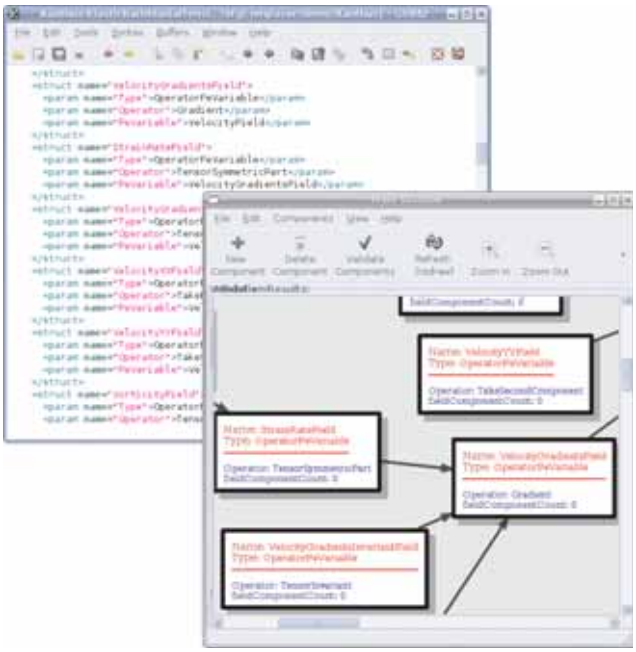

Figure 2. Modelers describe the composition of numerical and physical concepts that make a desired phenomenalogical model in an XML based model description file. The back image shows an example of how field operators are composed. These components have parameters and associations to other components, exampled by the generated graph in the front image.

changes into a small plugin (encapsulation), which then adds these artifacts to the StGermain factories. This helps prevent users from directly modifying the layers of utilized toolboxes.

5.4. StGermain's Lightweight components

StGermain components are intended for the control (management and algorithms) of fine grain computational concerns. They differ from the concept of self–describing service architectures, such as the Common Component Architecture (CCA) [34]. They exhibit six pre–defined services: a default constructor for use in the resource broker, and then the five phases of construction, build, initialize, execute and destroy. They are implemented as virtual functions, in the C++ sense, through StGermain's optimized implementation of inheritance in C. They are available only to the local memory of a process.

A component must implement all its phases and has no mechanisms for publishing more capabilities. This limitation aligns with the fine–grain nature and HPC concerns. Hence the resource broker is merely an instance of the factory design pattern. Also, due to the memory intensiveness and book keeping of (for example) finite element codes, the build and initialization occurs in distinct phases.

StGermain components, written as C ".C" and ".H" pairs, are required to provide a supplementary ".META" description file. This requirement is enforced by the StGermain build system. Some of this metadata is auto–generated into string symbols and linked into the relevant library. This provides the ability to audit the copyright or referenced papers per component at run–time, for example. Some are used for run–time type information, which for example, can be used to validate parameters and associations. This information is also used to generate a component reference document. The entities include: copyright, licenses, description, parameter documentation, and association documentation. A possibility is to make this schema Dublin Core [35] compliant, where ultimately each component is sufficiently self–described and instantiable from an RDF [36] perspective. Another possible approach is via MDS [37].

The use of this generic component abstract type, and the ability to interchange components, prompts the question of "what is a valid phenomenological model". This issue is known as composability [38]. It is an emerging issue in the real–time simulation and web services domains, where new applications are being built from an established component base. In StGermain, syntactic composability is addressed through the validation of component associations by means of the meta information. However, semantic composability is not address thus far. Semantic composability asks the question: does the association of any two components make sense? In Underworld's case an example of this is using both the Arrhenius and FrankKamenetskii temperature rheology components at the same time.

5.5. Computational Physicist

A computational physicist is concerned with the development of constitutive models. This includes the implementation of laws for viscosity, plasticity, elasticity, and the respective parameterization of materials. Often the adjoining test or benchmark to a new or modified constitutive model is based on laboratory experiments. We tend to encourage using them as part of the regression test suite. Computational physicist typically wish to work in an environment that hides explicit parallelism. However some physical concepts are inherently global, which in turn can be limited to light parallel programming, and in turn may effect parallel scaling. A modeler can chose whether they wish to use such a feature.

Here entails a long–standing complication: let us assume a computational physicist wishes to add a new yielding model (i.e. a model for describing how and when a material becomes plastic as a function of the present strain and stress state), for example Drucker–Prager. Structured and typed–based object–oriented programming would require that the original source be modified in three ways. It would include adding the new behavior in the section of code that associates constitutive behavior to material types. It would implement the behavior itself. It would also add the necessary parameters to the material model (which can take the form of a lookup table, and/or history variables on the discretised material). However, to the computational physicist, this yielding model is one distinct concept. Furthermore, such coding practice is against our objectives.

5.6. Separation of concerns

Jacobson and Ng [39] highlights two crosscutting concerns that are evident in the example above: peers and extensions. Concerns are functional and non–functional requirements (e.g. memory access speed and numerical ideas are concerns that are typically not considered as functional requirements). Ideally one aims to encapsulate all concerns with classes or components. However, inevitably some concerns are incompatible with the object model, and consequently impact mul-

tiple classes. The yielding model and the concept of a material are peers, that is, they are distinct concerns. Similarly the yielding model is a concern implemented on an established constitutive matrix builder (i.e. an extension of an another concern). An implementation that will have better separation of concerns, and arguably better extensibility, is to have the yielding code localized. Where parts of this yielding model encompass other concerns, those parts are patched into the appropriate concerns at compile– or run–time. This is in essence the aspect–oriented methodology [39].

5.7. StGermain entry points and extensions

StGermain supports an aspect oriented methodology through means of entry points and extensions, for behavioral and state patching respectively.

An entry point is essentially a container of function pointers (hooks), which by default execute in order. The containers can add, replace all, insert before, and other such operations typical of containers. The implementation of concerns that have known crosscuts instantiate entry points during the build phase and execute them at the appropriate spot (the runtime overhead is at most 20% and is considered acceptable [7]).

Extensions aim to mimic dynamic typing, where by one concern/component may extend a data structure on another concern/component at runtime. Similarly to entry points, concerns with known crosscuts need to instantiate an extension for the data structure. If extension contributions are added before the data structure is instantiated, then the extension is a natural extension of the data structure. Otherwise, a separate memory block for the extension is allocated. The component phases and association construction ordering are used to promote the former case, increasing memory locality.

Having to know the crosscuts is a shortfall of this implementation. However, given the objectives of HPC performances, and layered software, this technical limitation is deemed acceptable. An alleviating feature is the StGermain context, which is essentially a component implemented by only entry points. Its purpose is to help nurture dynamic coding.

5.8. Computational Mathematician

A computational mathematician is concerned with the discretisation and numerical schemes of computational problems. Unfortunately, there is an invariable exposure to parallelism to some facets of this competency. Whereas code to use an operator on a field (e.g. taking a gradient) requires no exposure to parallelism, implementing a new operator may as the field is decomposed over the processor space. Changes made by a computational mathematician typically involve low level constructs and in turn typically have significant impact. It further compounds the need to have an effective method for ensuring the separation of concerns.

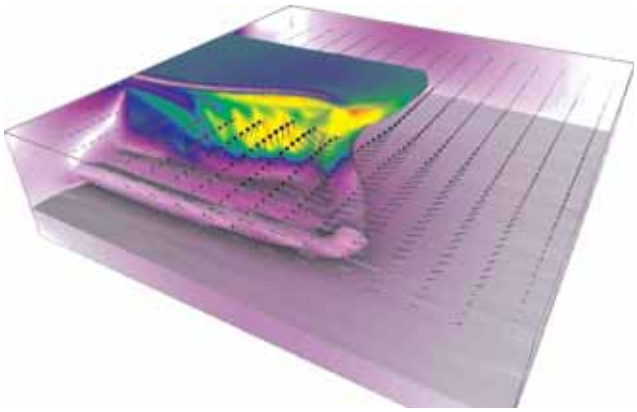

Figure 3. A snopshot of a slab subduction model, illustrating the inherently 3D phenomena. The visualisation was produced by gLucifer a parallel compute–time visualisation toolbox built upon StGermain.

5.9. Computer Scientist

In this scheme, the computer science competency is concerned with performance, parallelism and facilitating this environment. That is, it encompassed software design and computational science fundamentals. The development of StGermain is by this competency.

6. Underworld Examples

We present two examples based on the Underworld toolbox. They share very many components, but differ mostly in exact consititutive behavior used.

6.1. Slab subduction

One of the Geodynamics modeling applications that the StGermain–to–Underworld framework has been applied to is an important area of plate tectonics: the subduction of tectonic plates or "slabs" [12]. In particular, researchers wished to investigate the three dimensional flow and rollback regime as a result of different plate geometries and material models (see Figure 3).

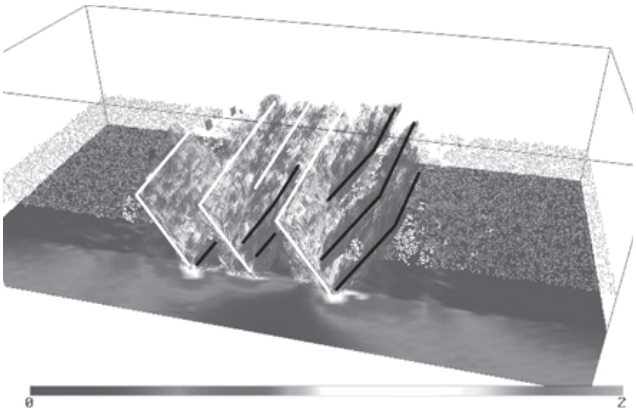

Figure 4. Emergent shear bands from rheological behaviour models.

This modeling goal required capabilities from, and drove improvements to, the framework at all levels, such as checkpointing very large 3D datasets for both restart and analysis; and methods to visualise very large particle sets as isovolumes. Models range from 150,000 to 1,500,000 unknowns on up to 32CPUs. Prior work in this field had typically been constrained to 2D models.

6.2. 3D faulting

Another example is the scaling of rheological models based on [11] to scales of geological relevance in 3D (see Figure 4). This model needs upwards of 2 million unknowns, which would be more practical with scalability to over one hundred CPUs. To reach this objective, we have identified that despite StGermain meshing being able of 3D decomposition, the original finite element equation numbering component is not. A current activity is the development of this feature. When ready, using it should be as easy as declaratively substituting it in the model description file, as was proven when the multi–grid component became available.

7. Conclusion

As demonstrated in the examples, distinctive and ever increasingly scalable models are being developed in the StGermain environment. The driving objectives are founded on an established community need, and demonstrated through the 12year evolution and divarication of CONVECT to Underworld.

The mixture of fine grain components for controller code over an array based data model, a declarative language for phenomenological model composition, and aspect oriented technologies for encapsulating scientific concepts are the programming constructs used to facilitate this environment. By creating an environment that nurtures interoperable numerical schemes and constitutive behavior, and encouraging their development by a broad community, real reuse of HPC ready computational mechanics software is possible.

8. Acknowledgements

Funding for StGermain has been predominantly provided by: VPAC, the ACceSS MNRF Snark project, the APAC CTT program. We would like to thank the significant contributions to StGermain from Computational Infrastructure in Geodynamics, Caltech's GeoFramework NSF ITR, and the University of Deakin's Xanthus project. We would also like to thank Alan Lo, Luke Hodkinson, Raquibul Hassan and Kathleen Humble for their contributions to StGermain, Julian Giordani and Kent Humphries for their meta data work, and linear algebra, FEM, PIC and Underworld contributions of Matt Knepley, Dave May, Mirko Velic, Rob Turnbul. Dave Stegman and Justin Freeman have been instrumental in driving the development of the infrastructure with their slab models 3. gLucifer is maintained by Cecile Duboz.

References

[1] `http://www.stgermainproject.org`
[2] `http://www.geodynamics.org`
[3] S. Zhong, et al., Report to the CIG from the Boulder Mantle Convection Workshop, `http://www.geodynamics.org`
[4] D. Harry, L. Lavier, and S. Willet, Report to CIG from the NSF Workshop on Tectonic Modeling, `http://www.geodynamics.org`
[5] M. Spiegelman and L. Montesi, Report to the CIG from the 2006 Magma Migration Workshop, `http://www.geodynamics.org`
[6] D. E. Post and L. G. Votta., Computational Science Demands a New Paradigm, *Phys. Today*, January (2005) 35.
[7] S. M. Quenette and B. F. Appelbe and M. Gurnis and L. J. Hodkinson and L. Moresi and P. D. Sunter", An investigation into design for performance and code maintainability in high performance computing, *ANZIAM J.*, 46, C1001-C1016, 2005
[8] L. Moresi, F. Dufour, and H. B. Muhlhaus., Mantle convection modeling with viscoelastic/brittle lithosphere: Numerical methodology and plate tectonic modelling, *Pure And Applied Geophysics*, 159(10) pp 2335-2356, August 2002.
[9] `http://access.edu.au`

[10] Moresi, L. and Parsons, B., Interpreting gravity, geoid, and topography for convection with temperature-dependent viscosity - application to surfacefeatures on venus, *J. Geophys. Res.-Planets*, 100, 21155-21171, 1995

[11] Moresi, L. and Muhlhaus, H.-B., Anisotropic viscous models of large-deformation Mohr-Coulomb failure, *Philosophical Magazine*, 86, 3287-3305, 2006

[12] Stegman, D. R. and Freeman, J. and Schellart, W. P. and Moresi, L. and May, D. A., Influence of trench width on subduction hinge retreat reates in 3D models of slab rollback, *Geochem. Geophys. Geosys.*, 7, Q03012, 2006

[13] Moresi, L. and Solomatov, V. S., Numerical investigations of 2D convection with extremely large viscosity contrasts, *Phys. Fluids*, 7, 2154-2162, 1995.

[14] Billen, Magali I., Gurnis, Michael and Simons, Mark., Multiscale dynamics of the Tonga-Kermadec subduction zone, *Geophysical Journal International*, 153 (2), 359-388, 2003.

[15] Zhong, S., M. Gurnis, and L. Moresi., The role of faults, nonlinear rheology, and viscosity structure in generating plates from instantaneous mantle flow models, *J. Geophys. Res.*, 103, 15255-15268, 1998

[16] Zhong, S., M. T. Zuber, L. Moresi, and M. Gurnis., Role of temperature-dependent viscosity and surface plates in spherical shell models of mantle convection, *J. Geophys. Res.*, 105, 11063-11082, 2000.

[17] Moresi, Louis and Solomatov, Viatcheslav., Mantle convection with a brittle lithosphere: thoughts on the global tectonic styles of the Earth and Venus, *Geophysical Journal International*, 133 (3), 669-682, 1998.

[18] D. Sulsky, and H. Schreyer, Antisymmetric form of the material point method with applications to upsetting and taylor impact problems, *Comput. Methods Appl. Mech. Eng.*, 139 (1996) 409-429.

[19] O'Neill, C., Moresi, L., Mller, R.D., Albert, R. and Dufour, F., Ellipsis 3D: a particle-in-cell finite element hybrid code for modelling mantle convection and lithospheric deformation, *Computers and Geosciences*, 32, 1769-1799, 2006.

[20] L. Moresi, D. May, J. Freeman and B. Appelbe, Mantle convection modeling with viscoelastic/brittle lithosphere: Numerical and computational methodology Computational Science, *ICCS 2003, Pt III, Proceedings 2659*, pp 781-787, 2003.

[21] S. Balay, V. Eijkhout, W. D. Gropp, L. C. McInnes, and B. F. Smith., Efficient Management of Parallelism in Object Oriented Numerical Software Libraries, *Modern Software Tools in Scientific Computing. E. Arge, A. M. Bruaset, and H.P. Langtangen, editors, Birkhauser Press*, pages 163-202, 1997

[22] W. Gropp and E. Lusk and N. Doss and A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, *Parallel Computing*, 22, 6, 789-828, 1996.

[23] Grundy, J., Aspect-oriented requirements engineering for component-based software systems, *Requirements Engineering, 1999. Proceedings. IEEE International Symposium on*, vol., no.pp.84-91, 1999

[24] Brown, A.W.; Wallnan, K.C., Engineering of component-based systems, *Engineering of Complex Computer Systems, 1996. Proceedings., Second IEEE International Conference on*, vol., no.pp.414-422, 21-25 Oct 1996

[25] F. Brooks, Jr., The Mythical Man-Month (20th Anniversary edition), *Addison-Wesley*, 1995.

[26] S. M. Quenette, L. Moresi, P. D. Sunter, L. J. Hodkinson, R. Hassan, A. Lo, B. F. Appelbe, R. Turnbull., Supporting community based computational code development, *Proceedings of APAC05*, 2005

[27] Fullsack, Philippe., An arbitrary Lagrangian-Eulerian formulation for creeping flows and its application in tectonic models, *Geophysical Journal International*. Vol. 120, no. 1, pp. 1-23. Jan. 1995

[28] Moresi, L. and Zhong, S. J. and Gurnis, M., The accuracy of finite element solutions of Stokes' flow with strongly varying viscosity, *Phys. Earth Planet. Inter.*, 83-94, 97, 1996

[29] Kaus B.J.P. (2005). Modelling approaches to geodynamic processes, *PhD-thesis. ETH-Zurich, Switzerland.*

[30] Gorbatov, A., Limaye, A. and Sambridge, M., Tomoeye: A Matlab package for visualization of three-dimensional tomographic models, *Geochem. Geophys. Geosyst.*, 5, No. 4, 9 April 2004.

[31] J. Kepner, HPC Productivity: an Overarching View, *International Journal of High Performance Computing and Applications: Special Issue on HPC Productivity (ed. Kepner)*, vol. 18, no. 4, Winter 2004.

[32] Diomidis Spinellis, Choosing a Programming Language, *IEEE Software*, vol. 23, no. 4, pp. 62-63, Jul/Aug, 2006

[33] Bill Appelbe, Louis Moresi, Steve Quenette, and Patrick Sunter., Scientific Software Frameworks and Grid Computing: Improving Programming Productivity, *WoCo9: Grid-Based Problem Solving Environments: Implications for Development and Deployment of Numerical Software*, in press, 2006

[34] Rob Armstrong, Dennis Gannon, Al Geist, Katarzyna Keahey, Scott Kohn, Lois McInnes, Steve Parker, and Brent Smolinski., Toward a common component architecture for high performance scientific computing, *In Proceedings of the 8th High Performance Distributed Computing (HPDC'99)*, 1999.

[35] `http://dublincore.org`

[36] `http://www.w3.org/RDF`

[37] `http://www.globus.org/toolkit/mds`

[38] M. D. Petty and E. W. Weisel, A Composability Lexicon, *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, Orlando FL, March 30-April 4 2003, 03S-SIW-023.

[39] Ivar Jacobson and Pan-Wei Ng, Aspect-Oriented Software Development with Use Cases, *Addison-Wesley*, 2005.

[40] E. Gamma, R. Helm, R. Johnson, J. Vlissides., Design Pafterns, *Addison-Wesley*, 1995.