Applying embedded system feature guided architecture development in the high performance computing problem space

Charles J Gillan
Academic staff:
  Prof Stan Scott (HPC)
  Dr Ivor Spence
  Dr John Brown
  Dr Peter Kilpatrick
  Mr Rabih Bashroush

PhD students:
  Ms Rachel Gawley

The group has focused on applying software engineering principles particularly to embedded soft real-time telecommunications systems (SDH and SONET NEs).

soft => 50 ms threshold.

It was formed out of the JIGSAW collaboration with Nortel Networks during 1999-2001.
This short talk will highlight:

1. Embedded Software Architecture (telecomms transmission domain)
2. ECIT Toolset and Software Product Lines (SPL)
3. Practical experiences in HPC so far (work in progress)
1. Embedded Software Architecture

Background and similarities to HPC
Example system

A typical telecomms embedded system is an amalgam of heterogeneous components:

- multiple cards connected by a backplane, over which runs an inter-card comms protocol (even IP over PCI Express is possible)

- each card is a single board computer with a processor (typically Freescale), RAM and Flash.

- each card has several components: FPGAs, ASICs, Laser(s), Avalanche PhotoDiodes, which are controlled/managed/loaded from the s/w on the processor.
Conceptually, the inter-card programming model has had much in common with message passing in distributed HPC system.

Our suggestion is that HPC can now learn from the way in which these embedded systems handle heterogeneous electronic devices such as the FPGAs.

NB: Telecoms transmission behaviour is strictly defined in ITU-T standards; HPC behaviour is defined in math!
Typical single card embedded system

- Timer, interrupts
- Alarm correlation
- User I/f
- Device Driver (FPGA, ASIC)
- Logger
- Watchdog

Data structures passed via message queues
The system on chip revolution, including FPGA technology, has a DISRUPTIVE effect on the software architecture (substantial redesign costs).

Multiple *features* can migrate into hardware making entire sections of message passing dialogues disappear and requiring radically new architectural features => multiple products from common code base.

Example: Performance Monitoring counters (which are critical to SLA reporting/billing for telcos.)

Clearly, its better to have a design where the potential migration of features into hardware is incorporated.
Software companies/embedded equipment vendors have large existing revenue generating applications and IP assets. So, can’t just throw away and start from scratch. SPL methodology seeks to address this for commercial s/w.

Clearly, HPC community has similar constraints. e.g. 2007 is 50th anniversary of Fortran …
2. The ECIT Toolset and Software Product Lines
Feature driven architecture development:

- Define a feature model and feature driven architecture for the product family.
- Analyze architecture to suggest relevant implementation patterns.
- Apply generative programming techniques to derive skeleton code.

The research in Belfast covers all of these areas and aims to produce a toolset to support implementations in the embedded domain and more generally (HPC!).
1. Develop (reverse engineer) a feature model for the product family – use UCM notation for behavior.

2. Define a feature driven architecture for the family taking account of platform migration capabilities.

3. Analyze architecture to suggest relevant implementation patterns

4. Apply generative programming techniques to derive code.

GUI presentation of underlying textual representation.
The basis of this toolset lies in Software Product Line (SPL) engineering principles. This is a large subject in its own right including principles for organizational and project management.

Some organizations/companies have made the move from monolithic SPL production? Success stories can be found at [http://www.softwareproductlines.com](http://www.softwareproductlines.com)
SPL in practice

Product Plans

Domain Engineering

Core Assets with variation points

Software Factory

Application Engineering

Product A

Product B

Product C
Motivation

Number of products

Cost in $

Product centred
No SPL

Pioneering SPL
e.g. Cummins

New SPL
e.g. Engenio Inc.

ROI

Upfront costs too high to permit adoption of SPL

Low entry costs with INCREMENTAL ROI

Number of products
Why Software Architecture?

- Early, high-level system model
- Stakeholder understanding and communication
- Focus on specific system properties
- Separation of concerns
- Early analysis and simulation
- Improved processes and project management
3. Experiences so far

Feature modelling example
Architecture Example
Implementation Example
Bi-directional feature modelling to capture the feature variability within the operating platform as well as the software domain.
e-H Scattering at Intermediate Energy

- Incident electron
- H atom
  - sphere of radius ‘a’
- Scattered electron
Step 1:
A single particle differential equation with model radial potential and fixed angular momentum barrier defined by \( I \), is solved for the first \( n \) eigenvalues. This generates single particle basis functions.

Step 2:
The two electron wave function is expanded in terms of products of these functions subject to restrictions of angular momentum, spin and parity couplings of the two electron system: \( \psi_p(r_1, r_2) \)

Step 3:
Non-relativistic Hamiltonian matrix is formed as

\[
H_{pq} = \langle \psi_p(r_1, r_2) | K.E + P.E | \psi_q(r_1, r_2) \rangle
\]
Each Hamiltonian matrix element is defined in terms of a unique set of indices

\[ n_1 l_1 \quad n_2 l_2 \quad n_3 l_3 \quad n_4 l_4 \quad \lambda \]

where \( \lambda \) is determined by the angular momentum coupling. The matrix elements reduce to two dimensional integrals as follows:

\[ I = \int_{0}^{a} u_{n_1 l_1}(r) u_{n_3 l_3}(r) \left[ \frac{1}{r^{\lambda+1}} I_1(r) + r^\lambda I_2(r) \right] \, dr \]
where the terms $I_1$ and $I_2$ are inner integrals with limits dependent on the variable of integration, $r$, as follows:

\[
I_1(r) = \int_0^r t^\lambda u_{n_2l_2}(t) u_{n_4l_4}(t) \, dt
\]

\[
I_2(r) = \int_r^a \frac{1}{t^{\lambda+1}} u_{n_2l_2}(t) u_{n_4l_4}(t) \, dt
\]
Some sequential timings as follows (IBM Power 4 system):

- L = S = P = even
- \( n_{\text{max}} = 20 \quad l_{\text{max}} = 7 \)
- 1680 matrix elements
- 8,187,600 integrals
- 16 minutes

- L = 4, S = 0, P = even
- \( n_{\text{max}} = 20 \quad l_{\text{max}} = 11 \)
- 8900 matrix elements
- 302, 869, 500 integrals
- approx 10 hours
Active Object Pattern

Table in global storage

In-bound data encapsulated within envelope objects

Out-bound data encapsulated within envelope objects

Compute threads

Posix threads – here equivalent performance to OpenMP
Embedded s/w engineers may have something to offer HPC as it moves into a *heterogeneous* compute environment.

What’s missing from the ECIT Toolset is an analysis of adequate bit sizing for numerics of each problem. Other groups have such tools available.

Subset of available patterns may be applicable.
End