Software Tools Strategy

Tony Debling

February 16, 1988

1 Introduction

This outline document identifies the issues to be considered in producing the software development plan for the next two years.

1.1 General Strategy

We have almost completed the IBM TDS. This will provide a good quality interactive development system for a single user or small team of developers building embedded systems based largely on occam. We must now aim to support larger scale developments in mixed programming languages.

To support larger scale multi-user development the basic strategy is to build a more conventional programming toolset, designed so that the components can be easily integrated into emerging project support environments. The tools will support modular software development.

We must provide a general framework for access to system services, and provide the basic services for supported hosts.

We need to continue to improve the basic software development tools and how they fit together, and to fill in the gaps in these tools. High quality basic products are essential for our credibility. The quality of the compilers we produce directly affects the performance achieved using our processors. More effort will be spent on supporting scientific languages. Most effort should be spent on supporting C. Support for debugging mixed systems is required.

We must produce timely support for new variants of the transputer.

A team should evaluate and demonstrate opportunities for advanced development environments.

We need to make considerable improvements to our own software development environment and engineering practices. The quality of our current software in terms of design documentation, version and configuration control, and engineering standards is poor, even though the quality of algorithms and programming may be good.

1.2 Very general user requirements

1. Make it easy to develop and debug software for transputers
2. Tools should fit together well as a set
3. Tools should fit into their own environment
4. Tools should support multi-user development
5. Tools should support scientific languages in addition to occam

1.3 Understanding the market

We need more information about our intended markets; retrospective information about sales and trends.

How many sales do we expect to make to PC, VAX, SUN3, SUN4 customers over the next two to three years? What kind of development environments are they using now, will they be using?

What kind of tools are these customers expecting now, in two years time?

1.4 Hosts to be supported

Now: IBM/NEC PCs, VAX and SUN 3

Later: add SUN 4, SONY Workstation, ATARI, others?

What hardware will we be plugging into these hosts?

1.5 Portability

For the core development tools (compilers, configurers etc) portability should be easily achieved through careful use of C and restricted use of host services. The use of C however can be very ill-disciplined and error prone, I would favour choosing C++ as an alternative (it can be pre-processed into C retaining the portability). C++ is more secure and better supports larger scale software development. Objective C is another candidate.

2 Influencing factors

2.1 Toolset environment

The current toolset products have been extracted from the TDS (initially to provide a cheap compiler for Microway). There are a number of shortcomings with the existing product. The compiler will only detect a single error. Order of compilation of source becomes an apparent problem as it is no longer done for free by the TDS. The library mechanism is poor. It is basically an evolution of a short term solution.

The sources for each target machine are in variants of B1-occam as these are the only compilers we have which target to the VAX and SUN machines.

The toolset products are not compatible with the TDS; file formats, notion of what a library is etc.

2.2 Gaps in existing tools

1. Debugging for scientific languages.
2. Profiling.
3. Monitoring of program execution.
4. Message routing across transputer networks.
5. Software design tools.

2.3 Transputer evolution

The evolutionary transputers are likely to provide some additional instructions and hardware message routing.

The instruction extensions are likely to cause some small changes to the compiler and configurer. It will be important to have a redesigned library scheme which addresses the transputer type X error mode problem.

The virtual link support will require:

1. Software simulation of the virtual links. Message routing processes are added to the network of application processes to simulate the virtual links. This scheme can be supported with the existing compiler and configurer.

2. New definition for configuration language; including a hardware description language. A configurer which automatically inserts message routing processes for existing generation transputers and invokes hardware message routing for new transputers.

3 In-house development environment

We take too long to develop software.

3.1 Hardware requirements and Software tools

Significant time has been wasted by not having appropriate tools for the job (VAX and SUN toolset developments) and inadequate version control (libraries, servers, linker formats). Rebuilding a product can be a long, tedious and error prone task; in terms of collecting the components required build the product, finding a large enough home for the components, and processing them into the end result.

We have clear need for...

- Ethernet connections between our PCs
- VT220 emulation for PC
- Shared Software and Documentation Database
- Version control and configuration control system
- Bug reporting and change control systems
- Electronic mail
- Diagram drawing tools
- C programming and debugging tools on the PC
- Easy access to printers (a laser printer and line printer in wing)

We should set up a project to work out how to best meet these requirements.

3.2 Education and training

New methods of working, hew languages and environments will require training.

3.3 Organisation and standards

We require a framework for projects. What kind of documentation should be produced by who and when? How should software designs be expressed?

What should be delivered at each stage of the development process? How is development work reviewed?

3.4 Software support

We will be very much under resourced to support the emerging toolset products.

4 Development areas and issues

4.1 TDS development

We should freeze the TDS development after the product release. We must expect to provide a maintenance update in about one year. The upgrade should fix reported bugs and provide minor enhancements. We will also need to give consideration to compatibility with toolset products.

4.2 Toolset environment

Short term adjustments

The syntax checker (front end of the occam compiler in C) should be supplied as part of the occam 2 toolset. Without this facility for quickly checking the syntax (with multiple errors reported) I fear that the toolset will be unusable for software development: but ok for re-building systems.

The public domain make utility should be made available as a giveaway for VAX and PC customers.

Interfaces between tools

In order to support multi-user development we need to rethink how the components of the toolset fit together. Firstly we need to review and define standard file formats for linkage and debug information, secondly we need to reimplement the linker/librarian in C.

The key objectives of this development work are to

1. Provide file level interfaces between toolset components, and define how the tools should work with one another.

2. Provide a module structure for occam which separates interface syntax specification and the module implementation, and which provides a basis for separate compilation. References between separately compiled components and libraries should be machine independent in as far as is practically possible.

3. Support larger scale development. Modular system supporting information hiding: particularly restricting the scope of names within a large system.

4. Support mixed language systems. Review of calling conventions between different languages. We should aim to avoid wheeling in separate copies of the run time libraries for each scientific language process.

5. Relax the constraints on order of compilation. It is impractical to re-compile the whole system if a low level interface routine changes.

6. A library system which supports transputer variants and error modes in an efficient manner. Which provides proper scoping for names, and which provides an appropriate mechanism for code sharing

7. Improved interfaces and definitions for third party developers and the CAD system support.

System services

Host filing and terminal services are required both by our tools and by the programs developed by our tools.

We need to review the method by which host services are accessed by programs running, on transputers. The existing file and terminal servers should be based on a two layer protocol, the lower level routing packets across the transputer network and the upper level supporting requests to and responses from particular services.

We should provide improved versions of servers at the basic level we provide currently, and more explicit instruction on how developers can make their own services available across the network by adding to the upper level protocol. SUN developers for example are bound to want to windowing services for their application programs.

Access to these services currently makes programming networks of transputers much harder than necessary; it is awkward to embed terminal write statements in processes for debugging, tricky multiplexing is required to when running scientific language components over a network. Improvements in this area would make programming easier.

It would be appropriate to base the lower level protocol on the simulation work required for the virtual link development.

We require more flexible schemes for loading software to networks of transputers and running programs on networks of transputers.

This work should be done collaboratively with the CAD system developers where there is already some significant experience.

4.3 Integrated toolsets

Folding editor

We currently have no folding editor to support the toolset. We could write a new one in C. adapt the public domain EMACS editor, supply macros for the EMACS editor, adapt the TDS editor, leave users to use their own editor. We need a folding editor which uses text files and which runs on PC, VAX and SUN for our in-house development.

We should define a standard text file format for folded files.

Simple integration of tools on host

We need to ensure that our tools fit into the individual host systems and write command files and small programs to support this requirement. For example on the SUN we should be able to generate make files.

IPSES

Emphasis should be placed on fitting into other vendors support environments rather than building one from scratch ourselves. We need more details of the interfaces which would have to be satisfied; for PCTE or to fit within ISTAR for example. We need to go out into the world and see what is being used. There is scope for collaboration here.

Advanced Development Tools

User Interface Development (especially for debug, profiling and monitoring tools). We should look at object oriented languages for building these interfaces.

Network Monitoring Tools. What do Parsytec have at present? We should develop these ourselves on top of the message routing services.

Graphical design tools; animation of designs. There should be opportunities to develop such tools as part of collaboration.

Steps towards program proving.

We should pick up GRAIL, make it work with the SUN toolset, see what SUN developers think of it. This would be the fastest way to provide a profiling tool for occam.

4.4 OCCAM support

OCCAM is a unique INMOS offering, we must continue to develop it. We should look towards exploiting the OCCAM formalisms. We should spend effort optimising the OCCAM compiler code generation.

OCCAM compiler in C

Considerable effort has been made on this compiler already. It offers a number of advantages over the existing occam compiler; more portable, executes faster, provides multiple error detection in occam source, provides slightly improved code efficiency, provides a better basis for further code optimisations and support of new transputer variants. The first phase of this development should produce a replacement for the current toolset compiler.

OCCAM compiler/configurer in C

The second phase of the compiler development will be to extend the compiler to perform the configuration functions. The current configuration language restrictions will be removed; in particular all code for a given processor will not need to be within a single SC, scientific language components can be directly imported.

OCCAM profiler

A profiler is an obvious omission from our current toolset. It is a useful tool for developers wishing to tune transputer performance. We should build a portable OCCAM profiler with a view to extending the profiler to mixed language programs. It would be best to start with the GRAIL work before designing the profiler from scratch.

OCCAM for non-transputer targets

1. Helps development on hosts like diskless SUN nodes. Perhaps running occam using the simulator will be sufficient if not much better.
2. Helps spread occam. No need to invest in special hardware to write and run occam programs. Give it away to universities.
3. Improves the portability of occam. Gives us an option to use occam rather than C for hosted software.

On balance here it seems to be a good thing to do if someone else will do it for us. We could provide sources for occam compiler in C and kernel for SUN or VAX under some agreement. Meiko expressed an interest in producing a 68000 (and SPARC?) version of the new occam compiler.

OCCAM language development

OCCAM 3? Records, Recursion, Modules, ...?

4.5 Scientific Languages

We should have full source control of the key scientific language compilers so that we are able to improve the code generated, to support new silicon directly and quickly, and to exploit our position as silicon manufacturer.

Much compiler development must happen outside of INMOS. We should actively promote this development work. Do we have worldwide white pages, how up to date is it?

Currently we spend little development resource in this area; Anthony has spent significant amounts of time negotiating with suppliers or fending off customers in this camp. We very much need to build on and extend our compiler expertise.

We should buy copies of other people's compilers and evaluate them both for performance and as products.

C

Assuming appropriate business terms can be agreed we should produce a C compiler and debugger based on the Perihelion C compiler. The compiler will provide a direct replacement for the Lattice C compiler; the debugger

should integrate in a seamless manner with the occam debugger. Perihelion only has a very crude debugger at present. If the Perihelion deal falls through we should find another source supplier as soon as possible.

We should continue to optimise the code generation of the C compiler.

A parallel C compiler?

FORTRAN

We currently have no alternative for the Lattice FORTRAN. The only other contender here is Meiko? Ideally we would have C sources for the compiler!

The FORTRAN compiler needs to be considered more in the context of the accelerator market than the development market. I have recently had a request for a VAX FORTRAN compiler for transputer.

Parallelising FORTRAN compilers are emerging in the US.

PASCAL

ADA

Other Languages

LISP? Modula-2, Prolog, C++.

There are a lot of COBOL programs.

4.6 Board and Hardware Support

The software group should be responsible for the board support software. The software will be developed jointly by board and software engineers, the software group will be responsible for masters, archiving and delivery to manufacturing.

Module Motherboard System

The motherboard configuration software should be distributed as a standalone product with the motherboards. There are extensions required to this software to support connections across boards. This software should be supplied as a standalone product with the motherboard.

Memory Interface and EPROM support

Only support for these is through the TDS. We must make some provision for these facilities for VAX and SUN developers using the toolset.

Ethernet support

We are currently planning to provide low level packet passing procedures which allow packets to be passed between Ethernet boards.

We need to support standard communications protocols over Ethernet. Investment in obtaining and building on expertise in this area is very important.

Board interface drivers

4.7 External developers

We should create explicit products available to, external developers.

New hosts

We should provide explicit support and documentation for developers wishing to port our software to their hosts. We currently provide source releases of our software. We should improve the technical documentation of what needs to be done.

External compiler writers

The compiler writer's guide, standalone compiler implementor's guide, and T414 simulator should be packaged as an identifiable product to support external compiler writers.

Simulators for other transputers: T8? This is required both for in-house and external compiler writers.

4.8 Source Releases

In general our software sources should be made available under licence. This should be produced to meet internal requirements as a matter of course. We

currently have some difficulty with our VAX and SUN hosted products; in these cases we have to supply the host compilers in an unsupported form.