SIMMER: Software and Systems Integration Modelling Metrics and Risks (Getting to Level 4)

Brian Chatters
ICL, Manchester, UK
Peter Henderson
University of Southampton, Southampton, UK
Chris Rostron
ICL, Manchester, UK

Introduction

This article documents the mid-term progress and provisional conclusions of SIMMER; an ESSI funded Process Improvement Experiment. The overall objective of the experiment is to produce a more effective means of planning and controlling complex software and systems integration projects.

In order to remain competitive, ICL (as well as many other companies) needs to continually improve its predictability of costs and schedules for integration projects, to reduce time to market and to reduce costs without detriment to the quality of the products. The developments of our complex software and systems rely more and more on using commodity components and collaborations as a way to meet these business objectives. The ability to accurately predict effort and time scales and the ability to keep within budget is becoming increasingly difficult in such projects.

The specific purposes of the experiment are to demonstrate the applicability of the "Cellular Manufacturing Process Model" (CMPM) technology to a business critical, live software and systems development project and to develop and tailor the model and associated metrics to improve the project management processes.

Starting Scenario

Over recent years, software and systems development has become more complex and the trend has been towards the use of bought-in components. There is an expectation that, by buying in components, the time to bring a system to market can be significantly reduced. However, the use of third party components introduces a number of unknowns into the development activities, increases risks and jeopardises delivered quality. This fact is particularly true when the component is software that may not have been exposed to the specific operational environment previously, often leading to performance problems.

An internal assessment of our development processes (incidentally, achieved by participation in the SPICE/2 trial) identified the need to improve:

- the supplier management process - to better integrate the supplier's engineering processes with those of ICL, particularly in the area of support during software and systems integration
- implementation of organisational wide process metrics to help understanding and improve predictability

SIMMER makes a significant contribution to addressing these two areas for improvement. Traditionally, ICL has used the V-diagram waterfall life cycle model to plan and control software and systems development. Estimates of effort and time scales are based on an understanding of the architecture of the solution, and expert opinion of the degree of difficulty and potential problems likely to be encountered during the integration activities. The availability of resources and the team size is also taken into account when making the estimates.

A number of alternative life cycle models and development methods, such as DSDM [7] and Boehm's spiral model [2], are now in existence but none of them adequately address the issue of managing complex integration of third-party supplied components.

The CMPM, developed jointly by ICL and Peter Henderson of Southampton University, is a more appropriate model for the changing business. It provides a better means of capture and metrication of the interfaces between the contributing supplier, development and integration activities, enabling earlier and more comprehensive verification and validation of software products.

The Cellular Manufacturing Process Model

General Description

The CMPM is defined to be a set of “Manufacturing Cells” with the relationships between the cells described as a set of metrics. The model is based on Watts Humphrey's network models of software development [4], and on the “value chain” model developed by Michael Porter [5].

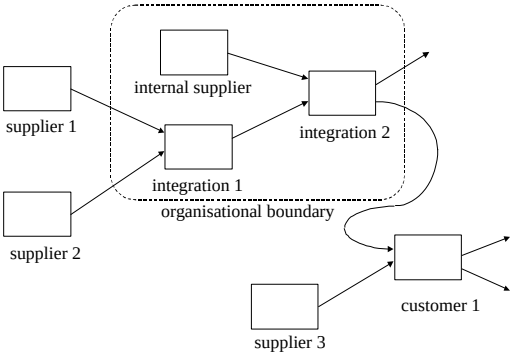

Fig.SIMM1: The Cellular Manufacturing Process

The model is based on a view of products that are integrated from a mixture of bought-in and self-built components (Fig.SIMM1). In this context, system integration is defined to be those activities which identify (and specify) components and develop "glue" to bind them. Some components will plug directly in (that is, they will not require any additional glue). For such components, the choice of one influences or restricts the choice of the others. The nature and quantity of glue required is a significant property of the system design. Each integration activity is defined as a cell within the CMPM. The model is clearly hierarchical. Each cell can be a component in a higher-level integration activity. Each component used by a cell can, itself, have been integrated from lower level components, either by in-house development or supplied by a third party.
Products with hierarchical structures lend themselves to being developed and built in a network of manufacturing cells. Each cell is responsible for one level of integration. The cell receives components from suppliers, makes some components locally, glues the components into an integrated product (which is tested to output standards) and ships the integrated product to a customer or to the cell performing the next level of integration. Note that the traditional software development process can be defined as an integration cell within the definition of the CMPM. It makes all components (lines of code) in-house and glues (compiles and builds) them into a software module or software product.
The behaviour within a cell can be as formalised or as ad hoc as the business or product demands. The measurement regime of the CMPM (the metrics) is not dependent upon detailed knowledge of how each cell performs its integration tasks, only how it meets its external obligations.

Metrics within CMPM

Six metrics are defined by CMPM (Table 1). Fig.SIMM2 illustrates how these metrics are associated with each cell within the network. The reassuring thing about these metrics are that they are clearly at the

management level, not down at the detailed code level.

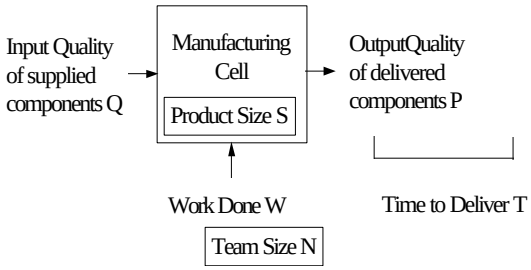

Fig.SIMM2: Metrics Associated with Each Cell within CMPM

The measurements of the quality of the input to and output from each cell are converted to percentage reliability measures (100% indicating total reliability). The values will be based on actual measures of the quality or based on an expert judgement from the project team members. Clearly, the quality metrics are directly related to project risks.

W	Effort	*The work done on each cell, in net person days (excludes project management overheads). Factors affecting W include unforeseen problems, changes in requirements, poor estimates of S, late handovers, and product problems (the number of and the cost of resolution that may be further complicated by supplier support interfaces).*
T	Elapsed time	*The schedule of deliveries of components from each cell, in elapsed working days (excludes weekends and public holidays).*
N	Team size	*The average team size (W=T*N)*
S	Size	*A measure of the size of the product, in "Standard Integration Units" (SIU's) - in the context of integration, size is determined by "hard" cost drivers which are quality independent. The drivers cover costs for building systems, installing products, regression testing, producing project infrastructure, and making glue/in-house components.*
Q	Input quality	*The average quality of incoming components, on a scale of 0 to 5 - the costs incurred as a result of the values assigned to this metric (and the required output quality P) are determined by "soft" drivers which are dependent upon the issues, risks, and problems inherent in the supplied components.*
P	Delivered quality	*The target quality of outgoing components, on a scale of 0 to 5*

Table 1: CMPM Metrics

Predicting Costs and Time Scales Using CMPM

COCOMO (Constructive Cost Model) is a model for estimating software effort, cost and schedule for a number of different types of software development projects [1]. COCOMO predicts a relationship between S, W, N and T. CMPM postulates that a relationship also exists between Q, W and P. The conjecture is that effort can be modelled against size (S) but that input and output quality (P and Q) will also have a significant impact on any predicted costs. That is:

$$W = f(Q,P,S)$$

Where W = effort, Q = input quality, P = output quality and S = size.

The function will demonstrate the behaviour that, if the target P increases, either W will increase or Q needs to be increased. Similarly, if S increases, then W or Q will need to be increased in order to achieve the same level of output quality (Fig.SIMM3).

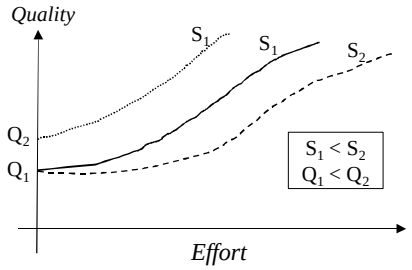

Fig.SIMM3: Illustration of effect of S, P and Q on W

The function can be determined from historical data and used to predict costs and time scales for future projects.

Relationship of CMPM to Capability Maturity

The effectiveness of CMPM as a predictor of costs will depend upon how well an organisation understands its processes. Only then can it make reasoned estimates of the size of the tasks that need to be performed to achieve its deliverables. There are (at least) two key factors that affect how well an organisation can make predictions.

Firstly, it needs to ensure that its activities are **managed** effectively. To achieve effective management, it needs to gather data on the performance of its activities and to analyse the data to identify potential predictors of future performance. Basic data collection of cost, schedule and problems is a requirement of level 2 of SPICE [6].

Secondly, it needs to be able to define the tasks that need to be performed within a cell. The CMPM allows the description of a cell to be as formal or as ad hoc as the business demands. However, to gain maximum benefit, our experiment demands a level of description which enables project manager's to plan detailed tasks in a repeatable and, to some degree, a predictable manner. To achieve this, we have advocated that the project's processes need to be **defined**. That is, they need to operate at level 3 within the SPICE model before the application of CMPM can have full effect.

With these two key building blocks in place, CMPM can then be used to further improve the predictability of the performance of a project (hence, the reason for the subtitle of this article: "Getting to Level Four").

Plans and Expected Outcome

Objectives

The overall business objective of this experiment is to produce a more effective means for the design and planning of complex software and systems integration projects, involving the use of commodity components, collaborations with third parties, or the reuse of existing components. The new process will improve the understanding of how to exploit the components in a new project and hence, help to mitigate against risks and to improve predictability.
Specifically, the aims of the experiment are:

- to demonstrate the applicability of the CMPM technology to a business critical, live software and systems development project;
- to develop and tailor the model specifically for software and systems integration and validation;
- to develop appropriate metrics to support the project;
- to quantify the benefits as a result of applying the technology.

Benefits

The specific benefits are expected to be:

- **More accurate predictions of costs and schedules.** The major expected benefit by the application of the new method will be to significantly improve the accuracy of the predictions of project costs and schedules, thus enabling more realistic forecasts. This improvement, in turn, will lead to better overall business planning and enable the organisation to have more confidence in ensuring that its return on investment will be protected.
- **Improved Product Quality.** The new method will ensure that a better focus is given to quality requirements and ensure that preventive action is planned to mitigate against potentially high-risk components. This focus, in turn, will ensure better test and validation coverage. The process improvement will have the effect of significantly reducing the number of customer detected bugs once the product is released, resulting in a reduction in support and maintenance costs and improved customer satisfaction.
- **Reduced Time to Market.** The biggest cause of delays to the planned schedule of a project involving supplied components, is the time it takes to resolve unexpected problems. A better understanding of the quality of the input and its impact on the project will enable such problems to be pre-empted and resolved much faster, thus reducing the time to market.
- **Exploitation of CMPM.** The results of the experiment will allow the CMPM to be enhanced and exploited as a key aid to improving software and systems supply, development, and integration processes.

The method will be able to be applied to any organisation, which designs, integrates or tests complex software systems, using commodity products, collaborating with third parties, or reusing existing designs and components.

Definition of the Baseline Project

The baseline project is part of a broader programme aimed at providing platforms, which exploit emerging technologies and meet the future needs of ICL's customer base. A number of systems management products are included in the system and a minimal amount of non-invasive integration is undertaken to make them easier to use and to improve their RAS (reliability, availability, serviceability) characteristics.
The project is split into seven teams with an average size of eight people. The software is a combination of COTS products, in-house development, and collaborative development with partners. Software products, for example for backup, performance monitoring, printing and event management, are included. The suppliers of the relevant hardware modules provide platform specific software products (e.g. peripheral drivers). Regular, incremental, deliveries of the system are made to the customer base.

The Plan

1. A set of baseline measures will be established by analysing historical data. Variations between predicted and actual measures will be recorded. A log of problem reports from the baseline project activities will be set up, identifying where the problem was found, its severity, which component, process, or supplier caused the problem and how long it took to resolve it.

2. The generic CMPM will be used to define a specific model, tailored to the baseline project. The baseline project activities will be split into cells, which reflect the interfaces between the development teams for each incremental release of the system. A set of metrics will be defined and collected for each cell and initial estimates will be made of the values for these metrics. The input quality levels (Q) of the supplied components will be determined by review and discussion with the project staff and the output levels (P) will be defined by the project requirements.

3. The incurred effort (W), team sizes (N) and time scales (T) of the baseline project will be monitored as part of the normal project management activities. Every month, actual measures or revised estimates will be made of size (S), input quality (Q) and delivered quality (P), causing revised estimates of total effort to be made, if necessary. Additional data will be collected from the problem management activities and the problem log will be updated.

4. The metrics data collected by the baseline project will be used to carry out ongoing analyses of the relationships between effort (W), size (S), and quality (Q and P). Results of the analyses will be used to evolve a cost estimation model, which will be piloted by the baseline project throughout the lifetime of the experiment to predict effort and time scales for future activities.

5. Data collected throughout the experiment will be analysed and the differences between the predicted and actual values, when compared to the measurements taken prior to the experiment, will be used as a measure of the effectiveness of the process.

Progress against the Plan

Establishment of Baseline Measurements

Effort is recorded and reported each month by project members and gives a breakdown for each incremental release. This breakdown was not available for the historical data and so each team made an estimate on the contribution made to each incremental release.

Representatives from each of the teams also estimated values of S, P and Q. Q was estimated by using a checklist to identify the drivers for and potential causes of poor input quality (Table 2). The representatives were asked to give a rating for each of the suppliers to the cell, on a scale of zero to five, of the degree to which they agreed with the statement concerning the attribute (zero = totally disagree; five = fully agree). An overall rating was derived. This rating was left to the judgement of the team representative because not all suppliers have the same impact on the project plans. However, in many cases, an average was computed. Output quality was also estimated in a similar way, based on hindsight of problems experienced during the implementation of the cell.

Attribute
A. The impact of the product and development process characteristics are fully understood
B. The inherent quality of component will guarantee no problems for your activities
C. The supplier is easy to work with
D. We have full synergy with the supplier
E. The supplier is dependable

Table 2: Attributes used to measure Q and P

Estimation of size S was based on a "standard integration unit" (SIU). All integration projects need to carry out standard tasks such as establishment of the project environment, building systems, installing products, and regression testing. The total amount of effort is also dependent upon the number of supplied components, the amount of in-house development, and the number of incremental builds. The costs for these basic activities are independent of the input quality of the supplied

components and the output quality of the integrated system. As an initial attempt to set values of size for each cell, one specific (arbitrary) integration cell was defined to be of size 100 SIU's. Relative values of size were estimated for all the others cells through a facilitated session with all the team leaders and the overall project manager. This approach ensured consistency in the way that size was estimated.

Application of CMPM to the Baseline Project

The project is structured as a network of cells which are classified as “software development”, “hardware development”, “systems integration”, or “build and release” to reflect the different nature of the activities that are carried out within each cell. At this point, it is unclear whether the different categories of cell will display different behaviours, which may need to be reflected in the cost estimation algorithm. Cells are defined for the contribution each team makes to each incremental delivery of the system. Not all teams contributed to every release and, in some teams, activities were carried out in parallel to support a number of releases.

Data Collection

Standard project control processes demand regular progress reports identifying the percentage of activities achieved, the status of the key milestones, issues, and spend to date. The process has been enhanced to ensure predicted and actual values of the six CMPM metrics are reported as well. For each incremental release, each team (cell) within the project estimates values of S, Q and P. The values are then used to estimate W and T from a given N. Typically, a release date is set as a requirement and resources (N) are made available to underpin the milestone, subject to budgetary constraints. As part of the regular reporting, actual or re-estimated values of S, Q and P are recorded and used to review the estimates of the outstanding effort needed to complete the release.

Development of Cost Estimation Algorithms

Early experimentation of the model using the historical data highlighted some inconsistencies in the way that Q and P had been estimated. For example, one team had estimated the same values of S, P and Q for three increments even though the actual effort varied widely. The assumption of the model is that S, P and Q are sufficient synthetics to enable effort to be estimated. If this was the case, equal values of S, P and Q must yield the same estimates of effort. As a consequence, a more objective means of measuring Q and P is required and revised definitions of Q and P are being implemented.

For the development and integration cells,

- Q=5 is defined to be "all components are fault free"
- Q=0 is defined to be "all components are faulty"

A component is defined to be the lowest level unit that, if faulty, will have only one fault. This revised definition requires a value to be set on the number of components and, for practical purposes, the value is set by estimating the maximum (realistic) number of expected faults. That is,

the estimator is asked to consider the worst case scenario which, typically he would do as part of the risk assessment process. For example, a software module may be estimated to have, in the worst case, a hundred faults. Thus the number of "components" in the product is defined to be one hundred.
For the build and release cells,

- Q=5 is defined to be "only one component build is required, per component" (typically, components will be batched together to reduce the total number of builds required)
- Q=0 is defined to be "every component is faulty and requires a rebuild to correct the fault"

In the context of integration, a fault equates to any problem that incurs cost to correct and thus includes process faults, product faults, and issues arising from dealing with suppliers.
P is defined to be the level of achievement against predefined release criteria. If all criteria are fully satisfied, P=5. The project release processes include a formal review of achievement against the release criteria, thus enabling a value of P to be assigned.

Measured Results and Lessons Learned

Baseline Measurements

Table 3 provides an example of the historical data for one team (cell). Each record gives the values of the metrics for the work done to deliver components to each incremental release of the system. The values of Q and P are the subjective values obtained by interviewing team members. As already stated, the plan is to introduce more objective measures of P and Q based on the number of problems expected/experienced.

Project	***W***	***T***	***N***	***S***	***Q***	***P***	***Problem***
software team a	548.5	86	7.6	50	3.1	4	11
software team a	31.92	21	1.8	10	3.6	4	0
software team a	723.1	169	4.8	40	3.6	4	23

Table 3: Baseline Data

Table 4 summarises the actual and estimated values for W, T and N for each incremental release of the system. Much of the extra cost is incurred due to quality problems. The summary shows slips of between 52% and 148% and overspends of between -24% (underspend) and 83%. Note that this data is used for internal planning purposes and does not reflect commitments made to customers. The data provides a starting point against which the results of the experiment can be compared.

	Actual			*Forecast*				
Release	**W_a**	**T_a**	**N_a**	**W_e**	**T_e**	**N_e**	**(W_a/W_e)%**	**(T_a/T_e)%**
release a	2284	149	28	1320	60	28	173	248

release b	440	84	7.9	576	40	18	76	210
release c	2758	213	15	1512	140	14	183	152
release d	793.4	213	4.2	640	140	4.5	124	152

Table4: Summary of Metrics by Release (Actual vs. Forecast)

Experiments with the Model

The collection of the historical data is providing a better insight into the behaviour of the software and systems integration process. Various hypotheses have been postulated and experiments have been undertaken to model the behaviour using curve-fitting techniques. Fig.SIMM4 is an example of the results of one such experiment. It shows the distribution of the values of two constants (b and k) used in a COCOMO-type formula to predict effort from the historical values of S, Q and P.

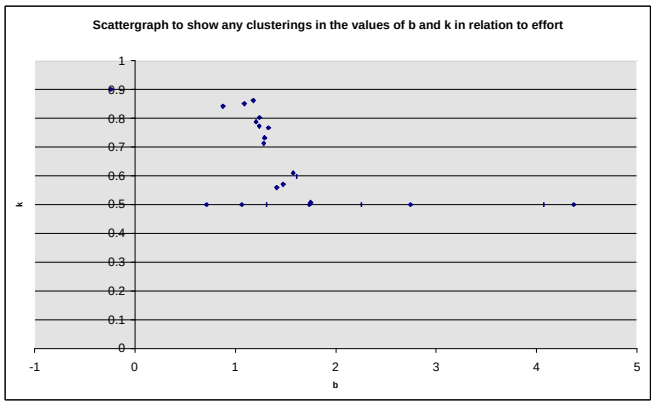

Fig.SIMM4: Computed values of b and k

Some of the reasons for the scatter have already been discussed but these early results are encouraging. They suggest that S, Q and P do indeed influence W and a relationship probably exists that can be modelled and used to predict future project behaviour.
Future integration teams will be multi-disciplined and thus, the CMPM may reduce to two cells - one for integration and one for build and release. This change in organisation should help to reduce some of the anomalies, as all teams should exhibit similar behaviour. However, the size of the teams will be of the order of existing teams (about 6-8 people) and the difficulties in estimating costs for small teams need to be addressed.

Lessons Learned

Even if we are unable to find values to support the theoretical model, there is still considerable benefit (which should not be underestimated!) to the project in using the metrics set to manage its business more effectively. The metrics will support the management practices that are required to achieve level 4 on the SPICE maturity scale.
Specific lessons learned so far include:

- The experiment only works with the full collaboration from the project.
- The metrics need to be simple and easy to collect. Data collection

needs to be established as part of normal business; carried out by project staff. Build on what projects currently care about and are likely to have data. Then get disciplines in place to capture and analyse the data.

- Metrics provide objectivity into the project management decisions. The act of measuring alone can bring about improvement.
- Subjective measures are better than no measures at all but they are of limited use. They allow projects to think about the issues but they do not allow the development of cost estimation models.
- There is clear benefit in using the extended metrics set (S, P, and Q) in managing integration projects. The issues to be managed in an integration project are much broader than product problems (bugs) and all problems that impact costs significantly need to be considered. Many process problems can have a bigger impact on project overruns (for example, when working with suppliers).
- Metrics need to be based on a sound, objective basis - therefore, a project needs the equivalent of SPICE level 2/3 management practices in place before the full benefits of SIMMER can be realised.

References

[1] Boehm B. et al, The COCOMO II Model Definition Manual, University of Southern California, USA, 1996

[2] Boehm B.W., A Spiral Model of Software Development and Enhancement, in: *IEEE Computing 21(5)*, pp. 61-72, 1988

[3] Chatters B.W., Henderson P., Rostron C.J., The Cellular Manufacturing Process Model: Planning a Complex Software And Systems Integration Project, in: *Proceedings of the European Software Measurement Conference*, pp. 559-564, Technologisch Instituut vzw, 1998

[4] Humphrey W.S., Managing the Software Process, Addison-Wesley, 1990

[5] Porter M.E., Competitive Advantage: Creating and Sustaining Superior Performance, The Free Press, New York, 1985

[6] SPICE - PDTR ISO 15504, Software Process Improvement, Part 2: A Reference Model for Processes and Process Capability, Version 2.0, 1996

[7] Stapleton J., DSDM: Dynamic Systems Development Method, Addison-Wesley, UK, 1997

Appendix 1: Author Profiles

Brian Chatters

Brian Chatters is a Software and Systems Engineering consultant within ICL High Performance Systems. He is responsible for ensuring continuous development of the organisation's software and systems engineering capability and he has developed and introduced a framework based on SPICE and CMM, to discharge this responsibility. He has worked in the software industry for over 30 years in various roles including programming, strategic design and project management. In the last 12 years, he has focused on quality management and process improvement. He graduated from Bristol University with an honours degree in mathematics. He is a fellow of the Institution of Electrical Engineers, a Chartered Engineer and an active committee member of the BCS SPIN (UK) Special Interest Group. He is also a qualified ISO 9000 auditor and a qualified SPICE assessor.

Peter Henderson

Peter Henderson is Professor of Computer Science in the Department of Electronics and Computer Science at the University of Southampton in the UK. Prior to his move to Southampton in 1987 he was Professor of Computer Science at the University of Stirling, also in the UK. Henderson is also a visiting ICL Fellow. He is head of the Declarative Systems and Software Engineering Research Group (see http://www.dsse.ecs.soton.ac.uk/) which combines research interests in Software Engineering, Formal Methods and Programming Languages. His own research includes executable specifications, component-based systems, process modelling and the software development process. He has consulted for many companies, including ICL, on diverse topics in Software Engineering but in particular on Software Development Process Improvement. He has published two books and about thirty papers on Software Engineering. Currently Henderson is national co-ordinator for an EPSRC (Engineering and Physical Sciences Research Council) research programme entitled Systems Engineering for Business Process Change which has a legacy systems, COTS, component-based software development theme within it.

Chris Rostron

Chris Rostron is a Development Manager within ICL High Performance Systems. He is responsible for the planning; release management and quality assurance of a major UNIX based project. He has worked in the software industry for 30 years in various roles including programming, software design, release and configuration management, support including Product Introduction, business planning, and Quality assurance and project management. In the last 6 years he has focused on release and configuration management, planning and Quality Assurance. He gained qualifications in Cartography before joining the computer industry. He is a qualified ISO 9000 auditor and a qualified SPICE assessor.

Appendix 2: Company Descriptions

ICL is a leading supplier of IT systems and services. Operating in over 70 countries and employing over 19,000 people, the group's revenues for 1997 were £2,447 million generating a pre-tax profit of £30.0 million. The company implements IT systems for major projects and provides innovative services to a range of industries covering amongst others, retail, finance, travel, telecoms and utilities together with education and local and central government sectors. Its services include outsourcing, helpdesks, network services, inter/intranets, electronic commerce, interactive kiosks, smart card systems, digital cities and web sites. ICL plans to relist on the stock market in 2000.

ICL's website: http://www.icl.com

ICL's High Performance Systems Division (HPS) is responsible for the development, sales and marketing of enterprise-scale solutions and services. Working together with ICL business operations world-wide, our customer offerings are based on the Trimetra range, providing data centre solutions running OpenVME, Windows NT and UnixWare, and on i500, ICL's open directory software.
HPS is at the forefront of new technologies enabling the information society. These include interactive media servers, WWW-enabled software and advanced data centre solutions based on combining leading ICL and partner technologies and expertise.