RENDEZVOUS

NYHETSBLAD FÖR SESAM
Försvarssektorns Adaintressenters Användargrupp för Software Engineering

Nr 2 June 1998

NEWSLETTER FOR SESAM
The Swedish Defence Sector Software Engineering with Ada User Group

Special Issue in connection with

Ada-Europe '98

International Conference on Reliable Software Technologies
Uppsala, June 8-12

Contents:

SESAM

The Defence Sector Software Engineering with Ada User Group

What is SESAM?

SESAM was created in 1988 to organize and stimulate cooperation and collaboration in Software Engineering between the Swedish Defense Industry, the Defense Materiel Administration (FMV) and the National Defense Research Agency (FOA).

According to the governing agreement between the participants, the purpose of SESAM is to promote, through organized cooperation between the members of the User Group, dependability and efficiency in development and sustainment of Ada software within the Defense Sector. Within this framework SESAM shall adjust, profile and renew its activities with regard to changing technical and other circumstances of importance for the area of interest.

* SESAM in general shall work to disseminate information regarding factors which influence the possibilities for dependable and efficient development of software systems. In particular the significance of Ada in this connection shall be clarified.

* SESAM in its activities shall continously monitor the opportunities to collect, create and disseminate objective measurement and other results and experiences generated in use of software engineering principles and the Ada language.

* SESAM deals with the process of developing and sustaining software systems. Implicit in this is of course that processes used shall assure that resulting products meet sought for properties. Product properties dependant on the processes therefore are of primary interest in SESAM's activities.

* SESAM in its activities, shall attach great importance to facilitate the co-existence of Ada-programs and software written in other languages. In particular issues arising in the use of COTS-software shall be considered.

* SESAM shall, wherever possible, set concrete and quantifiable goals for its activities during defiend periods.

SESAM is governed by a Council with participants representing the Members. The council has at its disposal an Executive Committee and a Secretariat.

The present chairman of the council is Ingemar Carlsson, FMV, ph. 08-782 6516

Executive Committee

Christopher Bengtsson , FMV
chben@tranet.fmv.se
Roger Brandt , FMV
robra@fmv.se
Ingemar Carlsson, FMV
ic@tranet.fmv.se
Sune Ekfeldt, Enator Telub AB
sune.ekfeldt@enator.se
Billy Johansson, CelsiusTech Electronics AB
lifo@celsiustech.se
Björn Källberg, CelsiusTech Systems AB
bjkae@celsiustech.se

SESAM activities are primarily carried out in a number of Working Groups. The following groups are currently active:

Process/Metrics
Göran Anger, Industrilogik
anger@L4i.se
Programming
Magnus Ericson, Ericsson Saab Avionics
Magnus.Ericson@esavionics.se
Real-time systems
Gilbert Kennedy, Saab Dynamics
gilke@weald.air.saab.se
Systems
Håkan Edler, CTH/Datorteknik
edler@ce.chalmers.se
Reuse
Barbro Sjöland, Sjöland och Thyselius Datakonsulter
barbro.sjoland@st.se

The Information Committee
Christopher Bengtsson, FMV
chben@tranet.fmv.se

Who can participate in SESAM?

The Members of SESAM are Swedish Companies, Organizations and Agencies with connection to the Defense Sector.
Members are divided into the following cathegories.
- full member
- working group member
- information member

Admission to SESAM

Admission for all member cathegories is decided by the Council.

For admission as full member or working group member status as Contractor to FMV is required. Further a written pledge to carry out the obligations as full or working group member is required.

For admission as information member (only receives Rendezvous and seminar invitations), status as Contractor to FMV or position as Agency of the Total Defense is required. The Council, however, can admit another party as information member.

For Application for memebership, please contact to the Secretariat.

SESAM-Secretariat
Ms Anna Kåsjö, FMV:INFOSYST, 115 88 STOCKHOLM
Phone: 08-782 6745, Fax: 08-66 77 392
Email: alkas@tranet.fmv.se

Full members of SESAM: Bofors, Celsius Aerotech, CelsiusTech Electronics, CelsiusTech Systems, Combitech Software, Defense Materiel Administration (FMV), Defense Research Agency (FOA), Enator Telub, Ericsson Microwave, Ericsson Saab Avionics, Kockums, Saab, Saab Dynamics.

SESAM salutes Ada-Europe

This issue of Rendezvous appears in English in order to illustrate for participants from abroad in the Ada-Euorpe'98 Conference at Uppsala, that Ada is alive and kicking in the Swedish Defense Community. The SESAM User Group is trying to contribute to showing the way forward for Software Engineering with Ada.

We have been active now for ten years and it was very interesting to note that in the discussions recently about the wording of a modernized agreement between the members of SESAM (primarily because the mergers and acquisitions which have taken place during the last few years have changed the composition of the member roster significantly), when the question was asked whether we ought not to drop Ada from the name of the User Group, there was a unanimous decision not to do so; Ada is still the fabric that holds the Group together.

This does not mean that we do not acknowledge that there are other forces afoot that will also contribute to the advancement of Software Engineering in the Defense Sector. We try to keep an open mind and be prepared to promote interest in other promising technologies that are emerging.

This year for example, SESAM has chosen as its common theme between all the Working Groups ”Software Engineering with Java and Ada”, since we see a great potential in Java and in the combination of Java and Ada. By yearend, we hope to be able to give our member companies and agencies a comprehensive review of the state-of-the art and prospects of Java and Ada as seen from the various vantage points of the Working Groups.

The use of Ada in Swedish Defence Systems is very strong in i real-time oriented applications in Weapon Platforms, Weapon, Sensor and C3I systems and in Modeling and Simulation, but quite week (or non-existant) in Information Systems. This latter situation is something we hope to be able to change with the advent of Ada 95. Perhaps some assistance in this direction will come from the fact that there is just this month out Ada 95 bindings to the EasyDB true Object Data Base Management System, both developed here in Sweden, which will allow efficient development of applications that can take advantage of an ODBMS. (See separate article and the AIS/SESAM CD-ROM distributed to participants in the conference.)

Regarding Ada 95, we had one of the AJPO Ada 95 Early Adopter projects here in Sweden in the Ericsson Microwave Combat Control System for the new Bofors BAMSE Surface/Air Missile. The experience with Ada 95 from this project is very positive, as are the results from the FBSIM Force Level Simulation System described elsewhere in this issue.

The use of Ada in aircraft systems has so far been very low due to a very specific reason, namely that the new Swedish Air Force Combat Aircraft the Saab JAS 39 Gripen uses a specially developed inter-optimized triad consisting of a the PascalD80 (”Ada-like”) language, the D80 Multiprocessor Computer and the PUS80 Integrated Development Environment, whose development was initiated (in 1978) before Ada was available as a viable alternative on space-constrained airborne computers. Now it is time to replace D80 with a new computer and we can read in this issue that the MACS computer from Ericsson Microwave will be able to run both PascalD80 and Ada95 programs efficiently. However we have alreday had a very significant Ada inroad right into the core of JAS 39 through the new Fly-by-wire Flight Control System. Ada-Europe conference participants will be able to hear a paper given on this interesting Ada application by Bo Frisberg of Saab.

As another Swedish Ada-example, I would like to mention the Objects for Systems (O4S) Systems Development method which uses a subset of Ada 95 for executable descriptions of system structuring and object behaviour. With the newly released Tofs tool, described in a separate article , this should be a very interesting combination for getting a better handle on on some of the more elusive systems engineering aspects of complex systems.

The probably most significant new Ada development in the Swedish Defense Sector, is the contract let last month by FMV, the Swedish Defense Administration, to CelsiusTech Systems for the CETRIS C4I2 system for the new Visby Class Corvettes, the complement to the Gothenburg Class Coastal Corvettes, which carry the CelsiusTech Systems Ship System 2000, widely acclaimed for being a prime example of a system developed in Ada with a Product Line Management reuse approach. The new system is described in an article by WG9 and SESAM member Björn Källberg.

We also have résumés from two recent conferences attended by SESAM members, the mammoth Java One and one on Requirements Capture and Traceability in Systems Engineering.

There is an interesting opportunity to investigate some very pressing Software Engineering matters in a new program for Defense Adaptation soon to be launched by FMV. There is not room for going into the details of these propsals here, so we are only listing their titles in a separate article, to indicate the type of questions SESAM members feel will be important for the next few years.

Finally we are happy to be able to re-publish two papers which have originally appeared in other connections. One is a paper on the design by the Swedish company Pitch, of a run-time infrastructure (RTI) implemented in Java for the new Modeling and Simulation High Level Architecture (HLA) standard. The other is the paper by Harold Lawson for the Binary Critic column of the February Issue of IEEE Computer on ”Salvation From System Complexity”. This is a very thought provoking article that we all need to give serious consideration, not least in the Defense Community.

SESAM wishes Ada-Europe success with the
’98 Conference!

Ingemar Carlsson
1997-98 Chairman of the SESAM Council

CelsiusTech wins Visby combat system contract

In April, CelsiusTech Systems was awarded a SEK 700 million-plus ($ USD 88.8 million) contract by the Swedish Defence Materiel Administration (FMV) to supply the combat management system (CMS) for the Royal Swedish Navy's new Visby-class stealth corvettes.

The order marks the first sale of the 9LV Mk3E system. An evolution of the earlier 9LV Mk3 command and weapon control system, the Mk3E adopts an industry-standard open-system architecture and uses mainstream commercial-off-the-shelf (COTS) products, such as Pentium PRO processors, a 100Mbit/s FDDI local area network, the Windows NT operating system.

Aboard the Visby class, the 9LV Mk3E CMS will form part of the CETRIS on-board command, control and communications system.

The CETRIS command support function and interoperability with the wider LIM Swedish Navy command, control and communications architecture will consist of the CMS, the MAST decision support aid and a communications system.

The new CMS will be based around the CeCOTS multi-function console. Developed by CelsiusTech and Norwegian company Hitec, it features dual flat-screen colour displays and intuitive human machine techniques.

The 9LV Mk3E re-uses the functionality of earlier 9LV Mk3 Ada applications software on industry-standard hardware. Applications are run under Windows NT (rather than the proprietary OS 2000 operating system) so as to facilitate easy migration to new COTS hardware.

Four Visby class Series I ships are on order from Karlskronavarvet. The first is due to be delivered in early 2000 and, following extensive trials, will become fully operational in 2002.

The software in the CETRIS system is based on reuse of the existing Ada software, the Mk3 generation. This software has been delivered to many navies over the world, and has a proven reliability, accuracy and functionality. The new console with its software, based on windows NT and developed completely by CelsiusTech is integrated in this system. The major part of the console software is still coded in Ada. However, the layer that handles the windows is developed using the Microsoft Visual C++ development environment, with its very good support for coding windows application.

/Björn Källberg

Object Data Base Management System now with Ada 95 Interface

The DBI Ada 95 interface is the Data Manipulation Language (DML) interface offered by the ODBMS (Object Data Base Management System) EasyDB for program development in the object oriented programming language Ada 95.

EasyDB is an object database management system from Basesoft Open Systems.

EasyDB is designed for distributed environments and with high performance in mind. It has a multi-client, multi-server architecture and supports distributed storage and access transparent to the user. It is also available as a single-user data manager (EasyDB Lite). EasyDB has well integrated language bindings to C, C++ and Ada and has also an interactive ad-hoc query language (O-SQL). It uses a conceptual modeling approach - Data Definition Language (DDL) - based on ERA-technique combined with object orientation and a graphical notation. EasyDB is implemented on a number of operating systems.

Ada 95 is a revision of the original Ada 83 language including amongst other features full object oriented programming (OOP) facilities, being the first ISO standard to do so. It has also been extended with better tasking facilities, a hierarchical structured library, greater ability to manipulate pointers/references and more flexible means of communication with other languages and systems.

The DBI Ada 95 interface has two main parts: A predefined class library and a view generator program.

The class library interface provides all general functionality of the data base manager interface. The view generator program generates schema-specific classes, types and constants which extend the existing class library with faster, more specialized functionality.

A DBI Ada 95 application is created in the following steps:

1) Create a schema to be stored in the Data Dictionary.

2) Define the application view (subschema)

3) Generate the ready-to-use schema-specific classes, types and constants

4) Write the Ada application program utilizing the DBI Ada 95 library and the generated classes to create a data base and manipulate its content. It is possible, and sometimes very useful, to write an application without using generated classes at all. This is called dynamic DBI versus static DBI where you use the names defined in the schema.

A limited capacity version of EasyDB Lite single-user data manager with the Ada 95 Interface,including full descriptions as well as course material on using ODBMS's in general and EasyDB in particular, will be included in the AiS-SESAM CD-ROM to be distributed to Ada-Europe'98 conference participants at Uppsala.

The EasyDB DBI Ada 95 interface was released in May of 1998 from Basesoft Open Systems AB, Stockholm (info@basesoft.se). Its development was sponsored by the Defense Materiel Administration (FMV).

Stig Lorenzi
Basesoft Open Systems AB

With Ada and Tofs towards Systems Engineering

Ingmar Ögren, Romet AB,

When Ada was first introduced in the Swedish defense community in the early eighties an urgent question was: how do we make the new language popular and how do we teach people Ada's principles?

An early idea was to invite Grady Booch. He came, gave an Ada course and introduced the principles of "compositive object orientation" as described in the book "Software Engineering with Ada".

This led to the idea that Ada could be used, not only for programming, but also to describe system structures and object behavior. Further that if we could simplify Ada's syntax the descriptions might be possible to explain to end-users with some hope of their understanding.

The idea resulted in a simplified Ada-based pseudo language, Adel (Ada-based DEsign Language) a syntax checker for Adel, called Adela (Adel Analyzer). Through the years, the language and the tool was used for various defense and industrial projects. One experience was the Ada-based pseudo language could be used, not only to describe software structures, but also to define complete systems, including operators and hardware.

Another experience was that the basic principle in Ada, with packages depending on each other is quite useful for systems engineering as it gives very clear structures which show how top-level parts of a systems depend on lower-level parts and on external supporting parts. This principle also proved to be quite useful for requirements' distribution through a system structure.

Two years ago development of the MS-DOS based Adela tool was discontinued and it was time to develop a new tool kit, using the experience from Adel and Adela. The language was updated to be more formal and named Odel (Object DEsign Language). It is now based on Ada 95, but with some important deviations such as omission of tasking to allow for formal and deterministic descriptions of systems with concurrent processes. The new tool, called Tofs (TOol For Systems), has the ambition to support engineering of complex systems where operators cooperate with software and hardware modules to complete Missions.

Some important qualities of the Tofs tool kit are:

- Unique Ada 95 based object structure to give overview and detail
  Tofs assists you in modeling your system as a set of objects, which are dependent on each other. This makes it possible to show a "tree graph" which gives a system overview. At the same time you can use Odel (Object Design Language), which is basically "formalized English" to define details.

- Integrated requirements and problem management
  Everyone agrees that requirements must govern system design. It is less obvious that design will cause requirements update and reveal problems. Tofs allows dynamic requirements management and problem management with distribution and tracing of requirements. It also includes matrices to couple fulfillment requirements to test cases and test results for each object.

- Mission objects help to concentrate systems work
  It used to be difficult to understand how the different parts of a system contribute to completion of the system's missions. With Tofs, you can create Mission objects and create a dependency structure, which shows clearly which objects contribute to each mission.

- Consistency checks and system level debugging
  With Tofs you define system structure and behavior in Odel (formalized English), including type definitions, to ensure that each concept in a system is defined. This makes it possible for Tofs to check consistency for your system. Also for you to walk trough your system with a “system level debugger” (The Odel Executor).

- Dependency analysis of critical systems
  Critical systems are traditionally managed with formal methods and techniques like FTA (Fault Tree Analysis) and FMEA (Failure Mode and Effects Analysis). With Tofs you get understandable formalism (formalized English) and the object structure allows application of FTA and FMEA on complete systems including operators, hardware and software modules. The result is that you get help towards creation of dependable systems for critical applications.

- Tailorable documentation, which satisfies standards
  Tofs is delivered with a set of MS Word templates to help you satisfy the MIL-STD-498 and the IEEE/EIA 12207 standards. The templates are easy to tailor to satisfy the requirements from your organization and project.

- Low cost platform
  Tofs runs under MS Windows NT on a regular PC and is integrated with MS Office to keep down hardware, software and education costs. As systems engineering must be a team effort, each Tofs license allows ten users in a PC network.

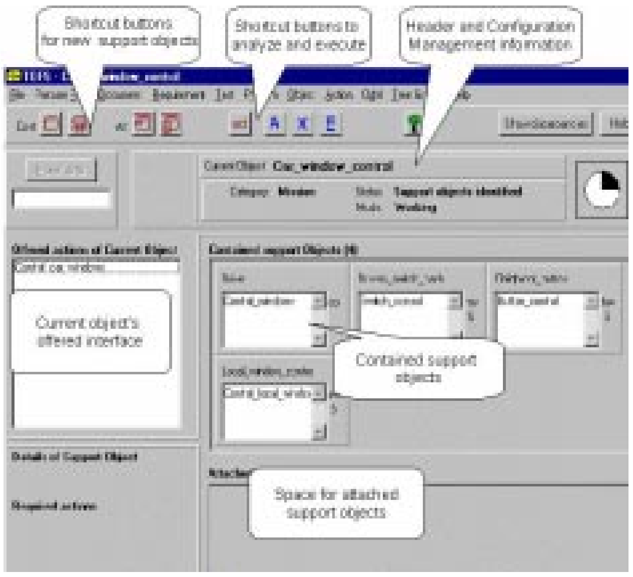
The basic understanding behind Tofs is that systems and software engineering is qualified work, which you should not try to automate. However, as the process to engineer complex systems requires skilled large teams and considerable time, communication is crucial in two ways:

- Communication between people of different categories (stakeholders of the system)
- Communication over time for a single developer to keep track of his or her own work.

The large amount of information in systems engineering raises the need for computer tools to store, keep track of, check, document and communicate the information. An important feature of such a tool is that it supports and assists, but does not try to replace, the innovative manual processes required for successful systems engineering.

Below is shown the main object graph in Tofs. It has the “look and feel” of an ordinary Microsoft Office program. It is also closely integrated with Microsoft Word, which is used for editing and for document production.

The functionality in Tofs concentrates on production of formally correct and consistent system models, which can then be used as specifications to govern detailed work on software and hardware modules and also to analyze operator roles.


Shortcut buttons for new support objects
Shortcut buttons to analyze and execute
Header and Configuration Management information
Current object's offered interface
Contained support objects
Space for attached support objects

FbSim - Force Level Simulation in Ada 95

Richard Elg, Sjöland & Thyselius

1 Abstract

FbSim is a general system for building force level simulations. Today it is used for simulation of aircraft, air defense units and various armored units.

FbSim has been developed since 1989 by several major defense industries in Sweden as well as by the Defence Research Establishment (Foa).

The simulated units are made as agents: they have sources of information (sensors, communication with other agents, a combat plan), they use a set of rules to make decisions on this input and then act using weapons or vehicles or by giving orders.

An operator can take command of one of the agents and interact in real time in the simulation through a user interface.

The system has a client/server architecture with one server application and (currently) five client applications. The server handles the calculations and processing during the simulation and the clients are different user interfaces or interfaces to other applications.

The applications and source-code of FbSim are freely provided for military government projects in Sweden.

All applications in the system run on Windows NT. The server also runs in a Unix environment.

The server has an object-oriented design made with emphasis on expandability with new simulated entities. The programming language is Ada95.

The clients are written in C++ or Java. They interact with the server using ONC RPC.

FbSim handles DIS as well as HLA. DIS and HLA are standards for making distributed simulations.

2 The project

The goal of the project “FbSim” is to develop software for simulation of military units and weapon systems. The project started in November 1989 as a part of the national IT program for industrial development (IT4), which ended in 1994. The result from this project was a general model and an implementation of the model as software written in Ada.

The IT4 project was followed by the project FBSIM/Lv which ended in 1996. During this project as software system for a specific application, air defense, was developed. This software was adapted to the study needs of the Army Air Defense Center.

During a third project, FBSIM/PV, which was commissioned by the Defence Material Administration (FMV), new software modules were developed. It was then possible to simulate a unit with anti-tank weapons and its enemy. The software was adapted to the study needs of the Army Brigade Center. FBSIM/PV ended in the spring of 1997.

Also during 1996 and 1997 the user interface applications for Windows NT were developed. In parallel to this, a major revision of the design was made to take full advantage of the object-oriented features of Ada95.

During Spring of 1998 new anti-air modules will be delivered for the RBS90, RBS77 and RBS923 Missile Systems, the LVKV mobile mounted A/A Gun System, and the FOLV Transportable A/A Gun Firing Unit, in addition to the existing RBS70.

The anti-air units use the LvMÅDS bis protocol.

During late Summer of 1998, 500 copies of a CD-ROM will be distributed free to Swedish officers. It will contain a new easy-to-use version of the system, mainly for practice of tactics.

Participants in the project are Bofors AB, Ericsson Microwave Systems AB, Celsius Aerotech AB, Saab Dynamics AB, Sjöland & Thyselius AB, the Defence Material Administration (FMV) and the Defence Research Establishment (Foa).

Users at the National Defense College (FHS), at the Defence Research Establishment, at the Army Air Defense Center, at the air defense regiment Lv3 and at the Army Brigade Center have participated in the development process.

The project has been financed to approximately 35% by Bofors, EMW, Celsius and Saab and to 65% by the government trough Foa, FMV or the IT4-delegation. The total cost to date is somewhere around 40 million SEK.

3 The applications

3.1 Simulation server

The simulation server is the main application in the system. It contains the code that implements the models. This is where the calculation and processing of data takes place during the simulation.

Today the server consists of 350 files with Ada95 source code, a total of 180 000 lines of code. It also uses several thousand lines of C as well as C++.

The code is designed to be easily expanded with modules containing models of new types of units. This has been proven to work well.
The server has an ONC-RPC interface through which the clients interact.

3.2 Pre&Post

This application is a tool for definition, configuration and analysis of war games. It offers three separate views for different purposes:

1. `-map-view where graphical symbols are edited on a map. This is where the user deploys the participating units, draws planned tracks for aircrÒ˜t, defines firing-sectors etc.
2. A scenario-view holding the objects in the game. Here the user can edit mission-plans, equip the units, change organization and do most of the preparations for the game.
3. The log-view. In this a game can be replayed and analyzed in detail. This view co-operates with the map view where some log information is showed graphically, e.g. the state of the units and who sees who.

Pre&Post uses RPC and the database to communicate with the server. It is implemented in C++ (Visual).

3.3 FbSim MilMap

The application MilMap can be used for various map, terrain and GIS related tasks. It has a Plug-In interface that makes it easy to reuse all the functionality of the application. You can build a new module using the Plug-In interface and it will be dynamically linked to MilMap at runtime.

We have made one such plug-in module called “FbSim MilMap”. This is the tool where a training commander initiates and starts the game. He has control of the simulated time and has a Gods-view of the battle-field. He can interact in the game and give orders as a battalion commander.

FbSim MilMap uses RCP. It is implemented in C++ (Visual). MilMap is also implemented in C++, using the Borland compiler.

3.4 UndE23 MMI

This is an exact replica of the user interface used in the RBS23 intelligence unit where the intelligence commander and radar operator work. When starting this application the user takes control of one of the simulated intelligence units in the server. Starting two of these, one of them can act as intelligence commander and the other as radar operator of the same unit. Of course each unit in the game can be connected to this application.

UndE23 MMI uses RPC and is implemented in C++ (Visual).

3.5 FbHLA

This application is used when the server is to participate in a HLA simulation. HLA is an emerging international standard for how to distribute simulations. FbHLA synchronizes the server with other applications in the HLA-game; it ensures that the server has the same simulated time and information as the other applications and that the simulated units can interact over the network.
FbHLA is implemented in Java. It uses RPC.

3.6 FbDIS

FbDIS is similar to FbHLA but uses DIS instead of HLA. DIS is also an international standard for distributed simulations.

It uses RPC and is implemented in C++ (Visual).

3.7 STeG

This is a tool for generating synthetic terrain. Along with the FbSim system a terrain database of all Sweden is delivered. A part of the database which is used by the simulation to calculate sight and vehicle speeds (heights, terrain types) exists in a 50*50 meter raster. This is not accurate enough for some games, in which case STeG can be used to generate terrain data down to a 1*1 m raster.

4 The models

4.1 Introduction

FbSim has a library of models implemented in source code and data.
There are physical models of armaments, communications equipment, jammers, sensors, signatures, vehicles, weapons performance, weather and light conditions and the terrain.

There are also models of the decision-making in a military unit. The unit uses information from sensors, communication, vehicles and armament and combines this with knowledge of the terrain, its equipment, a mission plan and a set of rules. Based on this it can take actions like moving, sending orders, using its weapons, turning off the radar etc.

The simulated entity is an aggregate of equipment models and a decision model. Because of the decision-making the simulated entity acts like an automatic agent throughout the simulation.

The decision-making or parts of it can be replaced with a man-in-the-loop to let real people interact in the simulation.

The simulated time is event-driven, but can be synchronized with real time to allow human operators.

4.2 Agents

4.2.1 Aircraft

Two hierarchical levels exists, *Leader* and *Pilot*. When the aircraft act in groups of two or four one of them should be *Leader* and the others *Pilot*. The aircraft-agents can have two types of armament: common (*cannon* or *bomb)* or missiles. The missiles are themselves modeled as agents.

Data for decision-making models exist for the following types of aircraft:

- **Attack** Uses missiles of type *Attack* or *Stand-Off Attack.*
- **Anti-air hunter** Uses signal-seeking missiles.
- **Scout** Searches for enemies and sends their positions to other aircraft.
- **Bomber** Carries bombs to be dropped on position shown in mission plan.
- **Jamming aircraft** Carries a background jammer.

4.2.2 Air-To-Ground-Missile

The missiles are made as agents because they can use sensors and communication and choose targets.

The following types of missiles can modeled today:

- **Attack.** The pilot locks it on target before launch.
- **Stand-Off Attack** Navigates trough terrain and opens TV-seeker in target area.
- **Stand-Off Radar** Like *Stand-Off Attack* but uses signal seeker.
- **Target-Of-Opportunity.** Long range like *Stand-Off* but opens seeker directly after launch.

4.2.3 Lulis

A model of the air-situation known by the air-command. It is own aircraft as well as hostile spotted by long range radar. It sends information to anti-air intelligence and firing units.

4.2.4 Anti-Air Tactical commander

This agent is used to model the tactical commander of RBS70, RBS90, RBS23, RBS77 LvKv and FoLv.

4.2.5 Anti-Air Intelligence unit

These agents collect data with their radar, and use communication to get target information from Lulis or co-operating intelligence units. They receive orders from the tactical commander on how to use the radar etc. They send target information to the firing units.

Some of them use the LvMÅDS bis protocol when communicating with other intelligence or firing units.

Agent type	Sensor	Vehicle
UndE70	PS-70	TGB20
UndE90	PS-90	Bandvagn 208
UndE23	PS-2301	TGB20
UndE77	PS-90, PS-707R or PS-2301	TGB20

4.2.6 Anti-Air Firing unit

The Firing Unit receives target data from the intelligence units and other sources. Based on that data it seeks for targets with its own sensors. It models the weapon with its missile operator and loader as well as the C&C-unit with combat commander, target observation etc

Firing Unit	Armament	Sensors	Vehicle
RBS70	RB70	Eyes or TV	TGB20, LvRbBV 701
RBS90	RB90	PS91, IRV, TV, Eyes	Bandvagn 208
RBS77	RB77 Hawk	PE-541/R	TGB20
RBS23	RB23	PE-23	Bandvagn
LVKV	LvAkan, zone	Thomson Local Srr, Eyes	Stridsfordon 90
FOLV	LvAkan	Eyes, Laser	TGB20
RBS923	RB23 or RB90	IRS700, IRV	Stridsfordon 90

4.2.7 Anti-Tank/Tank Battalion Commander

Handles the decision-making made by a battalion commander in an Anti-Tank/Tank battle. Uses radio to get information and to send orders. Does not move or use weapons.

4.2.8 Anti-Tank/Tank Company Commander

Handles the decision-making made by a company commander in an Anti-Tank/Tank battle. Uses radio and own sensors to get information. Uses a vehicle of type BMP1, BMP3 or Strv 122. Does not use weapons. Four types of this agent exist, commanding different types of companies.

4.2.9 Anti-Tank/Tank Platoon Commander

Models a platoon commander. Uses radio or speech. Has own sensors: Eyes, Ears and IR. Moves on foot or in subordinate vehicle. Does not use weapons. Nine types of platoon commanders are modeled.

4.2.10 Anti-Tank/Tank Fighting vehicle

Models the fighting vehicles, with armament, transportation, sensors. It participates in the battle with its weapons. It can host a platoon commander and a group of soldiers. Ten types of this agent are modeled.

4.2.11 Anti-Tank/Tank Group of soldiers

The whole group is modeled as one agent. There are eight types.

5 Overview of design of simulation server

There is a **Simulation_Case**-object. It reads the scenario definition file and creates all agent objects that should participate in the simulation. The objects are stored in the *Object_Manager* object.

Object_Manager holds information of which *Signature_Object* exists and has operations so that objects in the simulation can check this and which objects are in a certain geometric volume.

Signature_Object defines objects that can be seen or heard by the agents. In other words they have a position, an orientation and a signature.

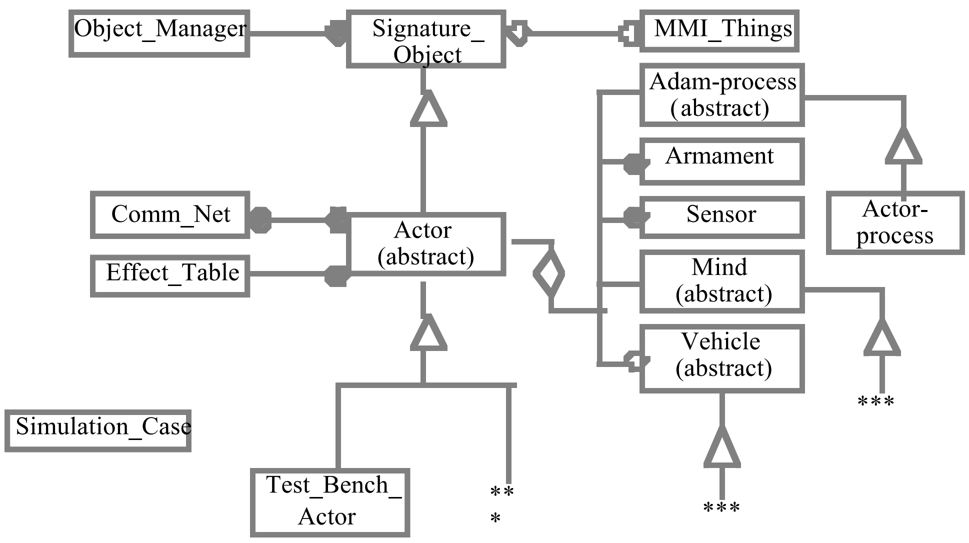


Figure 5. The class diagram of agent-objects (actor).

Actor is an abstract type. It has the properties that are common for all agent types: i.e. the objects in the simulation that acts automatically: they can for instance see, move, take decisions and fire. It has the Adam-process that is activated when the agent is about to make a decision or perform an action.

Test_Actor is an example of a type that end-defines an agent. This is where the abstract operations defined in *Actor* are implemented. It is for instance *Set_Mind* that creates the *Mind*-object of the right type for the agent.

MMI_Things is used to associate MMI-specific data to objects in the simulation. In that way there is no need to mix things that only belong with the user interface with the data types for the simulation. This makes it easier to adapt the program to different types of MMI.

Comm_Net handles communication nets. Agents in a common net can send messages to each other. A net contains many agents and an agent can be in many nets.

Effect_Table is a table where weapons performance between a certain type of target and type of weapon is described. An agent has one such table and each table can be used by several agents.

Adam-Process handles that the agent is scheduled on the time-axis and is activated on events or points in time.

Armament handles weapons, i.e. loading,

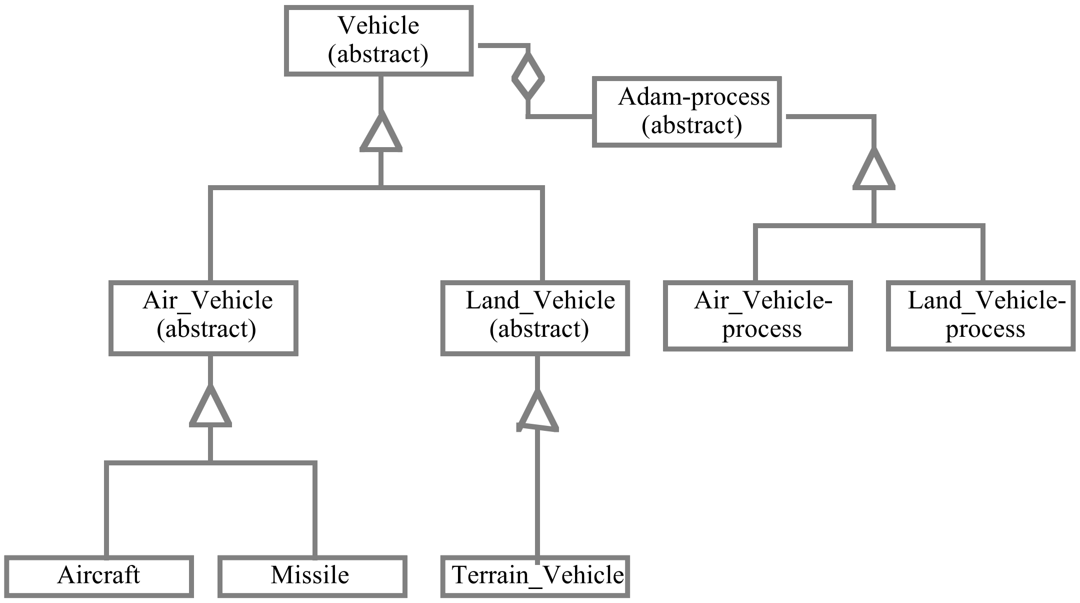


Figure 6. The class diagram of the vehicle.

target tracking, hit calculations, trajectories etc.

Sensor handles sensors.

In **Vehicle** those things that are common for all types of vehicles are handled. This and the next level (*Air_Vehicle* and *Land_Vehicle*) is what the agents (trough their *Mind*-object) use to control and sense their vehicle. *Simulation_Case* uses only this level, which makes it possible to add a new type parallel to *Air_Vehicle* and *Land_Vehicle* without having to make a change in *Simulation_Case*.

Air_Vehicle and **Land_Vehicle** define and implement common properties for vehicles using air-tracks and land-tracks respectively. These types are, just like *Vehicle*, abstract and yet another level is needed to finally define a vehicle.

Aircraft and **Missile** are two types of airborne vehicles with somewhat different aerodynamical properties. **Terrain_Vehicle** is a vehicle that can move on the ground in the terrain. New types can be added in parallel to those without any recompilation of existing files.

The abstract type **Mind** is used for the main part of the decision making of the agent. It is for instance the structure with Information-interpreter, Information-base, Situation-interpreter, Decision-base and Decision-machine. It has a non-abstract operation *Make_And_Execute_Decision* that is called from the Adam-process in *Actor* each time anything has happened that can lead to a decision or action. *Mind* has a number of abstract operations that it calls from *Make_And_Execute_Decision*. These operations decide how the agent understands its environment and situation, what conditions it has to make a decision and how it implements the decision.

Condition and **Consequence** are used for *Transform_Condition* and *Transform_Consequence*. In the rule-file are a set of rules with logical expressions of conditions and consequences. In **Rule_Collection** an image of the rule-file is built. To each condition in the file a *Condition*-object is created. The *Condition*-object has in principle only one operation: *Transform_Condition* (*condition*, *condition_info*) that returns True or False depending on the value of *condition_info* and the attributes of conditions. Consequence works in the same way.

Test_Actor.Minds is an example of a type where a *Mind*-object is finally defined and the abstract operations of *Mind* gets their implementation. Normally there will be one such type for each type of agent.

If there exist types on the *Test_Actor.Mind*-level that have mainly similar implementation, one can break out those common parts and create an abstract type between the *Test_Actor.Mind*-level and *Mind*. This is exemplified with the type **Common_1**.

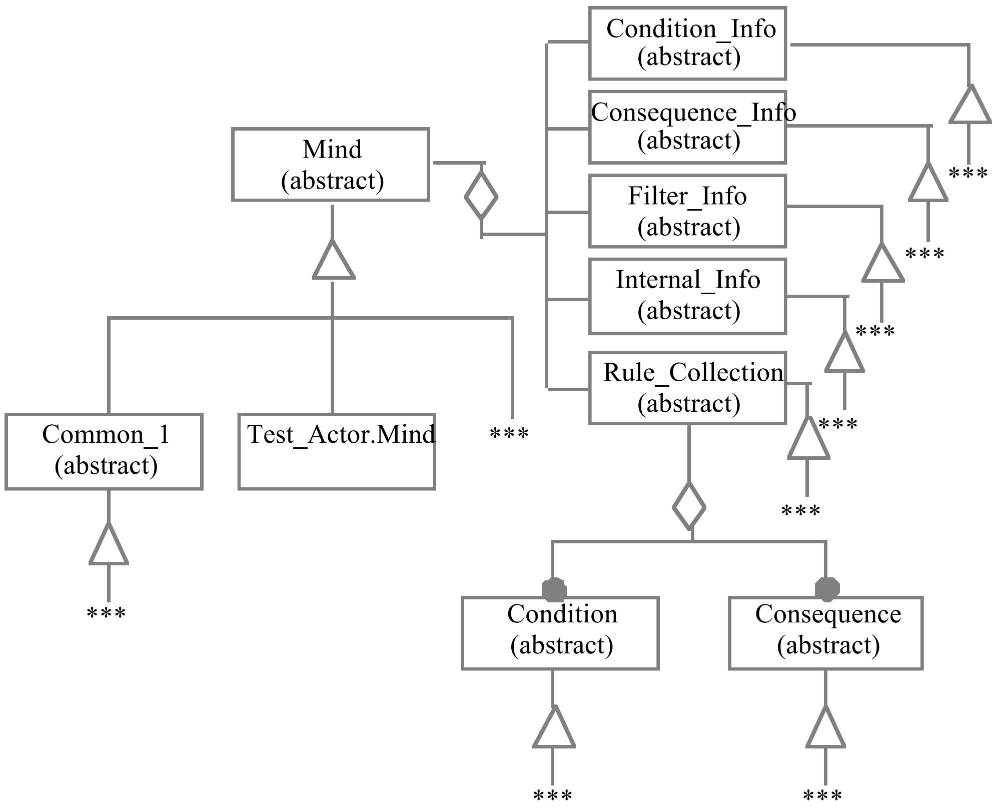


Figure 7. Class diagram of Mind.

6 Implementing in Ada95

As mentioned above the simulation server consists of 350 files with Ada95 source code, a total of 180 000 lines of code. Below is a simplified example of how we use the language to implement the reusable design described in chapter 5 Overview of design of simulation server.

The packages Mind and Actor contains the code reused when new types of actors are implemented like in Pv_Pluton_Chef_Actor. The package Base_Mind is a implementation trick to avoid circular dependency between Actor and Mind. The Base_Mind.Mind_Pointers are converted to Mind.Mind_Pointers in the body of Actor before it is used.

When the simulation is initiated, a set of files are read. One of them defines the actors. The operation Actor.Init_From_File is used in this process. Among other things it calls Actor.Create_Mind. It also creates and starts the Adam-process. At the call to Create_Mind Ada95 sees to that it will be the right body of the operation that executes. If the Actor_Pointer points at a Pv_Pluton_Chef_Actor.Actors-object it will be Pv_Pluton_Chef_Actor.Create_Mind. Thus a Pv_Pluton_Chef_Actor.Mind-object is created. The Adam-process will later call Mind.Make_And_Excecute_Decitions and inside it the right operations implemented Pv_Pluton_Chef_Actor will be executed.

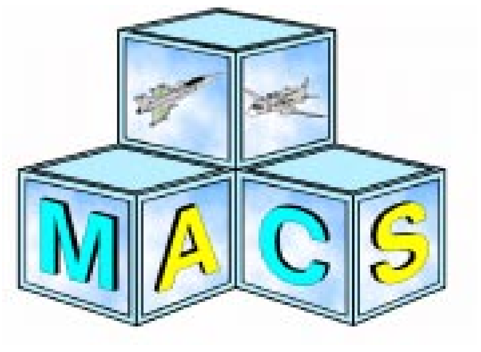
```
package Base_Mind is
  type Minds is abstract tagged null record;
  type Mind_Pointers is access all Minds'Class;
end Base_Mind;


with Actor;
package Mind is
  type Minds is abstract new Base_Mind.Minds with
private;
  type Mind_Pointers is access all Minds'Class;
  procedure Make_And_Execute_Decisions
    (The_Mind           : in out Minds'Class;
     The_Actor          : in     Actor.Actor_Pointers;
     Planned_Event      : in     Boolean;
     New_Planned_Event :     out Plan_Record);
  …
end Mind;

with Base_Mind;
package Actor is
   type Actors is abstract
      new Signature_Object.Objects with private;
   type Actor_Pointers is access all Actors'Class;
   function Create_Mind (The_Actor    : in Actors;
                  RuleFileName : in String)
         return Base_Mind.Mind_Pointers;
   procedure Init_From_File
     (Actor_Ptr : in     Actor_Pointers;
      The_File  : in out Ada.Text_IO.File_Type);
…
end Actor;

with Mind;
package body Actor is
  …
end Actor;

package Pv_Pluton_Chef_Actor is
  type Actors is new Actor.Actors with private;
  type Minds is new Mind.Minds with private;
  function Create_Mind (The_Actor    : in Actors;
                 RuleFileName : in String)
   return Base_Mind.Mind_Pointers;
…
end Pv_Pluton_Chef_Actor;
```



THE NEW COMPUTER SYSTEM IN JAS39 GRIPEN IS MACS

The standardized computersystem SDS80 is used in the Swedish JAS39 Gripen aircraft for the four main subsystems: the radar, the systems computer, the electronic presentation system and the electronic counter measures system. The system consists of the Pascal/D80 language, the PUS80 program development environment and the D80 modularized true multiprocessor computer. The Industry Group for JAS, IG JAS, has realised that future applications and functions will need better calculation performance and more memory.

The new standardised computing system is called MACS, Modular Airborne Computer System. It consists of a computer, MACS and a software engineering environment called SEEMACS (which replaces the present PUS80). As MACS is built up on general modules, it is possible to adapt a computer according to need.

The main purpose of MACS is to give the Swedish Air Force a new standardised computer system with much better performance than the present SDS80, the design of which was frozen in 1978 originally, and its processors upgraded to the D80E version in 1990. This will make it possible to introduce new features in the Gripen aircraft. By building MACS on standards and commercial components, the SDS MACS will be easy to enhance.

MACS, just like the SDS80, is a true multiprocessor computer with built-in support for real-time applications.

All source code for the D80E processor is written in the Pascal/D80 language. This is an Ada-like version of Pascal with real-time extensions and failure control. The language also supports multiprocessing. One requirement on MACS is that it has to be possible to use this investment in software with only minor changes. For new applications there shall be support for Ada (read Ada 95). The present solution in MACS supports Pascal/D80 and Ada to be used in separate processors. Communication between these processors/languages is made by using a common memory area.

The function units in MACS are
- Processor (built on PowerPC 740)
- Mass Memory (built on flash memory, PC-card)
- 1553B for communication

MACS is built to the maximum extent possible on commercial standards. For example the processor is built on VME, PowerPC, PCI and Ethernet. The hardware has built-in support for failure detection and ECC (Error Code Correction) for all memories.

MACS software development environment SEEMACS is built on Rational’s development environment VADS Cross. It will be ported to Rational Apex during 1998. The development environment includes a lot of development tools besides the compiler system, such as
- Trace
- Metrics
- Systems variable handler
- Real-time debugger
- Simulator
- Load file generator

The debugger has been developed in close co-operation with Rational. As an example of the performance of the debugger, reading a variable only disturbs real time typically 50ms. By using Rational’s development environment, it will be easy to add the different tools that Rational offers, by simply ”plug and play”.

Right now a Functional Model (FUM) is being developed. It has almost the same functional level as the final unit will. It is built in VME-format which makes it easy to fit into an ordinary office environmental, without the requirements on forced air. The major purpose of this FUM is to make possible an early marriage between software and hardware in the MACS system. The FUM will later be used by Saab testing the new functions in the enhanced systems computer.

JavaOne’98 Highlights

Torbjorn.Andreasson@emw.ericsson.se
Lennart.Bie@emw.ericsson.se

Ericsson Microwave Systems AB
1998-05-12

The JavaOne conference 1998 had more visitors than any of it’s predecessors. More than 14.000 “Java Maniacs”, which was 4000 more than the 1997 number and 7000 more than on the 1996 conference, visited Moscone Centre situated in San Francisco, California, USA during the week 24 - 27 March 1998. The Master of Ceremonies, John Gage, claimed that the 1998 JavaOne conference has become the largest software development conference ever and it indeed gave an overwhelming and mind shaking impression.

Introduction

More than 100 talks and 81 BOF’s, Birds Of a Feather (or Birds Of Feature) were presented during the conference. It also hosted two “Hacker’s Lab” with lots and lots of computer available to the visitors, a book shop and an exhibition hall with more than 250 exhibitors one of which was of course Sun. They had, as one could expect, a dominating part of both the exhibition and of the whole conference.

The conference vividly showed how rapidly the Java technology evolves and expands, both in number of companies endorsing this technology and in the number of developers and other persons who devote their heart to Java. But naturally, the expansion is as intensive in the growth of Java functionality including areas like new API’s (Application Program Interfaces) as it is in the area of compiler technology. The latter was presented to the flabbergasted audience in a live demonstration with the new HotSpot compiler competing with the best of the current Java compiler technologies. It showed that the coming compilers could well match traditional ones like C++ as well as any other compiler.

Lots of other “hot” Java technologies was also demonstrated including Embedded Systems, JavaPC, Enterprise JavaBeans as well as lots of new API’ like the 3D- API, the 2D-API, the JavaCard-API, the Security API, the JavaHelp-API and many others. Java more and more becomes the complete platform suitable for all type of development covering the whole range of applications, from the smallest embedded, like the JavaRing (see further down) to the big distributed enterprise and n-tier applications. Java has now gained all the functionality, performance and maturity that is essential for these type of applications.

The Java Ring

At the first JavaOne Conference in 1996, Sun predicted that at the next conference everyone would be given a Java Device. At the 1997 conference they where close to giving away a smartcard running a portion of the Java Virtual Machine. At the 1998 Conference every attendee was given a Java Ring - a ring that actually contains a Java Virtual Machine. The ring stored each attendee’s business card information and also calculated a point in a large fractal. The Java Ring is a secure Java-powered token with a continuously running, unalterable real-time clock and rugged packing. The jewel of the Java Ring is the iButton (single-chip) by Dallas Semiconductor: a million transistor microcomputer with a Java Virtual Machine housed in a rugged and secure stainless steel case.

Distributed computing

The goal with distributed computing (for over 20 years) has been to allow the sharing of distributed information and the computing environment. Traditional approaches have confronted environments that were extremely hostile to the programmer. Multiple machine architectures with compiled languages, multiple data formats, and heterogeneous programming languages and type systems. The Java technology has changed this set of rules by provide a homogeneous layer over heterogeneous machines and operating systems. The environment allows code to be moved from one machine to another and be dynamically loaded into existing programs. The built-in security allows the code to be run with much greater confidence that the recipient's computing environment will not be compromised. The result is a powerful distributed object-oriented platform and a development environment based on RMI and CORBA.

CORBA

Java IDL technology provides standards-based interoperability and connectivity for distributed applications using the CORBA/IIOP standard. Features in Java IDL technology include transient and persistent objects, Portable Object Adapter (POA), server activation, COS Name Service, Java Transaction Service (JTS) technology support, and IIOP/SSL. Part of this functionality will be in JDK-1.2, while the rest will be in an extension.

RMI, Remote Method invocation

RMI is the primary technology used to distribute Java programs. It has a simple programming model which supports the Java object model and other features like the movement of Java classes and libraries within the network, object grouping, object activation and persistent references. RMI supports two distribution techniques, RMI over JRMP (Java Remote Method Protocol) and RMI over IIOP (Internet InterORB Protocol, the CORBA wire protocol). RMI/IIOP allows interoperability with CORBA-compliant distributed applications with only some minor limitations compared to the full RMI/ JRMP model. This is about to change with the introduction of the CORBA 3 standard which aims at removing all such limitations. The RMI/IIOP product will ship with JDK 1.1.6 and 1.2 in summer 1998.

JavaBeans and the Glasgow project

JavaBeans, which is the component model for the Java platform, was introduced already last year with JDK 1.1. It has since been enhanced, in particular with the delivery of the JFC or SWING libraries and with the introduction of the InfoBus and JavaBeans Containment, introduced in the Glasgow project, which will provide a logical containment hierarchy of JavaBeans.

New security framework

One of the few core changes to JDK-1.2 is the modified security model. In JDK 1.0, the "sandbox" security architecture was inflexible and difficult to customize. JDK-1.1 added support for trust based digital signatures (signed programs) and took the first step toward enhancing the usability of the security framework. JDK- 1.2 now provides additional security capabilities by giving webmasters and network administrators more fine- grained control over that afforded to programs from a trusted source. By default, trusted programs now run in the "extended sandbox" and let the original sandbox to include a portion of a specific area on the local disk (/ sandbox). The extended sandbox comes automatically for digitally signed programs, and requires no special configuration on the part of network administrators.

Requirements capture and traceability in Systems Engineering

Reported from the EuroForum conference in London April 29th and 30th 1998

Ingmar Ogren, Romet AB

The conference gathered a large number of participants from the UK and the presenters represented a selection of leading experts from the UK and the US. The impression was that the presentations represented state-of-the-art for requirements' management for complex and safety-critical systems (not only software).

The conference was co-located with an exhibition with the tools RDD 100 from Ascent Logic, DOORS from QSS and CRADLE from 3SL. In the presentations the tools RTM and Tofs were also mentioned.

The presentations gave a feeling of agreement upon what is meant by a mature requirements' process. No one claims any longer that you can write a requirements' specification and have the job done to go into design. Everyone seems to agree that the requirements must be managed in a continuos process from the start of a project, through design and into maintenance, especially for large and critical projects.

The requirements' process encompasses capture, analysis, elaboration, tracing and verification of requirements on the complete system, including its operators. Especially the operator parts of a system were considered critical as real failures often depend on unexpected operator behavior. Elaboration of requirements includes weighting and derivation of requirements and this must be done in cooperation between contractor and end-user.

Several presenters talked about the need for better, cheaper and faster systems engineering and the recommended measures was to use more of the resources early in a project and to deepen the cooperation between customer and contractor. This leads towards new (iterative and interactive) principles for acquisition and towards establishment of a process for requirements' management.

If specialized tools should support the process depends on project size. Smaller projects can be managed without tools or with a ”home made” data base, while larger projects require ”real” tools with a built-in database. It was discussed if the requirements should be seen as objects (DOORS) or if requirements and test cases should be seen as attributes to design objects (Tofs and others). The latter alternative requires that requirements' management be done concurrently with design.

The understanding that requirements should be managed together with design led to discussions about how to model designs. A common opinion is to use the UML, but UML and other notations, with graphical behavior descriptions, were reported to give difficulties in reviewing. From this point of view textual behavior descriptions can be preferable, as they are easier to analyze automatically. Formalism is good, but it is crucial that the formal descriptions can be understood by, or at least be explained to, end-users. In this context several presenters mentioned Fault Tree Analysis and Failure Mode and Effects Analysis as recommendable techniques to analyze system dependability. An important aspect is to know what you are talking about. Problems with linguistic and cultural differences were reported. Glossaries, Data dictionaries and Type dictionaries were recommended.

It was discussed what should be included in a set of requirements. To get to the required faster, better and cheaper ways of work, it was considered important that you evaluate each requirement. This means that you consider each customer wish against available technology, budget and time to get a set of requirements, which is both realistic and expresses the customer's needs. This leads towards iterative ways of work, with successive releases, something which must however be applied with care, to avoid that you field a multitude of immature system versions. A practically proven technique is to apply SWOT analysis (Strengths, Weaknesses, Opportunities, Threats).

Laws and regulations form part of the requirements for many systems. These can be ”old fashioned” laws, which can be managed as check lists or newer risk based laws, which require that a contractor analyses risks and shows how they can be prevented through the system’s behavior.

An interesting special case is to work with the technology required for a business. You can show how products depend on technologies and how different technologies depend upon each other. It was shown both how you can use requirements’ management tools for this work and how it can be of crucial importance for a company’s ability to compete.

The conclusion is that requirements’ management must be a process through a project and that requirements’ management is crucial, if you want to get projects done faster, better and cheaper.

Science and Technology Programs for Defense Adaptation

The Swedish Parliament has allocated separate funds for a program to assisst industry in preparing for supporting the new Defense Doctrine of an Adaptable Defense, which should be able to change more easily, as the tasks of the Defense Forces change. SESAM has assisted FMV in generating proposals for projects in Software Engineering within this program. These proposals are now under consideration by the Defence Forces and FMV. The list of proposals is as follows:

Open Systems in Real-Time Applications

- Open (Military) Adaptive Interface Standards
- Tools and Component Development Using ASIS
- COTS och Object Orientation as a Basis for Design of Real-Time Applications
- Software Patterns and Component Reuse

Conflicts of Interest in Using COTS Products

- Adaptation of the Acquisition Process for COTS Products when Developing C3I-systems
- Ability of COTS Products to Meet Product Requirements in C3I-systems

Safety Critical Software

- Experimental Verification of Fault Tolerance
- Run-Time System Kernel for Safety Critical Complex Systems
- Formalisation, Analysis and Management of Requirements on Safety Critical Systems
- Transfer to Industry of Software Engineering Technology for Safety Critical Systems

Experiences from Implementing an RTI in Java

Mikael Karlsson, Staffan Löf, Björn Löfstrand
Pitch Kunskapsutveckling AB
Nygatan 35
Linköping, S-582 19, Sweden
+46 13 13 45 45

Keywords: HLA, RTI, Java, performance, platform independence, integration, interoperability

ABSTRACT: *During the last couple of years Java has emerged as an interesting development approach for many software developers. Since its introduction Java has been associated with a wide and extensive debate, ranging from business issues to language properties such as suitability for different kinds of applications.*

This paper focuses on two major aspects of Java – performance and integration with other languages. We will address these issues based on our experiences from implementing a Runtime Infrastructure (RTI) in Java.

The experiences presented in this paper will focus on: (1) performance including throughput and latency; (2) integration with federates implemented in languages such as C++ and Ada. The paper will also characterize the implemented RTI in terms of functionality, architecture, topology, and principal solutions to some typical problems. We further discuss implementation considerations made in connection with Java, including platform independence, robustness, and rapid application development.

This project has been carried out as an R&D project on behalf of the Swedish Defense Material Administration.

1. Introduction

The High Level Architecture (HLA) establishes a common high-level simulation architecture that facilitates interoperability of all types of models and simulations. The HLA also promotes reuse of M&S components.

The HLA Runtime Infrastructure (RTI) is the software component that makes interoperability in federations of simulations technically possible. At runtime, the RTI provides commonly required services to federates. These services are specified in the HLA Interface Specification [1].

Implementing and maintaining RTIs on multiple platforms is an expensive and time-consuming process. During the last few years Java [2] has emerged as a solution of interest for cross-platform development. Java is an architecture-neutral object-oriented language with syntax similar to C++. It is a robust language with features aimed at reducing the number of programming errors. These properties motivated us to choose Java for our implementation of an RTI.

Two major concerns regarding our Java RTI were performance and integration with federates written in other languages. High performance RTIs are crucial for some federations, and federates has been and will be developed in many different languages.

This paper focuses on: (1) performance including throughput and latency; (2) integration with federates implemented in other languages. The paper also characterizes our RTI implementation in terms of functionality, architecture, and topology. We also discuss our experiences with Java in terms of platform independence, robustness, and rapid application development.

2. RTI Lite

RTI Lite is our Java implementation of a selected subset of the services specified in the DoD HLA Interface Specification v1.2. No Data Distribution services are included, and RTI Lite implements reliable transport only.

The RTI Lite project was initiated to gain first-hand experience with HLA. RTI Lite was developed on Windows NT using Microsoft Visual J++ and Symantec Visual Café 2.0. It runs on a Java 1.1 Virtual Machine. RTI Lite is currently being tested in an aggregation/ deaggregation proto-federation.

2.1 Architecture

RTI Lite uses two different kinds of reliable transportation: Java Remote Method Invocation (RMI) and a proprietary transportation mode. RMI is a form of object oriented RPC, that uses Java Object Serialization to marshal and unmarshal parameters. RMI assumes the homogeneous environment of the Java Virtual Machine (VM).

To increase performance, we have also implemented a custom transport mode for *updateAttributeValues* and *reflectAttributeValues*. This transport mode is based on TCP streams and a specially designed packet format. No datatype information is contained in the packet. Instead, RTI Lite uses the loaded FOM to look up datatypes for attributes and parameters. Federates are required to handle the assignment of datatypes themselves. Since no type checking is done by the RTI, invalid packets may cause problems.

Packet Format

Packet size	16b	Size of packet
Service code	16b	Identifies service
Parameters	8b*n	Service parameters

Attribute / ParameterHandleValuePair Format

Id	16b	Handle
Data	8b * n	Value

Array Format

Element count	16b	# of elements in array
Elements	8b * n	Element data

ReflectAttributes(12, [[1, 1], [2, “Hello World”]])

Service code	0F	ReflectAttributes
Parameter data	00000012	object id
Element count	02	2
Attribute id	01	1
Attribute data	00000001	1
Attribute id	02	2
Attribute data	“Hello World”	string

Figure 1. RTI Lite Proprietary Packet Format

RTI Lite uses a multi-threaded design, which means that federates do not have to ‘tick’ the RTI. Federates are required to handle asynchronous calls to federate-implemented services.

2.2. Topology

A centralized topology is used where all data distribution is performed via the central RTI node using one-to-one communication to each of the federates. No caching of information is performed in the proxies, and they provide no further functionality except tunneling of service calls.

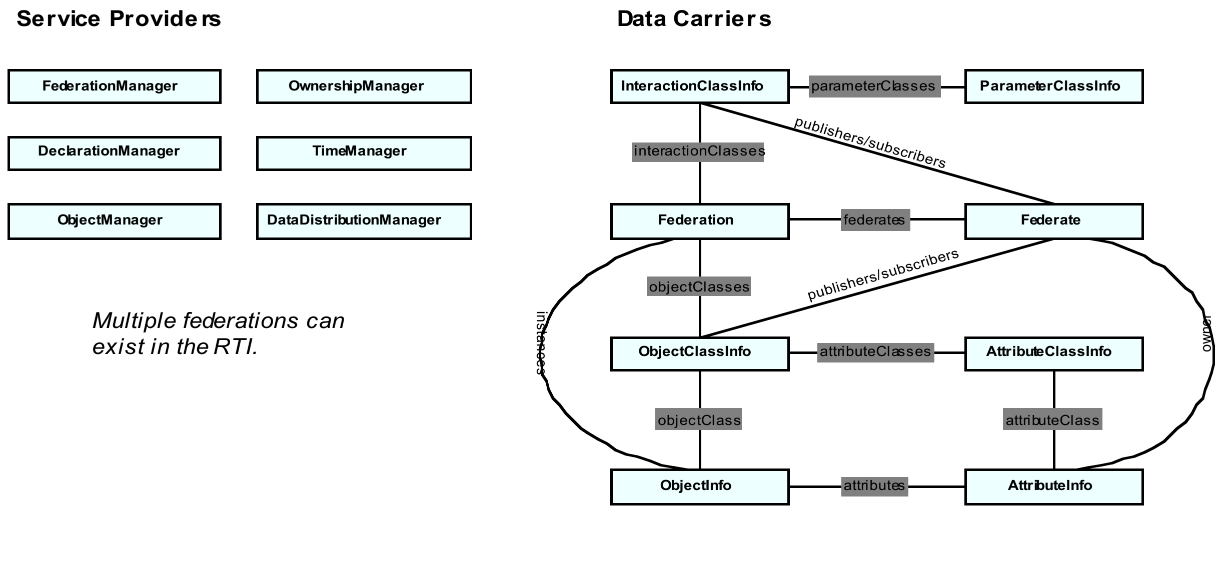


Figure 2. RTI Lite Module Design

3. Performance

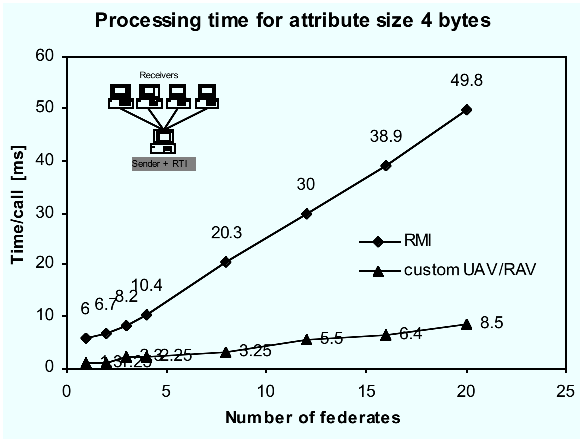
Using Java to implement an RTI raises questions regarding its ability to give satisfactory performance. Our goal was that RTI Lite should be able to handle a throughput of 100 updates per second of 2,000 bytes each with a latency of less than 100 ms.

3.1. Test Setup

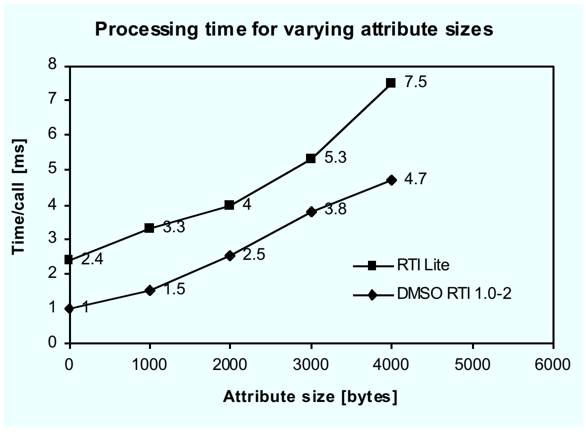
The following test setup was used: 100 Mbps Ethernet, 5 x Compaq Professional Workstation 5000, 64MB RAM, 200 MHz Pentium Pro, JDK 1.1.3, Symantec JustInTime Compiler version 210.063. The RID was generated with Visual OMT [3].

3.2. Results

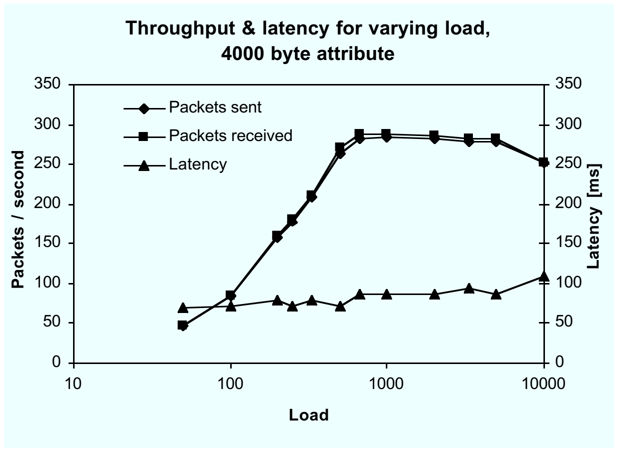
Processing time was measured by having a sender federate perform 1,000 *updateAttributeValues* to *m* receiver federates. Each receiver federate sent an interaction after receiving 1,000 *reflectAttributeValues*. Finally, after receiving *m* interactions, the sender federate reported elapsed time.



Processing time was also measured with attribute size varying between 1 and 4,000 bytes. The DMSO RTI 1.0-2 was used for comparison.



Latency measurements were made by having one sender federate perform 10,000 *updateAttributeValues* to one receiver federate. After receiving 10,000 *reflectAttributeValues*, the receiver sent an interaction back to the sender. The sender then reported latency as the time between when the last *updateAttributeValues* was sent and when the interaction arrived. The reported time was divided by 2 since it was measured in both directions. Measuring the last update in a large set ensured that the RTI was not buffering large amounts of data.



The amount of data sent and received from the machine running the RTI was analyzed with a network monitor tool. This test was also done using the DMSO RTI 1.0-2. The results of this analysis are presented in the table below.

	DMSO RTI	RTI Lite RMI	RTI Lite Custom
Packet information - 1000 reflects, 4 byte attribute			
Bytes/reflect	64	302	18
Reflects/packet	22	1	25
Packets	45	1000	41
Ratio packets/ack	3	1	1
1000 reflects - 1 recipient, 4 byte attribute			
Packets sent	116	1035	41
Bytes sent	70660	359204	23018
Packets received	48	1044	43
Bytes received	2672	79170	2635
1000 reflects - 2 recipients, 4 byte attribute			
Packets sent	232	2079	87
Bytes sent	141539	750058	46464
Packets received	97	2120	93
Bytes received	5451	160649	5640

3.3. Discussion

Performance measurements of the RTI are highly dependent upon the characteristics of the federation under which they are collected. Furthermore, Java performance depends very much on which virtual machine and which compiler is used.

The results from our tests indicate that using RMI as reliable transportation delivers a performance that is far from adequate. RMI generates approximately 16 times the amount of data required per reflect compared to our custom packet format. RMI also spent much time marshalling and unmarshalling.

Using the custom packet format made the performance improve dramatically. We were able to handle sustained updates of 2,000 bytes each at rates as high as 200 per second without negative effects on RTI latency. RTI Lite was able to process approximately 275 updates per second.

Using a compiler that generated native code did not affect performance significantly compared to the Just-In-Time compiler that we used. In the future, native code compilers will almost certainly give better results, at the cost of portability.

4. Integration

Any RTI created has to be able to interface with many different languages and operating environments.

RTI Lite has been used with federates written in two different languages – Java and Ada. The Java federates imported and used the RTI Lite proxy classes directly. Federates written in Ada used RPC to communicate with a small Java application that encapsulated the RTI Lite proxy. This approach could be used with federates written in any language that supports RPC.

Another way to integrate Java programs with C and C++ is through the Java Native Interface (JNI). We have performed some initial tests, but have not fully explored this possibility. One possible complication is that JNI is primarily intended for Java to access native code, and not the other way around.

One approach that we examined was to use a Java native compiler to create a Dynamic Link Library (DLL). Unfortunately, these DLLs used a proprietary export format that could only be used by Java applications that had been compiled with the same compiler. However, this solution would be satisfactory if a standardized export format was available.

The ideal interface should be easy to use for the federate developer, give maximum performance, and require minimal implementation effort on our part. Ease of use for federate developers means that the interface has to integrate tightly with the target language, e.g. a C++ programmer wants C++ classes. This, however, means that the RTI developer must support interfaces in several different languages. One possible solution is to use a language-independent Object Request Broker (ORB), such as CORBA, at the cost of performance.

5. Java

What is usually referred to as Java covers three aspects:

- A programming language;
- a virtual machine that executes bytecode; and
- a compiler that translates the programming language to bytecode.

5.1. Language

Java was designed to be simple and intuitive. The number of language constructs has been minimized to make the language easy to learn. Java looks like C++, without the unnecessary complexities of this language, making it familiar for current C++ programmers.

Because there are no pointers, Java is a robust language. The most severe runtime errors occur when pointers are misinterpreted, causing data corruption and memory overwrites. This type of problem does not occur in Java. In addition, it is not possible to turn an arbitrary integer into a pointer by casting. All dynamic type conversions are checked for correctness. Java also has a garbage collection feature that automatically frees unused memory. By eliminating pointers and simplifying memory management, Java reduces the amount of programming errors.

Java is still a language under development, and there are very few environments for application development that support the latest version of Java. During the development of RTI Lite, we have gone through five versions of Java and four different development environments.

5.2. Bytecode

The bytecode is an architecture-neutral object file format that can be executed on many processors, given the presence of the Java runtime system. The bytecodes can be translated on the fly (at runtime) into machine code for the particular CPU the application is running on, a process commonly referred to as Just-in-time compiling. This is somewhat like putting the final machine code generator in the dynamic loader.

Bytecode and the virtual machine are not limited to the Java programming language. For example, Aonix offers an Ada95 development environment that can generate Java bytecode.

6. Conclusions

We are satisfied with the performance of RTI Lite. The goal was that RTI Lite should be able to handle a throughput of a 2,000 bytes update at a rate of 100 updates per second with a latency of less than 100 ms. Java was able to provide the performance necessary to achieve this goal.

Theoretically, RTI Lite will run on any Java v1.1 supporting platform, however not all Virtual Machines are created equal and some, like Microsoft's Java VM, only partially supports Java v1.1. When writing an RTI in Java careful considerations must be made regarding on which Virtual Machines it will run and only use features in the language that all VMs support.

One very complicated issue regarding HLA is interoperability between different RTIs. Standardized data and packet formats can help, but the architecture and internal communication of a distributed RTI is difficult to standardize.

The Java language proved to be very rewarding to work with. No pointer- or memory-related errors occurred, except for a few null pointers. Overall, we encountered very few errors. The time spent looking for and fixing errors was short making development quick and trouble-free.

One thing that we missed from C++ was templatized collections. Another problem was the requirement that the directory structure must match the package hierarchy. We spent much time trying to get the CLASSPATH settings to work, especially when deploying federates.

There are several different ways to integrate an RTI written in Java with federates written in other languages. Most of the integration issues are not specific to Java, but common to RTIs implemented in any language. Support for federates developed in different languages is a complex issue that requires a lot of work on the part of the RTI developer.

7. Future Work

We plan to continue developing RTI Lite to make it compliant with emerging version of the HLA Interface Specification. Support for Data Distribution Management and best-effort transportation will be included. We will investigate different transportation modes to improve performance.

We will also look at different topologies, e.g. a decentralized RTI implementation and the use of smart proxies.

Acknowledgements

The work described in this paper has been carried out as an R&D project on behalf of the Swedish Defense Material Administration.

References

[1] Defense Modeling and Simulation Office (DMSO): ”Department of Defense High Level Architecture Interface Specification Version 1.2”, Online document at URL: http://hla.dmso.mil/, August 13 1997.

[2] Sun Microsystems, Inc.: ”The Java™ Language: An Overview”, Online document available at URL: http://www.javasoft.com, October 21 1997.

[3] Pitch Kunskapsutveckling AB: ”Visual OMT”, Online document available at URL: http://www.pitch.se, December 19 1997.

[4] Furuichi et al.: ”Performance Evaluation model of HLA-RTI and evaluation result of eRTI”, In Proc. Of the 97 fall SIW, pp. 1099 – 1110, 1997.

Author Biographies

MIKAEL KARLSSON is a software developer at Pitch Kunskapsutveckling AB, Linköping, Sweden. He is the developer in charge of the RTI Lite project. Current work also includes distributed simulations for army logistics training.

STAFFAN LÖF is President and CEO of Pitch Kunskapsutveckling AB. He holds an MSc in Computer Science from Uppsala University. He has extensive experience from running high-tech companies and managing R&D programs and projects. During 1995-96 he was Attaché of Science and Technology at the Swedish Office of Science and Technology in San Francisco.

BJÖRN LÖFSTRAND is a software developer at Pitch Kunskapsutveckling AB. He holds an MSc in Computer Science from Linköping Institute of Technology. Current work includes implementing tools for supporting HLA object model development.

This article was first presented at the 1998 Spring Simulation Interoperability Workshop, Orlando, FLA, March 9-13, 1998.

Salvation from System Complexity

Harold W. Lawson, Lawson Konsult AB

When examining the complex structural foundations of the computer and communication systems upon which our society depends, I shudder to think of the disasters—economic, personal, material, and ecological—that lie ahead. The presence of complexity inevitably leads to risks. Risks lead to exposure and hazardous situations, which inevitably lead to failures and accidents as well as malicious and criminal acts.

The ever increasing mountain of complex software will continue to escalate risk, at least until several significant catastrophes (or a single vital catastrophe) occur. At that point, one scenario might find the US government investigating sources of complexity and proposing corrective action.

The US Congress already received a painful introduction to computer-based system complexity via investigations into the failure of the US Federal Aviation Administration's air-traffic control project which cost US taxpayers more than $5 billion. Yet this is probably a small sum compared to the costs resulting from actual, major accidents.

Is there any salvation from the ever increasing risk related to complex computer and communication products? Certainly, there is no silver bullet, but suppliers and professionals involved must actively deal with increasing complexity and risk, hopefully via free-market forces. Ironically enough, catastrophes always lead to new business opportunities—a scenario preferable to dealing with constraining and complex attempts at regulation. Those suppliers and professionals who understand this situation and prepare for it should have a definite advantage.

NECESSARY AND UNNECESSARY COMPLEXITY

In computer and communication structures, there is, and always will be, necessary complexity associated with the useful functions these structures provide. However, current mainstream computer and (often) communication structures contain significant unnecessary complexity. This complexity stems from two main factors.

First, the provision of too many software functions. Some observers may not see this fat as harmful, yet it adds significantly to the sources of risk. This includes the increased mental load placed upon users and especially maintainers.

Second, unnecessary complexity arises from the mapping of application functions—via programming languages, operating systems, protocols, and middleware—onto poor or inappropriate execution platforms.

The first source is impossible to address with a technical solution. Hopefully, software developers will someday learn that more functionality is not necessarily better. Fortunately, we can address the second source via restructuring of the hardware and software architecture complex.

ENGINEERING COMPUTER-BASED SYSTEMS

Today's commercial products are laden with risk-ridden white elephant structures that seem too costly to replace. However, without architectural reform, we cannot improve upon the risk situation. If we continue to add software to the current mainstream product base, we will bury ourselves deeper into the black hole of complexity.

Improved architectures are no panacea. Just as important is the disciplined structuring and execution of life cycle processes with the support of methods and tools. Holistic system engineering of complex computer- and communication-based systems is vital. Only such an engineering approach accounts for system needs and requirements, trade-off analysis of alternative solutions, architecture definition, function allocation to parts of the architecture, process management, and so on.

Clearly, deploying inappropriate

architectural structures complicates systems engineering and always makes system solutions more costly and risky. One measure of computer-based system complexity is the amount of glue code that must be developed and maintained to get parts of a complex system to work together.

ARCHITECTURE RESTRUCTURING

In 1973, computer pioneers Jim Horning and Brian Randell wrote, "The appropriate use of structure is still a creative task, and is, in our opinion, a central factor of any system designers' responsibility." Twenty-five years later, this opinion is still valid. Designers must look at systems as a whole to make sensible trade-offs. Proper layering and allocation of functions is essential to reducing complexity and mitigating risk.

A full treatment of restructuring is beyond the scope of this column. I focus on three critical aspects that can reduce unnecessary complexity. These are not the only factors. However, in my experience, treating them improperly always leads to unnecessary complexity in nontrivial systems.

Synchrony and asynchrony

The first aspect involves the appropriate use of time- and event-driven approaches with respect to the goals of the systems in which they are deployed. Both approaches are valid and often need to be combined in complex systems. It is vital that developers make the appropriate choice.

When functionality requires time-deterministic processing and communication, employing a synchronous strategy is preferable. In contrast, systems supplying service-oriented functions that do not require time-determinism should deploy an asynchronous strategy.

This should be a rule of thumb. However, it is clear that adding large quantities of functions in a synchronous solution adds significant complexity and can, due to lack of adequate resources, make the solution intractable.

Although adding functions in an asynchronous solution is much more straightforward, the solution is nondeterministic. This often results in an inability to conveniently meet stringent time demands. Further, due to flexibility, there is a tendency to introduce unnecessary fat into the system; thereby complicating utilization and maintenance.

Languages and their "machines"

The second aspect involves the continued use of low-level languages including C and C++. The fact that the instruction sets of current microprocessors are poor "target machines" for higher level languages leads to a continuing motivation to program at low levels of abstraction. This applies to both CISC and RISC architectures. Deploying low level languages for programming is a major source of unnecessary complexity.

How many large programming projects have failed or fallen short because the complexities of the programming task were underestimated? How much unnecessary time do maintainers expend to comprehend the intricacies of large quantities of low-level language code? How safe is it to change structures that are not well-understood? These and other risk-related questions indicate that we must lift the level of programming abstraction.

In principle, the industry has suffered from stymied language development for almost 25 years. Java is only a minor improvement in comparison to what is required. No other language has significantly penetrated the market. There are several reasons for this, but a prime reason is the mismatch (semantic gap) between microprocessor instruction sets and higher level languages. Unnecessarily complex compilers must generate significant amounts of code to reconcile the semantics of the language and the RISC or CISC microprocessor.

Moving forward will require the introduction of instruction sets oriented to higher level languages. Further, it will require flexibility in the microarchitectures to permit convenient realization of multiple instruction sets.

Interpretation machines oriented to higher level languages must be so cost- and performance-effective as to render deploying lower level languages economically unreasonable. This is not a new idea, just one that has been and is now even more technically feasible.

Function definition and distribution

The third vital aspect is the ability to implement sets of system and even application functions as machines. By implementing them via machines with well-defined semantics, the set of functions become prime candidates for standardization. They provide a benchmark from which to measure implementation conformance.

To provide these well-defined machines, software and hardware developers must critically examine implementation trade-offs. Placing key functions closer to hardware can provide highly accelerated performance. Operating system, protocol, and middleware functions, for example, are prime candidates.

PROCESSOR ARCHITECTURE

Processor architectures must be selected for their appropriateness in respect to the goals of the system to be implemented. For most real-time processing and communication (including digital signal processing), the processor should be as simple as possible. While it may seem economically attractive to deploy inexpensive CISC chips in such environments, they are most often an overkill. Thus, unnecessarily complex system software and support tools lead to the increased costs and associated risks.

FUNCTION INTEGRATION MICROPROCESSORS

In service-oriented computing, CISC and RISC processors are dominant. However, to enable architecture restructuring, a new form of processor must evolve; Function Integration Microprocessors. FIMs provide a flexible microprogrammable machine adaptable to a wide variety of tasks.

Using FIMs, we can develop machines that can be efficiently programmed with high-level languages (past, present or future). They could emulate the capabilities of existing instruction sets. FIMs can also provide hardware support for function distribution; for example, for operating systems and protocol machines.

The implementation of Java Bytecode via a microprogrammable processor as well as the redistribution of multimedia functions from software to hardware represent ministeps toward restructuring. It remains to be seen how well the new Hewlett-Packard/Intel 64-bit processor will function as a flexible FIM. The microprogramming of Merced may be too complex.

Significant inspiration for FIMs can come from a few processors developed in the late 1960s and early 1970s. Some of these processors provided FIM properties. For example, the Swedish Datasaab FCPU (Flexible Central Processing Unit) was, in 1973, a 64-bit architecture providing high-level hardware functions and facilities for function distribution (H.W. Lawson, and B. Magnhagen, “Advantages of Structured Hardware,” *Proc. Second Ann. Symp. Computer Architecture,* IEEE Computer Society Press, Los Alamitos, Calif., 1975). It also included facilities for flexible and simplified microprogramming of machines oriented toward higher level languages.

With appropriate FIMs, it could be fruitful to develop processor-neutral machine designs for heavily used languages such as Cobol, Fortran, Ada, and even C and C++. This could be an alternative to pure emulation of multiple, older instruction repertoires. Coupled with compilers that are aware of and treat implementation incompatibilities, this may be a feasible (although inelegant) approach to dealing with legacy code.

As for operating system and protocol functions, inspiration can be obtained for defining machines and instructions for primitive functions of the desktop paradigm, scripting languages, Distributed Interactive Simulation, the US DOD Common Operating Environment, and Internet protocols (including cryptography standards).

At the time the FCPU and other flexible, microprogrammable processors were designed, hardware resources imposed severe limitations. Modern VLSI technology as well as today’s CAD tools make designing and producing similar processors no more complex (and likely easier) than doing so for today’s complex superscalar CISC and RISC processors.

REQUIREMENTS AND SPECIFICATIONS NEEDED

We need requirements and specifications for the machines and instruction sets. These should be publicly available (as with Java Bytecode) to all hardware and system software suppliers. Those suppliers implementing the best FIMs, associated system software and applications should be the most successful at reducing unnecessary complexity, while also winning in the marketplace.

Naturally, a move in these directions would have significant standardization implications. The approach will not succeed unless it can be enforced and lead to compatibility and interoperability. Thus testing, verification, and (hopefully) certification of conformance to machine semantics is essential. An agency or industry organization must assume responsibility for this critical work.

NEW R&D OPPORTUNITIES

The redirection of the creative capabilities of computer engineers and scientists offers exciting opportunities. Well-defined machines implemented close to the hardware can address problems of security and safety.

New means of testing and debugging that facilitate and reduce the cost of software maintenance can be built into the FIM framework.

By reducing total complexity, it is quite probable that FIMs, even running at slower clock speeds, will outperform today's complex hardware and software. Instruction traffic to be interpreted by the processor is significantly reduced. Finally, exploitation of large-grained parallelism can lead to even greater system performance gains.

REVOLUTION OR EVOLUTION?

As complicated and multifaceted as computers have become, it is impossible to be revolutionary; change must be evolutionary. The FIM approach provides a means for backward compatibility as well as an opportunity to move to new solutions.

As an example of the need for evolution, consider the severe problems associated with upgrading the US air-traffic control hand-over systems. Running on hardware from the late 1960s, the software is written largely in Jovial and assembly. Had the project included an evolutionary solution (based on FIMs, for example) there could have been a clear path to first work in parallel with the existing solution and then incrementally evolve to meet tomorrow's needs. Instead, the old system remains in place and no one dares touch the software. Society continues to be exposed to the risks associated with this white-elephant hardware and complex software.

Shall we wait for catastrophes or work toward corrective action now? Will it take governmental investigations and attempts at legislation? Complex applications are being built on poor foundations. We must establish firm, reliable foundations before we can rely upon off-the-shelf software components. By defining the machines upon which these components execute, we pin down concrete semantics and thus make it meaningful to test and validate software components. This is a key to the secure, cost-effective reuse of software. Dare we wait? Architectural restructuring will not provide salvation from all of the problems associated with computer-based system complexity. However, it is clear that without restructuring, there will be no salvation. v

Harold W. (Bud) Lawson is an independent consultant in Stockholm. An IEEE and ACM Fellow, he has contributed to several pioneering hardware, software, and application-related endeavors. Contact him at bud@lawson.se.

Note: This article was first published in the Binary Critic Column of the February 1998 issue of IEEE Computer.

Aktuellt kring SESAM

Från rådsmötet 13 maj

Vårens rådsmöte var återigen mycket välbesökt. Två av de viktigare punkterna som behandlades var:

Rådet blev överens om ett förslag till nytt ramavtal för SESAM fortsatta verksamhet. Det kommer att skickas ut till intressenterna för påtecknande inom kort. Det principiella innehållet är i huvudsak det samma som i gällande avtal, men större klarhet i begrepp och benämningar har eftersträvats samt viss allmän modernisering gjorts (hemsidan t ex). Dessutom har ju en hel del ändringar i företagskonstellationer etc skett under de gångna 10 åren, vilket kommer att avspeglas i det undertecknade avtalet.

Efter en redovisning av mötesfrekvens och deltagande i arbetsgrupperna som visade på ett deltagande i genomsnitt kanske hälften eller mindre, av det avtalade, följde en mycket livlig debatt med många idéer om vad som kunde göras för att åstadkomma en aktivering. En grupp med Claes Wadsten Celsius Aerotech, Magnus Ericson Ericsson Saab Avionics och Håkan Edler CTH fick i uppgift att till VU-mötet den 20/8 redovisa förslag till åtgärder. Alla SESAM-iter uppmanas att lämna synpunkter till gruppens medlemmar.

Bland övriga punkter som togs upp kan nämnas en genomgång av planerna för SESAM deltagande i utställningen vid Ada-Europe konferens i Uppsala den 8-11 juni och läget betr den CD-ROM med utbildningsmateriel för Ada 95 och diverse andra nyttoprogram som kommer att levereras till SESAM medlemmar och delas ut vid Ada-Europe konferensen i juni.

Protokollet från rådsmötet kommer att läggas ut på SESAM hemsida.

FoTA-läget

Projektförslagen till programmet för Forskning och Teknikutveckling går nu in på upploppet, d v s FMV och FM närmar sig tidpunkten att fatta beslut om hur förslagen skall hanteras och prioriteras. Det finns förslag till projekt som tillsammans kostar mer än dubbelt så mycket som tillgängliga medel. I valet mellan att stryka projekt eller att pruta ner summorna, föredrar FMV en tredje lösning där FM/FMV bidrar till 75 % av projektkostnaderna, undantaget universitet och högskolor samt små/fåmansföretag, där bidrag till hela kostnaden kan bli aktuellt, liksom för speciella projektledningsinsatser. Dessutom förordar FMV en komprimering av tidsskalan så att tillgängliga medel förbrukas före 991231. Finansiering för verksamhet som sträcker sig efter den tidpunkten lämnas att ev lösas senare. Detta innebär att projekt som sträcker sig efter 991231 måste ha ett värdefullt användbart delresultat vid den tidpunkten (om det inte skulle bli någon fortsättning), vilket kan leda till behov av viss omplanering i projektförslagen. Dessa alternativ har just börjat diskuteras mellan FMV och försvarsindustrierna och inga beslut är fattade. Intrycket hittills är att 25 % medfinansiering från industriparterna inte anses orimlig och också kan tas som ett tecken på seriöst intresse från förslagsställarna för att projekten blir genomförda.

Listan över aktuella inkomna projekt finns på annan plats i bladet. De intressenter som önskar delta i ett projekt och tror/vet att man inte finns med i det inlämnade förslaget, bör snarast anmäla det till kontaktpersonen för resp projekt (framgår av tidigare spridda listor), innan ”tåget går”!

Ordf.

Handbok Programvarusäkerhet (H PgmvSäk)

Under 1997 genomförde FMV en studie betr. krav vid anskaffning av säkerhetskritiska programsystem.

Denna resulterade i en första, icke fastställd utgåva av Handbok Programvarusäkerhet, vilken utöver krav på olika aktörer, de produkter dessa framställer och de processer, som därvid tillämpas ger en översikt över de i sammanhanget mest relevanta standarderna och handledningarna.

En andra etapp har under våren inletts. Under denna kommer en remissutgåva av handboken att färdigställas och en exempelsamling med kravformuleringar för FMVs olika typer av specifikationer att extraheras. Den som är intresserad av handboken i dess förstudieversion (1.0) kan vända sig till Anette Marke, FMV:InfosystT. Synpunkter kan lämnas direkt till mig.

Inga-Lill Bratteby-Ribbing, FMV
(bratteby-ribbing@acm.org)

SESAM hemsida

Tyvärr har som Ni säkert upptäckt vår hemsida inte blivit uppdaterad sedan i höstas p g a att det fordrats särskild personal för detta. Nu finns dock en maskinell lösning där uppdatering skall kunna ske direkt från sekretariatet, vilket bör innebära att hemsidan kan få en mer acceptabel aktualitetsgrad.

Rättelse om MACS

I artikeln om MACS i föregående nummer hade det råkat insmyga sig ett förargligt fel. Det stod att realtiden störs i 50 millisek av debuggern; det är alltså fel vilket säkert de flesta insåg. Det skall vara 50 mikrosek.

Kalender

980608—11	Ada-Europe‘98 konferens i Uppsala
980814	Ag Programmerin, Ericsson Saab Avionics
980820	VU på FMV
980827	Ag Process/Metrik, Celsius Aerotech i Arboga
980914—17	SEI Annual Symposium, Pittsburgh, PA
980925	Ag Återanvändning vid Celsius Aerotech, Arboga alt. Stockholm
981021	SESAM/AiS höstseminarium i Stockholm
981022	SESAM-rådets höstmöte hos FMV Stockholm
981108—12	ACM SIGAda, Washington D.C. (Ex TRI-Ada)
981127	Ag Återanvändning vid Sjöland & Thyselius, Stockholm