Future Generation Computer Systems 9 (1993) 105–117
North-Holland

The Japanese National Fifth Generation Project: Introduction, survey, and evaluation

Edward Feigenbaum [a] and Howard Shrobe [b]

[a] *Knowledge Systems Laboratory, Stanford University, 701 Welch Road, Building C, Palo Alto, CA 94304, USA*
[b] *AI Lab, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, USA*

Abstract

Projecting a great vision of intelligent systems in the service of the economy and society, the Japanese government in 1982 launched the national Fifth Generation Computer Systems (FGCS) project. The project was carried out by a central research institute, ICOT, with personnel from its member-owners, the Japanese computer manufacturers (JCMs) and other electronics industry firms. The project was planned for ten years, but continues through year eleven and beyond. ICOT chose to focus its efforts on language issues and programming methods for logic programming, supported by special hardware. Sequential 'inference machines' (PSI) and parallel 'inference machines' (PIM) were built. Performances of the hardware-software hybrid was measured in the range planned (150 million logical inferences per second). An excellent system for logic programming on parallel machines was constructed (KL1). However, applications were done in demonstration form only (not deployed). The lack of a stream of applications that computer customers found effective, and the sole use of a language outside the mainstream, Prolog, led to disenchantment among the JCMs.

Keywords. FGCS; inference engine; logic programming; parallel architecture.

1. Introduction

In 1981, the emergence of the government-industry project in Japan known as Fifth Generation Computer Systems was unexpected and dramatic. The Ministry of International Trade and Industry (MITI) and some of its scientists at Electrotechnical Laboratory (ETL) planned a project of remarkable scope, projecting both technical daring and major impact upon the economy and society.

This project captured the imagination of the Japanese people (e.g. a book in Japanese by Junichiro Uemae rcounting its birth was titled *The Japanese Dream*). It also captured the attention of the governments and computer industries of the USA and Europe, who were already wary of Japanese takeovers of important industries. A book by Feigenbaum and McCorduck, *The Fifth Generation*, was a widely-read manifestation of this concern [3]. The Japanese plan was grand but it was unrealistic, and was immediately seen to be so by the MITI planners and ETL scientists who took charge of the project. A revised planning document was issued in May 1982 that set more realistic objectives for the Fifth Generation Project.

Newton's third law, loosely interpreted to apply to the beliefs of people (and journalists), says that grand actions generate equal and opposite reactions. Ten years later, the opposite reactions have grown into a loud chorus of negative assessments of the achievements of the Fifth Generation Project that have appeared in scientific journals, in trade magazines, and in the popular press. These reactions are (symmetrically) unrealistically negative.

We believe that the Fifth Generation Project will have significant long term implications for the computer industry in Japan. Several of MITI's major goals were achieved. The foundation technology called for in the 1982 plan was brought into existence. By the end of the project, the hardware and software performed more or less as

0376-5075/93/$06.00 © 1993 – Elsevier Science Publishers B.V. All rights reserved

originally planned. As important, the original (but tacit) educational goal of MITI was achieved. Thousands of young Japanese engineers were trained in the advanced concepts of computer science and technology that they were not being taught in their universities; and they were trained to think and invent in a creative open-ended way. At a conference in Tokyo in 1986, one of MITI's main planners of the Fifth Generation Project, Mr. Konishi, told his audience that the education goal was MITI's main goal in funding the project!

At the project's 10th anniversary conference in 1992, Japan's foremost corporate executive in information technology research told us, "Do not underestimate the importance of the Fifth Generation Project; it has trained the next generation of Japanese computer architects."

The goals for a broad economic and societal application of intelligent systems (realized by logic programming) were not achieved. Of course this is the dimension that has led to the disenchantment. Where are the language, speech, vision, and problem-solving applications? They did not emerge in the first decade, but that does not mean that they will not. The fruits of the PIPS project (Pattern Information Processing Project) of the 1970s did not appear until the mid-to-late 1980s in the form of a superb national image processing capability. Perhaps the Fifth Generation Project was somewhat early with its advanced goals. We don't yet know where the capabilities that it developed will go in the Japanese industry and computer science.

In 1992, we served on an American visiting committee (called a JTEC committee) that made an intensive six day visit to Japan to assess Japanese applications of and research in knowledge-based systems. As part of that trip, we visited ICOT, and spoke with several of its leaders. We and they knew each other well as scientific colleagues, so the discussions were direct, open, and frank. Much of what they had to say is incorporated in the survey we will now present; and much of the text is taken from the report of that committee [4].

2. History and goals

ICOT was founded as the central research laboratory of the Fifth Generation Computer Systems project in 1982. It is now celebrating the end of its eleventh year. ICOT's life may be extended for a total of 3 more years with a reduced staffing level.

The Fifth Generation Computer Systems project was motivated by the observation that "Current computers are extremely weak in basic functions for processing speech, text, graphics, picture images, and other non-numeric data, and for artificial intelligence type processing such as inference, association, and learning." To address these shortcomings, the Fifth Generation project was commissioned to build the prototypes for a new (the fifth) generation of hardware and software. Early ICOT planning documents [6] identify the following requirements:

(1) To realize basic mechanisms for inference, association, and learning in hardware and make them the core functions of the fifth generation computers.
(2) To prepare basic artificial intelligence software to fully utilize the above functions.
(3) To take advantage of pattern recognition and artificial intelligence research achievements, and realize man-machine interfaces that are natural to man.
(4) To realize support systems for resolving the 'software crisis' and enhancing software production.

A fifth generation computer system in this early ICOT vision is distinguished by the centrality of

(1) Problem solving and inference,
(2) Knowledge-base management,
(3) Intelligent interfaces.

Such a system obviously requires enormous computing power. In ICOT's view, it also required a new type of computing power, one more symbolic and inferential in character than conventional systems. Also, the system was explicitly assumed to rely on very large knowledge bases and to provide specialized capabilities for knowledge and data base management. Finally, fifth generation systems were assumed to interact with people in a more human like manner, using natural language in both printed and spoken form.

The 1982 ICOT planning document [6] calls for a three tier system: At the base is a tier for Knowledge-Base Systems which includes a parallel database management hardware and knowledge base management software. This system was

envisioned as "... A database machine with 100 to 1000 Gb capacity" and able"... to retrieve the knowledge bases required for answering a question within a few seconds."

"The intention of software for the knowledge base management function will be to establish knowledge information processing technology where the targets will be development of knowledge representation systems, knowledge base design and maintenance support systems, large-scale knowledge base systems, knowledge acquisition experimental systems, and distributed knowledge management systems... One particularly important aim will be semi-automated knowledge acquisition, that is, systems will be equipped with a certain level of learning functions."

Built on top of this are a problem solving and inference tier, including hardware for parallel inference, abstract datatyping and dataflow machines; this tier includes software for a Fifth Generation kernel language (see below), cooperative problem solving mechanisms and parallel inference mechanisms.

The final tier is the Intelligent Man–Machine interface system. This was supposed to include dedicated special purpose hardware for speech and other signal processing tasks and software for natural language, speech graphics and image processing:

"The intelligent interface function will have to be capable of handling communication with the computer in natural language, speech, graphics, and picture images so that information can be exchanged in a ways natural to man. Ultimately the system will cover a basic vocabulary (excluding specialist terms) of up to 100,000 words and up to 2000 grammatical rules, with a 99% accuracy in syntactic analysis.

"The object speech inputs will be continuous speech in Japanese standard pronunciation by multiple speakers, and the aims here will be a vocabulary of 50,000 words, a 95% recognition rate for individual words, and recognition of processing within 3 times the real time of speech."

"The system should be capable of storing roughly 10,000 pieces of graphic and image information and utilizing them for knowledge information processing."

These three tiers were then supposed to support a very sophisticated program development environment to raise the level of software productivity and to support experimentation in new programming models. Also the basic three tiers were supposed to support a variety of basic application systems: Those listed in the 1982 document include: Machine Translation, Consultation Systems, and Intelligent programming systems (including automated program synthesis and verification).

The application systems were assumed to be of significant size and sophistication with the following being typical features:

- Number of objects: many thousands
- Inference rules: 10,000 or more
- Semi-automated knowledge acquisition
- Interfaces with system: Natural language and speech
- Vocabulary size: 50,000 words or more.

3. The commitment to logic programming

Achieving such revolutionary goals would seem to require revolutionary techniques. Conventional programming languages, particularly those common in the late 1970s and early 1980s offered little leverage. The requirements clearly suggested the use of a rich, symbolic programming language capable of supporting a broad spectrum of programming styles. Two candidates existed: LISP which was the mainstream language of the US Artificial Intelligence community and Prolog which had a dedicated following in Europe. LISP had been used extensively as a systems programming language and had a tradition of carrying with it a featureful programming environment; it also had already become a large and somewhat messy system. Prolog, in contrast, was small and clean, but lacked any experience as an implementation language for operating systems or programming environments.

It was decided early on that ICOT would base its work on Logic Programming rather than Lisp; that it would build on but significantly extend Prolog. Also it was decided that the Logic Programming language would be a 'Kernel Language' that would be used for a broad spectrum of software, ranging from the implementation of the system itself up through the application layers.

In practice, ICOT's central focus became the development of such a logic programming kernel language and the development of hardware tailored to the efficient execution of this language. The system's performance target was to be from 100 Mega LIPS (logical inferences per second, i.e. simple Prolog procedure calls) to 1 Giga LIPS. (As a reference point, ICOT documents estimate that 1 Logical Inference takes about 100 instructions on a conventional machine; a 1 MLIPS machine would therefore be roughly equivalent to a 100 MIPS processor, although this comparison may confuse more than it reveals.) The rather reasonable assumption was made that achieving such high performance would require parallel [1] processing: "... the essential research and development will concentrate... on high-level parallel architectures to support the symbol processing that is the key to inference." Furthermore, the assumption was made that achieving the desired performance target would require about 1000 processing elements per system given reasonable assumptions on the performance of a single such processing element.

ICOT's development plans were segmented into three phases. The goal for the initial phase was to develop a 'Personal Sequential Inference Machine' (PSI), i.e. a workstation tailored to efficient execution of a sequential logic programming language. This phase was also supposed to develop the system software for high capability programming environments for Fifth Generation software. The initial considerations of parallel systems were also to begin during this stage.

In the second phase, a refined Personal Sequential Inference would be developed, the model for parallel programming would be settled upon and initial exploratory parallel architectures would be prototyped.

The third phase would build the Parallel Inference Machines (PIM). This would include not only the hardware effort, but a parallel operating system and a second generation kernel language appropriate for parallel processing.

[1] In the early 1980s parallelism was the 'coming thing' in computer science. Since a goal of the fifth generation project was to 'jump headlong into the future of computing' it was not only necessary to embrace parallelism, it was desirable.

4. Accomplishments

4.1. Accomplishments in hardware

During the first 3 year phase of the project, the Personal Sequential Inference machine (PSI 1) was built and a reasonably rich programming environment was developed for it (*Fig. 1*).

To put this effort in context, we compare it to the US project which it most resembles: the MIT Lisp Machine. The MIT project had begun in the late 1970s and had just reached commercialization at the time of ICOT's inception. Like the MIT machine, PSI was a microprogrammed processor designed to support a symbolic processing language. The symbolic processing language played the role of a broad spectrum 'Kernel language' for the machine, spanning the range from low level operating system details up to application software. The hardware and its microcode were designed to execute the kernel language with high efficiency. The machine was a reasonably high performance work station with good graphics, networking and a sophisticated programming environment.

What made PSI different was the choice of language family. Unlike more conventional machines which are oriented toward numeric processing or the MIT machine which was oriented

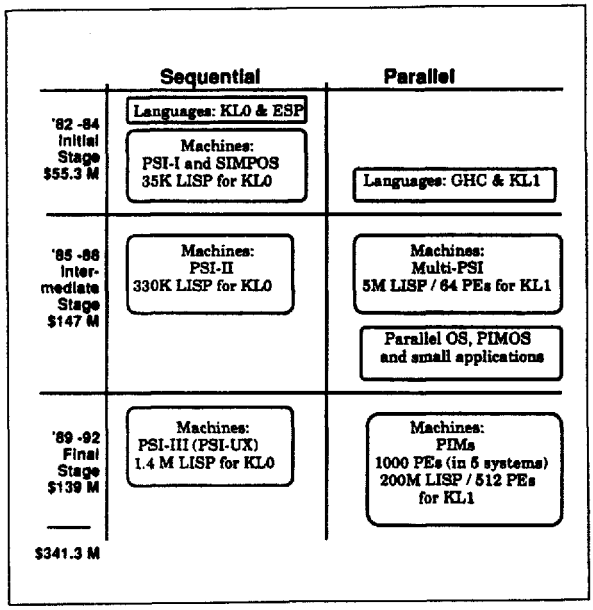

Fig. 1. Major accomplishments.

towards LISP the language chosen for PSI was Prolog. The primary appeal of Prolog-like languages to ICOT was the analogy between the basic operations of Prolog and simple rule-like logical inferencing. A procedure in such a language can be viewed as simply reducing a goal to its subgoals. Given the emphasis on inference as a key component of the FGCS vision, the choice seemed quite natural. However, the choice of a logic programming framework for the kernel language was a radical one since there had been essentially no experience anywhere with using logic programming as a framework for the implementation of core system functions.

PSI-1 achieved a performance of about 35K LIPS, comparable to DEC-10 Prolog, the Prolog performance of the Symbolics 3600 (one of the follow-ons to the MIT Lisp Machine) or Quintus's Prolog implementation for Sun-3 class machines. This was fast enough to allow the development of a rich operating system and programming environment, but still quite slow compared to the Phase 3 goals (1000 processors achieving 1 GLIPS, implies at least 1 MLIPS per processor). Two extended Prolog-like languages (ESP and KL0) were developed for PSI-1. ESP (Extended Self Contained Prolog) included a variety of features such as coroutining constructs, non-local cuts, etc. necessary to support system programming tasks as well as more advanced Logic Programming. SIMPOS, the operating system for the PSI machines, was written in ESP.

Several hundred PSI machines were built and installed at ICOT and collaborating facilities; and the machine was also sold commercially. However, even compared to specialized Lisp hardware in the US, the PSI machines were impractically expensive. The PSI (and other ICOT) machines had many features whose purpose was to support experimentation and whose cost/benefit tradeoff had not been evaluated as part of the design; the machines were inherently non-commercial.

During Phase 1, it was decided to explore an 'And-Parallel' approach to parallel logic programming. To simplify, this means that the subgoals of a clause are explored in parallel with shared variable bindings being the means of communication. The process solving one subgoal can communicate with a process solving a sibling subgoal by binding a shared variable to a concrete value. It was also observed that subgoals would have to spread out across the network of processors constituting the parallel machine and that it would require careful control to avoid the buildup of communication bottlenecks. By the end of Phase 1, the form of the parallel kernel language was clarified: it was to be a 'Flat Guarded Horn Clause' (FGHC) language. A Flat Guarded Horn Clause consists of three parts:

(1) The head,
(2) the guard,
(3) the body.

The head plays exactly the same role as the head of a Prolog Clause: it identifies a set of goals for which the clause is suitable (i.e. those goals which unify with the head). The guard and body collectively play the role of the body of a Prolog clause, i.e. they are a set of subgoals whose truth implies the truth of the head. However, the body of the clause is not executed until all variables in the guard are instantiated and all literals in the guard are satisfied. In the case where two (or more) clauses have heads that unify with the same goal, only the body of that clause whose guard is first satisfied will execute (hence the name guarded horn clause. 'Flat' means that the guard can only contain built-in predicates, rather than those which are evaluated by further chaining. This greatly simplifies the mechanisms, without significantly reducing the expressive power).

The execution of a FGHC program is summarized in four rules:

(1) *Relevance:* Those clauses whose head unifies with a goal are potentially executable.
(2) *Synchronization:* Until the caller has sufficiently instantiated the variables to allow the guard part of the clause to execute, execution of the guard is suspended.
(3) *Selection:* If there are more than one potentially executable clauses for a goal, that clause whose guard succeeds first will execute its body and the bodies of all other competing clauses will never execute.
(4) *Parallelism:* The subgoals in the body are executed in parallel.

Figure 2 shows a set of FGHC for a prime sieve algorithm and how they begin to elaborate a parallel process structure. One should notice that this interpretation model does not lead to an automatic search mechanism as in Prolog. In Prolog all relevant clauses are explored, and the

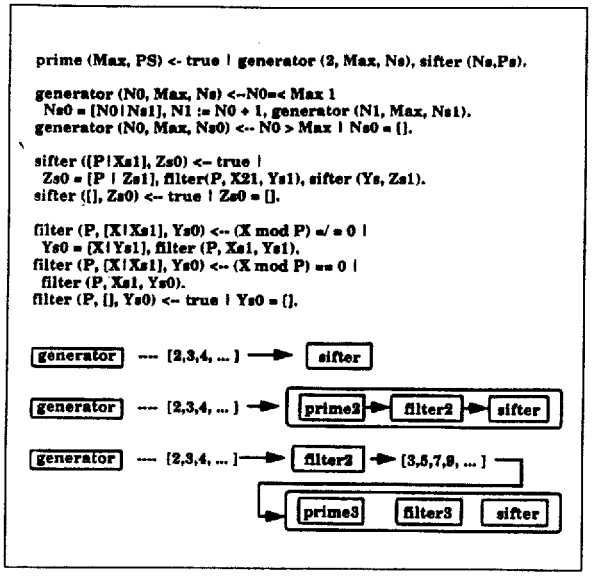

Fig. 2. Prime sieve algorithm using guarded horn clauses.

order of exploration is specified by the programming model. In FGHC only a single relevant clause is explored; ICOT has had to conduct research on how to recapture search capabilities within the FGHC framework.

The second 3 year phase saw the development of the PSI 2 machine which provided a significant speedup over PSI 1. Towards the end of Phase 2 a parallel machine (the Multi-PSI) was constructed to allow experimentation with the FGHC paradigm. This consisted of an 8×8 mesh of PSI 2 processors, running the ICOT Flat Guarded Horn Clause language KL1 (not to be confused with the knowledge representation language KL-ONE developed at Bolt Beranek and Newman). Multi-PSI supported the development of the ICOT parallel operating system (PIMOS) and some initial small scale parallel application development. PIMOS is a parallel operating system written in KL1; it provides parallel garbage collection algorithms, algorithms to control task distribution and communication, a parallel file system, etc.

Phase 3 has been centered around the refinement of the KL1 model and the development of massively parallel hardware systems to execute it. KL1 had been refined into a three level language. KL1-b is the machine level language underlying the other layers. KL1-c is the core language used to write most software; it extends the basic FGHC paradigm with a variety of useful features such as a macro language.

KL1-p includes the 'pragmas' for controlling the implementation of the parallelism. There are three main pragmas. The first of these is a meta-level execution control construct named 'shoen' which allows the programmer to treat a group of processes (i.e. a goal and its subgoals) as a unit of execution control. (A shoen is created by calling a special routine with the code and its arguments; this creates a new shoen executing the code and all generated subgoals. These subgoals are, however, running in parallel). A failure encountered by any sub-process of a shoen is isolated to that shoen. Each shoen has a message stream and a report stream by which it communicates with the operating system; shoens may be nested but the OS treats the shoen as a single element. Suspending a shoen results in the suspension of all its children, etc. Fine grain process management is handled by the shoen, freeing the OS from this responsibility.

The second pragma allows the programmer to specify the priority of a goal (and the process it spawns). Each shoen has a minimum and maximum priority for the goals belonging to it; the priority of a goal is specified relative to these.

The third pragma allows the programmer to specify the processor placement for a body goal. This may be a specific processor or a logical grouping of processors. All three of these pragmas are 'meta execution control' mechanisms which themselves execute at runtime; KL1-p thus allows dynamic determination of the appropriate priority, grouping and placement of processes.

Much of the current software is written in higher level languages embedded in KL1, particularly languages which establish an object orientation. Two such languages have been designed: A'UM and AYA. Objects are modeled as processes communicating with one another through message streams. The local state of an object is carried along in the cyclical call chain from dispatching routine to service subroutine back to dispatching routine. Synchronization between processes is achieved through the binding of variables in the list structure modeling the message stream. (See also Bal's contribution in this issue [1]).

Five distinct Parallel Inference Machines (PIMs) have been developed to execute KL1,

each built by a commercial hardware vendor associated with ICOT. They vary in processor design and communication network. The abstract model of all PIMs consists of a loosely coupled network connecting clusters of tightly coupled processors. Each cluster is, in effect, a shared memory multiprocessor; the processors in the cluster share a memory bus and implement a cache coherency protocol. Three of the PIMs are massively parallel machines: PIM/p, PIM/m and PIM/c. PIM/k and PIM/i are research machines designed to study specific intracluster issues such as caching and bus communication. Multi-Psi is a medium scale machine built by connecting 64 Psi's in a mesh architecture. PIM/m and Multi-Psi do not use a cluster architecture (but may be considered as degenerate cases having one processing element per cluster).

The main features of their communication systems are shown below:

	Topology	#Cluster	#PE	Memory/ cluster
PIM/p	hypercube×2	64	512	256 Mb
PIM/m	mesh	256	256	80 Mb
PIM/c	crossbar	32	256	160 Mb
PIM/k		4	16	1 Gb
PIM/i		2	16	320 Mb
MultiPsi	mesh	64	64	80 Mb

	#PEs/ Cluster	#Network Interfaces/ Cluster	Transfer Rate
PIM/p	8	8	33 Mb/sec×2
PIM/m	1	1	8 Mb/sec
PIM/c	8	1	40 Mb/sec
PIM/k	16		
PIM/i	8	1	
MultiPsi	1	1	10 Mb/sec

Relevant features about the processing elements' implementation technology are shown below:

	Instruction Set	Cycle Time	Fabrication Technology	Line Width
PIM/p	RISC	60 nsec	standard cell	0.96 micron
PIM/m	CISC (ucode)	65 nsec	standard cell	0.8 micron
PIM/c	CISC (ucode)	50 nsec	gate array	0.8 micron
PIM/k	RISC	100 nsec	custom	1.2 micron
PIM/i	RISC	100 nsec	standard cell	1.2 micron
MultiPsi	CISC (ucode)	200 nsec	gate array	2.0 micron

It should be noted that the cycle times for the processing elements are relatively modest. Commercial RISC chips have had cycle times lower than these for several years (the lower the cycle time, the faster the instruction rate). Newly emerging processor chips (such as the DEC ALPHA) have cycle times as low as 5 nsec. Even granting that special architectural features of the PIM processor chips may lead to a significant speedup (say a factor of 3 to be very generous), these chips are disappointing compared to the commercial state of the art. The networks used to interconnect the systems have respectable throughput, comparable to that of commercially available systems such as the CM-5. In certain of the PIMs each processor (or processor cluster) can have a set of disk drives; this may allow more balance between processing power and I/O bandwidth for database applications, but there is as yet no data to either confirm or refute this. (See also Tick's contribution in this issue [9].)

4.2. *Accomplishments in software*

ICOT's software vision has been radical. In the conventional view the hardware supports an Operating System and language implementation(s). These in turn support a window system and in the best case perhaps a text editor, graphics editor and a user interface management system. The application developer must then span the entire remaining distance to the needs of the end users. This is tractable if the end user is expected to need no more than a spreadsheet and word processor.

However, in ICOT's initial vision, the end-user is expected to communicate with the computer using natural language and images. The computer is required to know a fair amount about the real world and to be capable of performing intelligently. To imagine building such applications, one must assume a much higher level starting point. Therefore, ICOT's software strategy is based on providing a much deeper set of software layers to the application developer (see *Fig. 3*). In this view, the application developer builds on top of automated deduction systems, constraint systems, and natural language systems.

So far we have been discussing the bottom most layer, that concerned with the operating system and language runtime system for parallel logic programming. On this foundation, ICOT has pursued research into

(1) Databases and knowledge base support,
(2) Constraint logic programming,

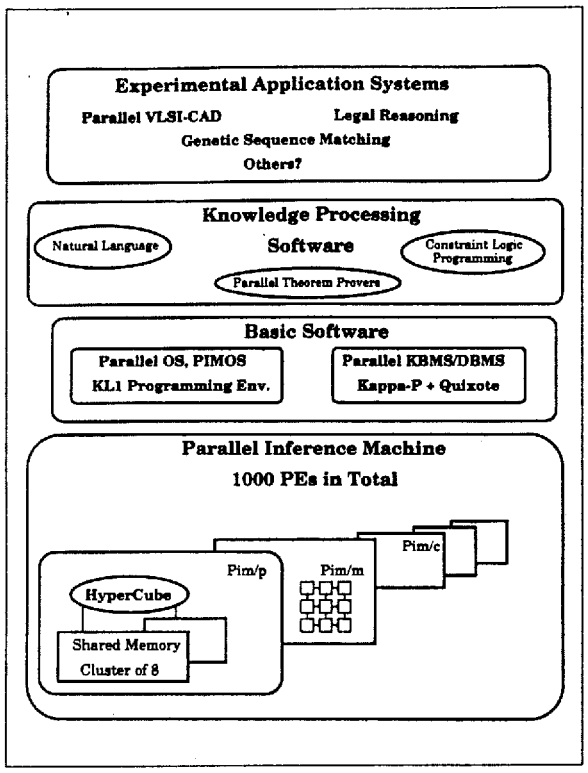

Fig. 3. ICOT's layered approach to software development.

(3) Parallel theorem proving, and
(4) Natural language understanding.

In the area of databases, ICOT has developed a parallel database system called Kappa-P. This is a 'nested relational' database system based on an earlier ICOT system called Kappa. Kappa-P is a parallel version of Kappa, re-implemented in KL1. It also adopts a distributed database framework to take advantage of the ability of the PIM machines to attach disk drives to many of the processing elements. Quixote is a Knowledge Representation language built on top of Kappa-P. It is a constraint logic programming language with object-orientation features such as object-identity, complex objects described by the decomposition into attributes and values, encapsulation, type hierarchy and methods. ICOT also describes Quixote as a Deductive Object Oriented Database (DOOD). Quixote and Kappa-P have been used to build a Molecular Biological Database, and a Legal Reasoning System. (See also Nishio's contribution in this issue [7].)

ICOT has been one of the world class centers for research into Constraint Logic Programming. All such languages share the idea of merging into a logic programming context constraint solvers for specific non-logical theories (such as linear equations or linear inequalities). Two languages of this type developed at ICOT are CAL (Constraint Avec Logique) which is a sequential constraint logic programming language which includes algebraic, Boolean, set and linear constraint solvers. A second language, GDCC (Guarded Definite Clauses with Constraints) is a parallel constraint logic programming language with algebraic, Boolean, linear and integer parallel constraint solvers.

Another area explored is that of Automatic Theorem Proving. ICOT has developed a parallel theorem prover called MGTP (Model Generation Theorem Prover). This is written in KL1 and runs on the PIMs. MGTP has obtained a more than 100 fold speedup on a 128 processing element PIM/m for a class of problems called 'condensed detachment problems'. MGTP is based on the Model Generation proving methods first developed in the SATCHMO system; however, the ICOT version uses the unification hardware of the PIMs to speed this up for certain very common cases. MGTP has been used as a utility in a demonstration legal reasoning system. It has also been used to explore non-monotonic and abductive reasoning. Finally, MGTP has been employed in some program synthesis explorations, including the synthesis of parallel programs. (See also Stickel's contribution in this issue [8].)

Natural Language Processing has been a final area of higher level support software developed at ICOT. There have been several areas of work:
(1) A Language Knowledge Base consisting of a Japanese syntax and dictionary
(2) A language tool box containing morphological and syntax analyzers, sentence generator, concordance system, etc.,
(3) A discourse system which rides on top of the first two. These are combined in a parallel, cooperative language understanding system using type inference. The dictionary has about 150,000 entries of which 40,000 are proper names (to facilitate analysis of newspaper articles). (See also Barnett and Yamada's contribution in this issue [2].)

On top of these tools a variety of *demonstration* application systems (not deployed applications) have been developed. These were shown

running on the PIM machines at the 10th anniversary FGCS conferences. They are discussed by van de Riet in this issue [11].

4.3. The legacy

When asked what they regarded as their legacy, the core achievement of the 10 year ICOT program, both Director Dr. Fuchi and his Deputy Dr. Chikayama said that it was KL1 (as opposed to the PIM hardware, the higher level software or the application demos).

There are at least three aspects to what has been achieved in KL1:

First the language itself is an interesting parallel programming language. KL1 bridges the abstraction gap between parallel hardware and knowledge based application programs. Also it is a language designed to support symbolic (as opposed to strictly numeric) parallel processing. It is an extended logic programming language which includes features needed for realistic programming (such as arrays). However, it should also be pointed out that like many other logic programming languages, KL1 will seem awkward to some and impoverished to others.

Second is the development of a body of optimization technology for such languages. Efficient implementation of a language such as KL1 required a whole new body of compiler optimization technology. Because there are several architecturally distinct PIMs (and the Multi-PSI) ICOT has been forced to develop a flexible implementation strategy for KL1. KL1 is compiled into an intermediate language KL1-B which plays a role similar to that which the WAM plays for sequential Prolog compilers. KL1-B is the abstract language implemented by each hardware system; it is a hardware model of a loosely coupled multiprocessor in which some processors are linked in tightly coupled clusters. To build a KL1 implementation, the architect must transform the abstract KL1-B specification into a physical realization; this is done semi automatically.

The third achievement is noticing where hardware can play a significant role in supporting the language implementation. Part of the architecture of the PIM (and PSI) processors is a tag checking component which provides support for the dynamic type checking and garbage collection needed to support a symbolic computing language. This asks for a small amount of additional hardware which provides a high degree of leverage for the language implementation without necessarily slowing down the processor or introducing undue complexity to the implementation. Such features, which might also support LISP and other more dynamic languages may eventually find their way into commercial processors.

At the time of the 10th anniversary Fifth Generation Conference all of the PIM designs were operational and were demonstrated running the applications listed above.

Table 1
Positive elements of evaluation

- ICOT has shown the ability of Japan to innovate in computer architectures.
- The ICOT architectures'peak parallel performance is within the range of the original performance goals.
- The PIMs represent an opportunity to study tradeoffs in parallel symbolic computing which does not exist elsewhere.
- KL1 is an interesting model for parallel symbolic computation, but one which is unlikely to capture the imagination of U.S. researchers.
- PIMOS has interesting ideas on control of distribution and communication which US researchers should evaluate seriously.
- ICOT has been one of the few research centers pursuing parallel symbolic computations.
- ICOT has been the only center with a sustained effort in this area.
- ICOT has shown significant (i.e. nearly linear) acceleration of non regular computations (i.e. those not suitable for data parallelism of vectorized pipelining).
- ICOT created a positive aura for AI, Knowledge Based Systems, and innovative computer architectures. Some of the best young researchers have entered these fields because of the existence of ICOT.

5. Evaluation

Positive elements of our evaluation of the Fifth Generation Computer Systems project are shown in *Table 1*. Negative elements are shown in *Table 2*.

The early ICOT documents suggested a push towards very advanced and very large scale knowledge based systems. This was a call for a quantum jump in the quality of services which computers provide.

Table 2
Negative elements of evaluation

- ICOT has done little to advance the state of knowledge based systems, or Artificial intelligence *per se*.
- ICOT's goals in the area of natural language were either dropped or spun out to EDR.
- Other areas of advanced man machine interfacing were dropped.
- Research on Very Large Knowledge bases were substantially dropped.
- ICOT's efforts have had little to do with commercial application of AI technology. Choice of language was critical.
- ICOT's architectures have been commercial failures. Required both a switch in programming model and the purchase of cost ineffective hardware.
- ICOT hardware has lagged behind U.S. hardware innovation (e.g. the MIT Lisp Machine and its descendants and the MIT Connection Machine and its descendants).
- Application systems of the scale described in the original goals have not been developed (yet).
- Very little work on knowledge acquisition.

In retrospect, it is clear that this was not achieved. The early documents discuss the management of very large knowledge bases, of large scale natural language understanding and image understanding with a strong emphasis on knowledge acquisition and learning. Each of these directions seems to have been either dropped, relegated to secondary status, absorbed into the work on parallelism or transferred to other research initiatives. [2]

ICOT set out to build a machine that could provide upwards of a 100 MLIPs of 'symbol crunching' power. However, there was no application whose clear need for such horsepower drove the development and whose success (or failure) would serve as the tangible evaluation of the effort. From our current perspective it is clear that there could not have been such an application, since any such application would necessarily have had to contain very large stores of knowledge; but in 1982 when the project began, there were no techniques available for capturing and managing such a large knowledge base. Even today, after more than a decade of research into knowledge representation, we have only a small base of experience and very few tested techniques for this task.

Thus the focus on a quantum jump in the quality of services provided by computers was replaced by a crisper but less ambitious one. The central perspective of ICOT's efforts has been to develop parallel symbolic programming (in particular, parallel logic programming) by developing a new language and by developing experimental hardware to support the language. Higher level facilities such as theorem provers, deductive databases and natural language support assumed a secondary role; applications assumed a tertiary role.

The set of demo applications for the PIM machines seem relatively routine. Even though each of these programs demonstrates the power of parallelism and even though each embeds some advance in parallel programming, when viewed as knowledge based systems, these systems bring little new to bear; they fail to qualitatively advance the kind of services which computers provide. For example, the multi-sequence matching program has a new approach to simulated annealing which uniquely capitalizes on the available parallelism; however, it knows essentially nothing about genetics and proteins and it provides only syntactically oriented services. Of the programs demonstrated, the legal reasoning system is the only one which might be fairly termed a knowledge based system. Here parallelism was used to accelerate both case retrieval and logical argumentation. Nevertheless, for all the computational power being brought to bear, the system did not seem to establish a new plateau of capability.

Were we to ask 'Has ICOT advanced the state of the art in Logic Programming and related techniques?' the answer would be 'clearly yes'. However, if we were to ask whether ICOT has *directly* accelerated the development of knowledge based technology in Japan so far, the answer would have to be no.

However, there are other questions which are also relevant:

(1) Has ICOT *indirectly* affected the state of knowledge based technology in Japan?
(2) Is the ICOT work likely to produce a platform which will ultimately accelerate knowledge based technology in Japan?

[2] Importantly, ICOT spun off a new research center, the Electronic Dictionary Research center, located in an adjacent building and run by a former ICOT manager.

(3) Has ICOT's work advanced the state of parallel processing technology in Japan (or elsewhere)?

The answer to (1) is almost certainly yes. Almost all companies we interviewed said that ICOT's work had little direct relevance to them. The reasons most frequently cited were: The high cost of the ICOT hardware, the choice of Prolog as a language and the concentration on parallelism. However, nearly as often our hosts cited the indirect effect of ICOT: the establishment of a national project with a focus on 'fifth generation technology' had attracted a great deal of attention for Artificial Intelligence and knowledge based technology. Several sites commented on the fact that this had attracted better people into the field and lent an aura of respectability to what had been previously regarded as esoteric. One professor in particular told us that AI now gets the best students and that this had not been true before the inception of ICOT and the Fifth Generation project.

Question (2) is considerably more difficult to answer. ICOT's work has built an elegant framework for parallel symbolic computing. Most A.I. people agree that without parallelism there will ultimately be a barrier to further progress due to the lack of compute power. However, this barrier does not seem imminent. Workstations with more than 100 MIPS of uniprocessor performance are scheduled for commercial introduction this year. With the exception of those sub-disciplines with a heavy signal processing component (e.g. vision, speech, robotics) we are more hampered by lack of large scale knowledge bases than we are by lack of parallelism. It is, however, quite possible that in the near future this will be reversed and we will be in need of parallel processing technology to support very large scale knowledge based systems. We will then be in dire need of programming methodology and techniques to capitalize on parallel hardware, and ICOT's work might well provide a solution.

Has the ICOT research significantly impacted parallel computing technology? There are arguments to be made on both sides of this question. On the positive side we can argue that KL1 is an interesting symbolic computing language. Furthermore, it is a parallel symbolic computing language and virtually no interesting work has been done elsewhere for expressing parallel symbolic computation. Another positive point is that ICOT has the test bed of the several PIM machines. This is an unusual opportunity; no other site has access to several distinct implementations of the same virtual parallel machine. It is not unreasonable to expect significant insights to emerge from this experimentation. Finally, we can add that ICOT has confronted a set of interesting technical questions about load distribution, communication and garbage collection in a parallel environment.

On the negative side we may cite several arguments as well. The ICOT work has tended to be a world closed in upon itself. In both the sequential and parallel phases of their research, there has been a new language developed which is only available on the ICOT hardware. Furthermore, the ICOT hardware has been experimental and not cost effective. This has prevented the ICOT technology from having any impact on or enrichment from the practical work.

Earlier we pointed out the similarities between the ICOT PSI systems and the MIT Lisp Machine and its commercial successors. It's noteworthy that only a few hundred PSI machines were sold commercially, while there were several thousand Lisp machines sold, some of which continue to be used in important commercial applications such American Express's Authorizers Assistant. The one commercial use we saw of the PSI machines was at Japan Air Lines, where the PSI-II machines were employed; ironically, they were remicrocoded as Lisp Machines. Furthermore, the MIT Lisp Machine acted as a catalyst, providing a powerful Lisp engine until better implementation techniques for Lisp were developed for stock hardware. As knowledge based technology has become more routinized in both the US and Japan, commercial KBS tools have been recoded in C. In the US the A.I. research community continues to use LISP as a vehicle for the rapid development of research insights; there seems to be little such use of the ICOT technology in Japan.

The PIM hardware seems destined for the same fate. The processing elements in the PIMs have cycle times no better than 60 ns; even assuming that the features which provide direct support for KL1 offer a speedup factor of 3, this leaves the uniprocessor performance lagging behind the best of today's conventional micropro-

cessors. Both HP and DEC have announced the imminent introduction of uniprocessors of between 100 and 200 MIPS. The interconnection networks in the PIMs do not seem to constitute an advance over those explored in other parallel systems. Finally, the PIMs are essentially 'integer machines'; they do not have floating point hardware. While the interconnection networks of the PIMs have reasonable performance, this performance is comparable to that of commercial parallel machines in the US such as the CM-5.

It is interesting to compare the PIMs to Thinking Machines Inc.'s CM-5; this is a massively parallel machine which is a descendant of the MIT Connection Machine project. The CM-5 is the third commercial machine in this line of development. It can support a thousand Sparc chips (and presumably other faster microprocessors as they arise) using an innovative interconnection scheme called Fat Trees. Although the Connection Machine project and ICOT started at about the same time, the CM-5 is commercially available and has found a market within which it is cost effective. One reason for this is it has quite good floating point performance. The PIMs are integer machines. It appears that the only established market for massive parallelism is in scientific computing, leaving the PIMs with a disadvantage which will be very difficult to overcome.

The leaders of ICOT were not unaware of these problems. ICOT is considering, or has begun, a project to build a KL1 system for commercially available processors. This would decouple the language from the experimental hardware and make it more generally available. This greater availability could in turn allow a greater number of researchers whose interests are in large knowledge based systems to begin to explore the use of the KL1 paradigm. Given their implementation strategy (explained above) this should not be an overwhelming task. One of the PIM hardware designers has also designed another parallel system (the AP-1000 which is mesh connected system of about 1000 SPARC chips); this might be a likely target for such an effort.

In contrast to the Connection Machine efforts (and virtually all other parallel system efforts) which have focused on massively parallel scientific computation, the ICOT effort has continued to focus on symbolic computing. In contrast to the MIT LISP Machine efforts, which didn't achieve great enough commercial viability to afford a push forward into parallelism, ICOT has has sustained long term government funding which allowed them to persevere.

6. Concluding observations

Japanese computer manufacturers (JCMs), the member-owners of ICOT, complain that ICOT efforts and funds were too much focused on specialized hardware. They complain that they now see that it would have been more pertinent and perspicacious of the MITI planners to have planned for a project involving client-server networks and UNIX software. They cite FGCS project difficulties which began with goals that were impossible to achieve and ended with squabbles over intellectual property rights. And they note finally that because of substantial project delays in the middle of the ten (now eleven) year project, not enough time was devoted to doing substantial engineering evaluations and real-world effective applications. ICOT's many excellent research contributions are drowned in a sea of complaints by the JCMs, whose work and product lines were essentially unaffected by the Fifth Generation Project.

Essentially the same sentiments were heard at the end of the Pattern Information Processing Project, Japan's first national project that incorporated AI-like goals (project of the 1970s). Where was the practical output, the companies asked. The practical output was in the methods, techniques, software, and know-how related to image processing that was deposited in the heads of a large number of Japanese engineers. This collective knowledge was used in the 1980s to enormous economic advantage in the engineering of inexpensive, reliable, and high performance devices like the Japanese fax machines, scanners, etc.

We believe that it will take several more years, perhaps a decade, for an analogous realization of the gain from the Fifth Generation national project to occur in Japan. But it will happen; and in parallel computation it is already happening.

For final words, we choose to quote from a recent personal retrospective on the Fifth Generation project [9]:

"ICOT did not create a revolution because it did not fundamentally change the manufacturers... Either another project, or a radical restructuring of the diametric cultures of education and industry, will be required to propagate the advances made in the FGCS project."

References

[1] H. Bal; Evaluation of KL1 and the inference machine, *Future Generation Comput. Syst.*, this issue.

[2] J. Barnett and K. Yamada, Evaluation of ICOT's natural laguage research, *Future Generation Comput. Syst.*, this issue.

[3] E. Feigenbaum and P. McCorduck, *The Fifth Generation, Artificial Intelligence and Japan's Computer Challenge to the World* (Addison-Wesley, Reading, MA 1983).

[4] E. Feigenbaum and H. Shrobe, The Japanese National Fifth Generation Project: Introduction, survey, and evaluation, *Future Generation Comput. Syst.*, this issue.

[5] E. Feigenbaum et al., JTEC panel on Knowledge Based Systems in Japan, Loyola College, Maryland, JTEC project, 1983, 1993 (forthcoming).

[6] ICOT, Fifth Generation Computer Sytems, ICOT Planning Report No. 1, May 1982 (note: this is a modification of MITI's original 1981 plan).

[7] S. Nishio, An evaluation of the FGCS data & knowledge base system – Expectations and Achievements, *Future Generation Comput. Syst.*, this issue.

[8] M. Stickel, Automated theorem proving research in the FGCS project: Model generation theorem provers, *Future Generation Comput. Syst.*, this issue.

[9] E. Tick, Appraisal of parallel processing research at ICOT, *Future Generation Comput. Syst.*, this issue.

[10] E. Tick, Launching the new era, *Comm. ACM* 36 (3) (1993) 90.

[11] R. van de Riet, An overview and appraisal of the Fifth Generation Computer System Project, *Future Generation Comput. Syst.*, this issue.

Edward Feigenbaum is Professor of Computer Science at the Computer Science Department, Stanford University. He is Co-Scientific Director of the Heuristic Programming Project at Stanford, a leading laboratory for work in Knowledge Engineering and Expert Systems. Dr. Feigenbaum is also Co-Principal Investigator of the national computer facility for applications of Artificial Intelligence to Medicine and Biology known as the SUMEX-AIM facility, established by NIH at Stanford University.

He has been Chairman of the Computer Science Department and Director of the Computer Center at Stanford University. He is the Past President of the American Association for Artificial Intelligence. He has served on the National Science Foundation Computer Science Advisory Board; is now serving on a DARPA advisory committee for Information Science and Technology; and has served on the National Research Council's Computer Science and Technology Board. He has been a member of the Board of Regents of the National Library of Medicine.

He is the co-editor of the encyclopedia, *The Handbook of Artificial Intelligence*, and of the early book, *Computers and Thought*, published by McGraw-Hill. He is co-author of the McGraw-Hill book, *Applications of Artificial Intelligence in Organic Chemistry: The DENDRAL Program* and was the founding editor of the McGraw-Hill Computer Science Series. He is co-author with Pamela McCorduck of the book *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*, published by Addison-Wesley (1983) and by New American Library (1984). He is also co-author with Penny Nii and Pamela McCorduck of the book, *The Rise of the Expert Company*, on corporate successes in the use of expert systems, published by Times Books in New York and Macmillan in London (1988).

He is a co-founder of three start-up firms in applied artificial intelligence, IntelliCorp, Teknowledge and Design Power Inc. and served as a member of the Board of Directors of IntelliCorp and Design Power Inc. He also was a Director of Sperry Corporation prior to its merger with Burroughs.

He was elected to the National Academy of Engineering in 1986. In the same year, he was elected to the Productivity Hall of Fame of the Republic of Singapore. He is an elected Fellow of the American Association for Artificial Intelligence and of the honorary American College of Medical Informatics. He was elected to the American Academy of Arts and Sciences in 1991. He is the first recipient of the Feigenbaum Medal, an award established in his honor by the World Congress of Expert Systems.

He received his B.S. from Carnegie Mellon University in 1956 and his Ph.D. from the same school in 1960. In 1989, he was awarded an honorary Doctor of Science degree from Aston University in England.

Howard Shrobe is a Principal Research Scientist at the MIT Artificial Intelligence Laboratory. Until January 1993, he was also Technical Director at Symbolics Inc. His work has spanned the areas of VLSI design, computer architecture, and Articial Intelligence. He received his M.S. and Ph.D. from the Artificial Intelligence Laboratory at MIT where he was a cofounder of the Programmer's Apprentice project. In 1979 he joined the staff of the MIT AI Lab as a Principal Research Scientist and in that role was one of the main designers of the Scheme-81 microprocessor (a Lisp interpreter on a chip) and of the DPL/Daedalus Integrated Circuit Design system. He also helped found the Hardware Troubleshooting project at the MIT AI Lab and is currently conducting research on Designing and Understanding Mechanisms.

At Symbolics Dr. Shrobe was one of the architects of the Ivory microprocessor and of the NS CAD system used to design it. Since that time he has led the effort to develop Joshua, an AI programming language which introduced the notion of a Protocol of Inference.

Dr. Shrobe is coauthor of the book *Interactive Programming Environments* together with David Barstow and Eric Sandewall. He also was editor of the AAAI book *Exploring Artificial Intelligence: Surveys from the National Conferences on Artificial Intelligence*.