JAPANESE COOPERATIVE R&D PROJECTS IN SOFTWARE TECHNOLOGY

Michael A. Cusumano

October 18, 1989 WP #3087-89-BPS

JAPANESE COOPERATIVE R&D PROJECTS IN SOFTWARE TECHNOLOGY

Hitachi, Toshiba, NEC, and Fujitsu all launched factory efforts between the late 1960s and late 1970s to promote internal process standardization and to diffuse good tools and techniques among in-house personnel as well as at subsidiaries and subcontractors. These companies also maintained R&D efforts in central and division laboratories, as well as in factory departments, to study or generate new capabilities, and refine process technology for application. Software development, and the needs of Japanese customers, presented similar problems to Japanese producers; not surprisingly, they tended to adopt similar solutions, creating factory organizations that refined U.S. tools and techniques while, as in other industries, seeking a tight integration among product objectives and production management, in the broad sense -- tools, techniques, controls, training, and components.

It is fitting that this working paper looks beyond the level of software factories at individual companies to consider mechanisms for taking better advantage of current knowledge as well as for moving the state of the industry and the technology forward. Despite limited results, cooperative R&D projects since the late 1960s in Japan provided a foundation for two major efforts in the 1980s sponsored by the Ministry of Trade and Industry (MITI) and carried out mainly with personnel from private firms: Sigma (Software Industrialized Generator and Maintenance Aids), which tried to make the tools, techniques, standards, and reuse concepts refined in software factories more common throughout the industry, especially at smaller software houses; and the Fifth Generation Computer Project, which experimented with logic processing and

parallel computing, areas of artificial-intelligence technology, and the required innovations in hardware and software architectures.

Comparisons with cooperative projects in the U.S. and Europe, as well as with efforts at two other Japanese producers not covered in the cases, Nippon Telegraph and Telephone (NTT) and Mitsubishi Electric, support conclusions already proposed in the factory cases: The leading Japanese computer and software manufacturers, on their own and in joint arrangements, were exploring nearly all available technologies related to software development. Major projects might not reach all their goals, and much of the technology being disseminated in Japan remained a refinement of tools and techniques promoted in the U.S. during the late 1960s and 1970s, rather than constituting a radical leap forward. Nonetheless, the Japanese demonstrated the skill and commitment to stay close to the forefront, if not in the lead, in managing the process of large-scale software development. Nor did this position come easily, as a relatively long history of failures in joint research preceded the modest achievements of projects in the late 1980s.

A MIXED HISTORY OF COOPERATIVE PROJECTS

In computer hardware, several government-sponsored projects dating back to the FONTAC effort of the early 1960s contributed to advances in the skills of Japanese firms in areas such as processor design, architectural standardization, graphics processing, and various topics in basic research. In semiconductors especially, during the late 1970s, several Japanese firms joined together under MITI sponsorship to develop better capabilities in VLSI fabrication and design. In software product and process technology, however, most cooperative efforts led to embarassing failures (Table 8.1).[1] Each attempt

floundered for slightly different reasons, although common themes emerged: poor planning, disagreements on objectives, and poor results, all affected by the difficulties of dealing with still-evolving technologies and markets.

Japan Software Company (1966-1972): MITI organized Japan's first cooperative effort in applied research not tied to hardware (as in the FONTAC) during 1966: the Japan Software Company, a joint venture of Hitachi, Fujitsu, NEC, and the Industrial Bank of Japan. The government provided a subsidy of 2 billion yen, recruited 200 software engineers, and charged them with producing a common development language that would allow firms to write basic software to operate on currently incompatible computers.

The effort to devise a common language failed completely. The architectures of Hitachi, Fujitsu, and NEC machines differed at this time, and the state of knowledge on portable computer languages remained primitive. Aside from these formidable technical hurdles, the members became distracted with different product strategies that, to a large degree, made a common language unnecessary. Hitachi and Fujitsu decided to adopt IBM-compatible architectures and thus use IBM as a standard (although Hitachi eventually varied its domestic architecture slightly). NEC, meanwhile, continued to support the incompatible architecture inherited from Honeywell. MITI dissolved the joint venture in 1972, after ending subsidies.[2]

IPA Package Project (1970-78): Despite problems with the Japan Software Company, MITI established the Information Processing Promotion Agency (IPA) in 1970 to promote the software industry in several ways. It provided billions of yen in operating expenses and loan guarantees for fledgling software producers, funds for research in software engineering, and money, as well as an

organization, to develop application packages for general use and register or buy existing packages for distribution. A major concern of the agency was to offset the growing, and labor-intensive, demand for custom programs. To alleviate this problem, the agency allocated 10 billion yen during 1970-1978 for package development and acquired 70 programs.

While software houses probably welcomed financial assistance in any form, neither the R&D work nor the package initiative proved useful. One problem was that IPA distributed funding over a large number of small firms that had insufficient expertise to develop general-purpose tools or programs. Several other factors severely limited the appeal of the packages: poor planning to insure that a program had more than one or two users, continued incompatibility in hardware architectures and operating systems, the strong preference among Japanese customers for tailored systems, and the insistence of many Japanese customers that computer manufacturers provide software free of charge. On the other hand, IPA itself survived into the 1980s and appeared to be a relatively useful agency, organizing a Software Technology Development Center in 1981 to conduct R&D in areas such as language compilers, CAD/CAM, database systems, and process methodologies and tools, and assisting in administering joint projects, including Sigma.[3]

PIPS Project (1971-1980): Another MITI and IPA initiative during the 1970s had a more positive impact on the technical capabilities of individual firms, although not process technology specifically: the Pattern Information Processing System (PIPS) project, begun in 1971 with about 22 billion yen in funding over 10 years. This focused on graphics technology needed for Japanese character (*kanji*) recognition -- an important topic because of the difficulty involved in entering and processing Japanese characters on a computer.

Project members included the major computer manufacturers as well as MITI's Electro-technical Laboratory, which integrated subsystems developed at individual firms. Although work done under the auspices of PIPS tended to merge with initiatives underway at individual firms, companies clearly utilized or built on some of the technology generated through the project funding. In particular, both Toshiba and Fujitsu introduced several products, including machines for the post office that read addresses automatically, and graphics displays with high resolutions. In addition, much of the experience IPA and member firms gained in managing and disseminating cooperative R&D, as well as some of the image-processing technology, they channelled into subsequent projects, especially Sigma and the Fifth Generation.

Software Module Project (1973-1976): MITI in 1973 started a 4-year effort known as the Software Module Project, channelling 3 billion yen in government funds to 40 independent software houses organized into five groups to develop standardized modules for applications programming. The major manufacturers did not appear to participate at all, however, and this initiative met the same fate as the IPA package project. The five groups produced little or no software that customers found appealing, reflecting poor planning and little coordination among participants regarding the content of the software developed, languages used, and portability strategies. On the other hand, this project seemed to generate interest in the concept of reusable modules as well as the need for standardization in products as well as tools and techniques. IPA quickly turned its attention to these issues and expressed them as part of a broader objective -- the software factory.[4]

Software Production Technology Project (1976-1981): MITI first directly

promoted the concept of a software factory in its very next initiative, started in 1976 -- the Software Production Technology Project (also known as the Program Productivity Development System Project). In the first stage, 17 Japanese firms, with 7.5 billion yen in funding over 5 years, came together to form the Joint System Development Corporation (JSD) and dedicate themselves to the creation of an "automated software factory system based on the concept of application software modularization." Initial objectives included the development of better languages to describe application systems and program components in a structured, modular fashion, as well as a module database. The members also hoped to build and disseminate general-purpose tools to support modular design, program generation, and testing. JSD itself built nothing but channelled R&D work to member firms through a series of projects.

Again, despite lofty goals, the project produced few concrete results. No group seemed to make progress during the first two years at all, prompting the directors of JSD to change course. Beginning in the third year, rather than working on centrally designed projects, JSD encouraged participants to use the government funding to devise support tools geared toward their specific needs, though still utilizing a base technology, such as a common programming language. Members completed approximately 20 tools, and individual firms used some in their facilities. Nevertheless, JSD failed to integrate the tools or disseminate them widely.

A major cause for this failure stemmed from technical judgments that, given changes in the technology and industry practices, proved unwise. First, especially during the initial two years, projects set out to devise tools that operated in a batch mode. By the late 1970s, however, more powerful hardware and basic-software capabilites made interactive programming and debugging through on-line terminals or work stations (rather than writing a program and

then running it later on) much more efficient and preferable for most applications. Batch-processing tools thus had limited usefulness by the end of the project.

Second, members chose to develop tools that supported programming in an uncommon language -- PL/I. They selected this because it seemed more neutral than FORTRAN (heavily tailored for engineering applications) or COBOL (dominant in business applications). In fact, PL/I combined features of FORTRAN and COBOL as well as ALGOL, an "algorithmic" language for scientific computations developed in the early 1960s designed specifically to be independent of hardware architectures and to facilitate automatic code generation -- important goals of the project.

PL/I thus had many good features and was extremely rich functionally. It even gained temporary popularity in some segments of the industry (such as basic-software groups in IBM), and Japanese mainframe producers continued to use it (or in-house variations) for operating-system development until the mid-1980s. But PL/I never became accepted as a general programming language. It proved too time-consuming to learn and difficult to use (in essence, almost requiring knowledge of all the features of three very different languages). The difficulty and lack of acceptance for PL/I in most applications limited the usefulness of the tools, most of which, in any case, supported batch processing. The project could have changed course with more flexible planning and foresight, but did not. Company teams carried out plans established in the early days of the project, so that Japan's cooperative software-factory initiative consisted merely of a few batch-processing tools for programming in PI/I.[5]

SMEF Project (1981-1986): Japanese government planners and company engineers continued to learn from past mistakes and, in 1981, launched a

follow-up initiative under the Joint System Development Corporation with 5 billion yen more in funding, called the Software Maintenance Engineering Facility (SMEF) Project. Not only did this attempt to build a UNIX-based (Berkeley version) integrated environment for maintaining and developing software in an interactive (rather than batch-processing) mode, but project members spent more time in planning, coordination, and reflection, while achieving more freedom to determine what tools to build. While SMEF constructed 10 maintenance environments and tool sets, including 8 that relied on UNIX, this project failed to produce tools considered good enough for broad dissemination. Still, Japanese companies learned a lot about the UNIX environment and support tools. In this regard, SMEF proved to be a useful preparation for Sigma (which followed directly in 1985) as well as for individual company efforts aimed at utilizing UNIX.[6]

Interoperable Database System Project (1985-1989): Once again under IPA sponsorship, the leading Japanese computer manufacturers, as well as Matsushita, Sharp, Oki, Sumitomo Electric, and several other firms, formed the Interoperable Database System Project in 1985. This 5-year program, with a budget of 1.5 billion yen, adopted the internationally recognized OSI (Open System Interface) protocols in order to promote communications or data transfers among various types of hardware (computers and peripherals, office equipment) from different manufacturers.[7]

OSI clearly represented a positive development toward standardization on a limited but important dimension. As of late 1989, most of the major computer and peripherals producers in the U.S. and Europe, including IBM, had adopted the OSI standards along with Japanese firms, and several producers worldwide had introduced products utilizing these specifications. OSI thus promised to

ease machine interconnections and simplify the building of networks, although these standards did not directly address issues such as software reusability or automation.[8]

FASET Project (1985-1989) The Joint System Development Corporation took another bold step in 1985 by launching the Formal Approach to Software Environment Technology (FASET) Project, with funding of 2.2 billion yen over 5 years, 80% provided by IPA and 20% from industry. JSD staff researchers, as well as personnel from several JSD member firms -- Case Cachexia Engineering, Software Research Associates, Kanri Kogaku, Mitsubishi Research Institute, NEC Software, Japan Information Service Company, and INTEC -- conducted the research.

As a first objective, FASET members evaluated existing specification tools and techniques prior to establishing a better methodology for generating executable code from formalized descriptions of system requirements. As a next step, they worked on creating a knowledge database of requirements or designs that developers could draw on in conjunction with tools to support specification of a software system. The last set of goals proved to be the most ambitious: to devise a practical but formalized (mathematical or algebraic) methodology for describing requirements, and then tools for optimization and transformation of the requirements into executable code (Figure 8.1). The FASET environment also assumed distributed development over a network of linked work stations and databases. The project schedule called for completion by 1989 of support tools for requirements analysis and description, database management, documentation generation, design retrieval, optimization, and maintenance. Other areas of research included tools and techniques to detect design errors, software standards, and methods for transferring specifications to different systems.

At the least, FASET brought more attention to the important goal of devising ways to generate computer programs from requirements -- thus eliminating the need to write detailed system, program, and module designs, as well as code. Executable requirements also eliminated the need to worry about reusability, although recycling specifications still seemed a useful way to reduce time required for developing new systems. Yet the FASET Project appeared to have little impact. The objective remained technically difficult to achieve except for very restricted applications, which limited the usefulness of the tools. Furthermore, the major Japanese software producers -- Fujitsu, Hitachi, NEC (except for a subsidiary), Toshiba, NTT, and Mitsubishi --did not directly participate, limiting the technical skills available to the FASET researchers. This lack of participation did not mean that the major Japanese computer producers saw no merit in FASET's agenda. To the contrary, as noted in the cases and later in this chapter, most of these companies had similar R&D efforts underway on their own, and probably saw no benefits to participating actively in this relatively small-scale project.[9]

The TRON Project (1984-1990s) Although not a software-engineering effort in the sense of developing support tools and techniques, or reusability techniques, a cooperative effort in Japan of rising attention was The Real-time Operating-System Nucleus (TRON) Project, started in 1984.[10] Unlike prior projects, TRON started as and remained an independent initiative not sponsored by the Japanese government but conceived by a professor at the University of Tokyo, Ken Sakamura. Individual firms or researchers then agreed to carry out the work needed to meet Sakamura's objective: to construct an open family of computer architectures built around a 32-bit microprocessor, with a high-performance operating system able to perform multi-tasking and real-time applications.

The unique feature of the architecture consisted of its design in well-planned layers, from the lowest level, the instruction-set processor, through the operating system and applications interfaces. Although developers had yet to complete all the components, standardized interfaces for each layer would make it possible for vendors to sell different types and sizes of computers as well as link them far more easily than most existing operating systems. The architecture would also greatly simplify communications and data transfer, as well as portability or reuse of software programs and tools. The ambitiousness of the effort can be seen in the sub-projects, which targeted embedded industrial systems (ITON), business-oriented work stations (BTRON), networking environments (CTRON), and interconnecting software objects (MTRON).

Despite MITI's support of UNIX through the Sigma Project, and TRON's origins as a private initiative, by 1988, scores of firms had joined the TRON association, including all the major Japanese computer and software manufacturers as well as foreign companies such as AT&T and IBM. The Japanese firms, led by Mitsubishi, Hitachi, Fujitsu, Matsushita, NEC, Toshiba, and NTT, had already introduced TRON VLSI processors and operating systems, as well as announced research results and concrete product plans. The Japanese Ministry of Education in 1987 provided a huge boost by adopting TRON as the operating-system standard for new computers introduced in Japanese schools, inspired in part by a useful feature of the standard TRON keyboard -- an electronic pen and tablet that made it relatively simple for users to input Japanese characters. Some TRON work stations also ran more than one operating system (such as UNIX as well as TRON), and this seemed likely to improve the diffusion of the new standard and TRON hardware.

Nevertheless, and despite the technical excellence of the TRON architecture, this project faced major obstacles in the marketplace. Computer

producers and, perhaps more importantly, computer users had enormous investments in existing hardware and software; to rewrite programs to work on TRON systems presented a daunting task few firms seemed likely to undertake without excellent reasons. In addition, most computer manufacturers were currently trying to link the interfaces among their incompatible machines, and making slow but steady progress. TRON presented a technically better but radically different solution that essentially involved discarding existing systems.

For new users, TRON offered potential benefits, although software producers still had to write programs to make the hardware and operating systems useful. On the other hand, TRON seemed likely to grow in popularity. At the least, Japanese school children would become widely exposed to TRON hardware and basic software, and this generation might disseminate the standard more widely. As the 1990s approached, however, TRON seemed most likely to remain one of many standards, probably used mainly in Japanese schools and specific real-time applications in industry where benefits were obvious and firms did not have major investments in other systems.

THE SIGMA PROJECT (1985-1990)

The Sigma Project, begun in 1985 and slated for completion in 1990, had about 25 billion yen in funding from government and private sources.[11] In terms of key personnel and goals, it represented considerable continuity with previous projects the Joint System Development Corporation sponsored, especially the Software Production Technology and the Software Maintenance Engineering Facility (SMEF) projects. Also in common with previous cooperative efforts, Sigma faced organizational hurdles and competition from still-evolving standards or technologies. Yet it was likely to affect the industry

positively because of modest goals: Sigma promised not to generate radically new tools or techniques, but to refine, standardize, and disseminate existing useful technology based on UNIX and closely resembling the tools used at Toshiba and other firms. Sigma thus supported a rising trend in that, while proprietary operating systems still dominated UNIX in market share, many Japanese firms were independently adopting this for their development environments and customers. One estimates held that UNIX would constitute about 25% of the world market for operating systems by 1990, including Japan.[12]

The center of activities for the Sigma Project, the Sigma Development Office, existed as part of the IPA structure. The staff, consisting of about 50 engineers on loan from 38 companies, took charge of planning, design, and management of the system. Private contractors building tools numbered at least 50 companies and 300 engineers. In total, as of 1989, 189 companies participated in the effort in some form. These included the major Japanese computer hardware and software manufacturers, producers of consumer electronics, and subsidiaries of U.S. computer makers operating in Japan (AT&T, Fuji Xerox, IBM Japan, NCR Japan, Nihon DEC, Nihon Sun Microsystems, Nihon Unisys, Nippon Data General, Olivetti of Japan, Yamatake Honeywell, and Yokogawa Hewlett-Packard).[13]

The director of Sigma's planning division, Noboru Akima, and a staff member, Fusatake Ooi, in stating their objectives in a 1989 article, made it clear that Sigma relied on concepts directly borrowed from previous factory efforts and other attempts to structure and automate software development: "Sigma....will industrialize the software-production process by using computerized development facilities and a nationwide communications network....The ultimate goal of the project is to produce software through manufacturing instead of

manual labor, moving the software industry from a labor-intensive to a knowledge intensive industry."[14] While Japanese companies pursued similar goals on their own, Sigma's potential contribution went beyond the individual firm: creation of a platform on which to integrate tools and hardware or reuse code from various vendors, and then make these tools and software available through an open network.

The project proceeded in two stages. In the first (1985-1987), researchers designed a prototype platform to evaluate user responses. This consisted of a hardware system, operating-system specifications, software tools, and a network to share tools, computer programs, and other information. In the second phase (1987-1990), they enhanced and finished the prototype systems. Beginning in spring 1990, the Sigma hardware platform and software were to become commercially available from several vendors. Customers also had to pay a fee to operate the system and support R&D to improve the network as needed.

The system contained three major components: the Sigma Center, the network, and the user sites, expected to number about 10,000 (Figure 8.2). The Sigma Center, located in downtown Tokyo, assisted users who were building Sigma environments and then producing software. The Center provided database services and demonstrations of tools, rather than time-sharing or remote job-entry services. The database services included information on existing software, such as applications and basic-software packages (although few were available, because of limited funds), tools, and database-management systems; software firms, such as what services companies offered; the Sigma system itself; available hardware; software standardization, such as technical articles, and Japanese as well as international practices; and other systems.

The network, the high-speed Digital Data Exchange-Pack Switching (DDX-PS) system leased from NTT, connected the Sigma Center to user sites and

external networks as well as user sites to each other. It handled Japanese characters and functions for message communication (electronic mail, bulletin boards, conferences) and file transmission (data, programs, documents), and let users access computers from remote locations, allowing them to test how a new program ran on a particular machine.

User sites came with Sigma work stations, a local area network (LAN), and the Sigma Gateway, which facilitated communication and converted protocols between the system and different hardware. The work stations were 32-bit machines with a high-resolution display, graphics support, and use of a mouse. While initial target prices came in at over $20,000, inexpensive versions appeared by 1989 for the equivalent of about $12,000. These probably would drop further since nearly a dozen firms had agreed to make the work stations and operating systems (Fujitsu, Hitachi, Matsushita, Mitsubishi, NEC, Oki, Omron, Sharp, Sumitomo Electric, Toshiba,and Yokogawa), all according to common specifications so they could run the Sigma operating system and have the same communication protocols. Users, therefore, would be able to assemble equipment of varying capabilities and deploy the machines in different ways, but still be able to communicate with other Sigma sites and the Sigma Center, as well as exchange tools and programs.

The external specifications of the Sigma operating system, designed by the Development Office staff, combined the better features of two versions of UNIX -- AT&T System V and Berkeley Software Distribution Version 4.2 -- while adding capabilities for Japanese-language processing, graphics, multiple windows, and databases. Individual manufacturers had to write the internal designs and actual code to complete their specific version of the operating system, tailored to different hardware systems.

The Sigma network provided approximately 30 support tools for each

development phase (Figure 8.3). The Sigma Development Office made and contracted for the tools, while member firms could put additional tools onto the network that met the interface specifications. Tools supported software development in COBOL as well as in FORTRAN and C, as well as assembly languages. Several tools also specifically supported engineering applications.

As of late 1989, Sigma was completing its testing phase, with more than 50 companies assisting in the evaluation of tools. There remained gaps, such as the shortage of applications packages and no good tools to support testing or designing the internal specifications of modules (which FASET was attempting). However, a few research groups were working on advanced tools for higher-level language processing and graphics-based prototyping. Future plans also called for Sigma to continue as a private company after 1990, jointly owned by the members, not only to maintain the system and the network but also to continue upgrading tools and network capabilities.

In addition to standardization of external specifications to facilitate tool portability, the open nature of the system allowed users to modify tools ordered through the Sigma office, again following the practice in Japanese software factories of tailoring tools to particular applications. To facilitate modification, when Sigma asked a vendor to develop a particular tool, rules called for the data architecture and source code that implements the tool to be available to users. The original tools ordered by Sigma become the joint property of Sigma and the vendor, and if a user desires to modify the tool, then negotiations must take place to determine royalties. Manufacturers of work stations who offered a tool might pay another firm to modify it to run on its work station or perform the modifications itself, paying only a licensing fee for the tool as determined by negotiations with the Sigma office. Firms that placed their proprietary tools on the Sigma network did not have to share source code.[15]

The objective of the tools and the overall support environment, again similar to factory approaches, was to provide support technology to reduce the need for highly skilled programmers, while still allowing users to tailor process technology, within limits that respected the network specifications and the rights of tool inventors. As Akima and Ooi asserted: "Sigma...supports all development phases and reduces the dependency of development efficiency on the skills and experience of each engineer. However, different software projects require different development environments. Furthermore, the development environment should keep growing and improving as advanced tools are introduced into the market. The Sigma system lets the development-environment designers customize the basic system to create an optimal development environment of their own."[16]

Another feature of Sigma, shown in Figure 8.2, is the integration of a software parts library onto the network. Similar to the systems at Toshiba and other Japanese software factories, the library contained documentation, program skeletons, and executable subroutines. A program-composition tool also generated source code from the design skeletons. As in the case of Toshiba and other firms emphasizing reusability, however, the value of this library depended on how well its contents matched customer requirements, and how systematic management promoted development of code for reuse and construction of new programs from reusable modules.

Individual Japanese firms were modifying some in-house tools for the Sigma network. Hitachi, for example, created a version of SDL/PAD to run on UNIX so that it would qualify as a Sigma tool. Other manufacturers, ranging from computer firms already heavily in the tool-development business to consumer-electronics manufacturers just entering the computer or peripherals markets, found Sigma attractive as a way to sell or lease tools. NTT, Hitachi,

Fujitsu, NEC, Data General, and Digital Equipment also benefitted by providing computer hardware and other systems for the Sigma Center. For many potentially useful tools already commercially available in Japan, however, Sigma provided no assistance. These included EAGLE, SEA/I, and an array of Fujitsu tools that ran on large computers incompatible with UNIX.

Sigma offered perhaps the most promise for small firms wishing to improve their level of support technology, although several issues remained unresolved for the government and tool developers. One obvious topic related to tool and code ownership, including network security. While the project set up an arbitration mechanism to settle disputes, tool modification and negotiated fees presented areas ripe for disagreements, especially if the open structure of the UNIX network allowed users to tap into databases and get access to tools and source code without the knowledge of the owners.[17]

Debates also continued in the U.S. and Europe regarding what versions of UNIX should become the international standard. This controversy positioned AT&T, which held the rights to UNIX and promoted Version V, against other firms that did not want AT&T to control the future of the system. While Japanese firms and Sigma might pursue an independent course, conflicts seemed likely over the evolution or international standardization of UNIX as well as over who owned the copyright to software based on the Sigma version of UNIX.[18]

Maintaining compatibility as well as harmony within Sigma presented another set of challenges, even putting aside the fact that member companies continued to support other versions of UNIX and many different operating systems for their individual product lines. Though they remained compatible with tools and programs on the network, Sigma in 1989 already had approximately a dozen firms creating different versions of the Sigma operating

system. Most expressed dissatisfaction with aspects of the existing standards and tried to introduce improvements, thus keeping the Sigma staff constantly concerned with standardization and compatibility.[19]

A related dilemma was how competitive and popular the Sigma work stations would prove to be with Japanese users. The Japanese were accustomed to using time-sharing terminals and proprietary work stations connected to mainframes. They also had the option of buying many other types of less expensive hardware, including proprietary work stations from the major vendors as well as NEC, IBM-compatible, and Apple personal computers, and even new machines running the TRON operating system. In software factories, managers eliminated this issue of choice by determining what hardware their developers would use. The Sigma staff, in contrast, had no control over users.

Every producer also faced a tradeoff between standardization and progress. While Sigma promised to disseminate practical tools and techniques, as in Japan's software factories, standardization also constrained technological evolution. Producers of the Sigma operating system had already voiced objections to the common specifications because they saw better ways to design the system. In addition, Sigma had no choice but to evolve within the confines of UNIX, a product of the late 1960s with a variety of limitations. More radical product and process innovations, such as represented by TRON, the Fifth Generation Project, and numerous efforts in the U.S. and Europe, would be difficult for Sigma users to assimilate.

Nor did Sigma, in contrast to individual software factories, provide users with a management structure to accompany tools and further objectives such as effective project management or reusability. In its early phases, Sigma also suffered from weaknesses similar to preceding cooperative projects. A few eager government officials, managers, and academics took the lead without

incorporating adequate input from sophisticated users and producers, such as organizations in Japan of software engineers, UNIX users, and university departments in information technology, although the project directors seem to have realized this and allowed their plans and objectives to evolve.

A final issue with regard to the value of Sigma tools and hardware remained: As in the case of SDC and other factories, tools always proved to be of limited value in software development. Individual firms and managers had to add the critical elements Sigma lacked -- such as the management and training infrastructures needed to use the tools effectively or produce reusable software and high-quality products systematically. There were no guarantees that small software houses would make the necessary investments in their organizations and people, although, based on the historical record established in other sectors, it seemed unwise to underestimate the determination and capabilities of Japanese companies in any industry.

OTHER EFFORTS: NTT AND MITSUBISHI

Japanese companies other than those discussed in the cases also had underway important efforts in software support technology, some comparable to Sigma and others attempting to move beyond this to more advanced technologies. After Hitachi, Toshiba, NEC, and Fujitsu, the two most important companies in terms of market shares and technical skills were NTT (Nippon Telegraph and Telephone), the largest firm in the world measured by the value of outstanding shares and Japan's biggest systems integrator; and Mitsubishi Electric, a diversified electrical and electronics equipment producer that made commercial and industrial computers as well as software, primarily for the Japanese market.

Nippon Telegraph and Telephone: As noted in previous chapters, NTT had for years played an important role in promoting quality control and standardization among Japanese computer manufacturers through its procurement of hardware and software for telephone and data-processing systems. Of particular significance has been DIPS (Distributed Information Processing System), begun in 1969 as Japan's domestic telephone switching network.[20] Although NTT produced some actual code in house or at newly formed subsidiaries, primarily for information systems it used internally, for much of the software it used, NTT personnel (about 6000 were involved in software development during 1989) completed only requirement specifications and functional designs, and then transferred documentation to subcontractors (including Hitachi, NEC, Fujitsu, and others) to build the actual software. NTT followed the same process with hardware, issuing designs only and contracting out for manufacturing.[21]

Channelling programing tasks to several organizations required standardization of specifications, designs, coding, and documentation, as well as an excellent mechanism for quality control to assure comparability and compatibility. NTT thus cultivated various standards and controls since the 1960s, and these had an impact on the practices of its suppliers. For example, its encouragement of the use of structured flow-charts for detailed design, , which it called Hierarchical Compact Charts (HCP), contributed to their acceptance at other Japanese firms during the 1970s.[22] NTT's rigorous quality standards also provided a model for other firms to improve their commercial operations.

As NEC, Fujitsu, and Hitachi became large-scale software producers during the 1970s and 1980s, they appeared to advance beyond NTT in technical skills and support technologies for program design and construction. However, along

with becoming a private firm in 1985, NTT introduced several initiatives that promised to improve its capabilities in software. These included the establishment of a centralized Software Development Division (SDD) and a new subsidiary, NTT Software, which adopted factory organizational concepts and technologies similar to the Sigma system, as well as an extensive R&D network, much of it devoted to software tool and methodology development.

The Software Development Division employed several hundred personnel at two main sites in Tokyo, with other groups linked through networks and on-line tools.[23] NTT had formerly dispersed these personnel throughout the Data Communications Sector and other groups developing basic software and videotext programs (Figure 8.6). Management assigned the new division four roles:

(1) *Program Production Process Standardization*: standardize the way groups designed modules, module interfaces, and functional procedures, and did coding, in order to improve reusability of software across different projects. As in the other major Japanese firms, NTT relied heavily on structured design charts and code generators.

(2) *Program Production*: implement specifications produced by industry-related or functional divisions in the Data Communications Sector utilizing a standardized process and tool set to serve as a software factory, focused on program construction rather than only design.

(3) *Enhancement and Maintenance of Debugging System*: maintain the debugging system NTT had developed for use by most software developers within the company.

(4) *Production Support System R&D*: conduct applied process R&D, including tools and techniques for automating program generating and reusability.[24]

The NTT Software Laboratories, part of the company's central R&D organization, conducted advanced R&D in language processing and design support, integrated software-development support and reuse systems, and systematization of production techniques and standards. The integrated production and reuse-support systems came in several versions, apparently for different product types, although they relied on NTT's communications network and the UNIX V operating system, as did Sigma.

For example, NTT introduced a major tool set for switching-systems software design during 1985-1988, called INSTEP. After noting a doubling of productivity over a large number of projects, NTT began moving much of its software design and in-house programming operations onto integrated support systems that provided a unified interface between the operating system and various tools.[25] Other versions of this concept included NAPSE (NTT Advanced Programming Support Environment) and SPACE (Software Production and Circulation Environment), which supported Ada, C, and COBOL.[26]

Most of the tools and techniques under development in NTT laboratories had counterparts in other Japanese firms, although NTT's researchers demonstrated particularly broad interests. Projects ranged from an Ada compiler that operated on different computers and target machines, to an automatic remote-testing system that made it possible to test switching and communications software from dispersed locations and without direct human intervention.[27] In the design area, NTT offered several promising tools: HD (HCP Design), a prototype of a CAD system, helped users write designs in HCP charts, even if they did not understand all the HCP conventions, as well as

reuse existing designs and automatically generate code.[28] SoftDA, another chart-based design system, supported reuse of designs and code as well as allowed users to execute and correct designs in a dynamic mode, among other functions. Adam, a module management system, supported reuse of flow-chart designs, code, and specifications. SDE (Software Design Environment), another experimental tool, facilitated design and maintenance of communications software, with a specialized language that described communications functions and a subsystem that automatically translated specifications into executable code. The laboratories also worked on various AI and knowledge-processing technologies.[29]

Mitsubishi Electric: Mitsubishi had only a small market share in the Japanese data-processing industry (see Table 4.1) and many of its customers came from the Mitsubishi group. Among users of its hardware, however, the company scored well in various areas related to systems and applications programming (see Chapter 1 and Appendix C). Mitsubishi also experimented with a factory organizational approach as well as conducted advanced R&D on software tools and methodologies.

Mitsubishi's main center of software development, the Computer Works, located in Kamakura, nearby Tokyo, developed hardware as well as basic software and applications programs. In late 1987, it had approximately 700 personnel in applications and basic software serving 300 systems engineers (the later remained organizationally outside the Computer Works). The structure introduced in 1987 separated systems engineering from applications development. Mitsubishi then combined applications with basic software, and systems engineering with computer sales, and operated each of these two groups as a set of independent profit centers by product (machine) line. Management did

this to make both the systems-engineering and basic-software groups more conscious of costs and profits, as well as to provide more opportunities for streamlining software production.[30]

The tools in use at the Computer Works and under development closely resembled those in the Sigma Project and other Japanese firms that utilized UNIX. Mitsubishi built an integrated UNIX work bench (also offered through Sigma) for technical or embedded engineering software, called SEWS (Software Engineering Work Station). This operated with Mitsubishi's SOLON (Software Engineers Land-On Network) for distributed development across multiple sites. The system supported NTT's HCP structured charts and automatic code generation in C and assembly language, as well as object-oriented interfaces among modules, high-speed graphics and text displays, and use of a mouse to point to objects on screens to reduce the need to input text or commands on a keyboard.[31]

Mitsubishi's Information Systems and Electronics Development Laboratory, located next to the Computer Works, also established an experimental software factory on one floor of its facility in 1985, to serve as a working area for software engineers in the laboratory and as a pilot model for an integrated software development environment. This included common facilities (review rooms, terminals), work stations linked by local-area networks, standardized development and management procedures, and a formal system for program registration and reusability promotion.[32]

Reuse promotion also resembled the approaches of other Japanese firms as well as Sigma. Mitsubishi defined frequently reusable black-box parts, consisting of general-purpose executable subroutines, and specialized white-box parts, in the form of designs or executable code and intended for specific applications. Mitsubishi departed somewhat from other firms in its use of a

separate group of "reuse engineers," for both basic and applications software, whose job consisted of reviewing existing designs and code as well as newly written software for potential reuse or modification for reuse (Figure 8.7). Other Japanese firms utilized reuse engineers probably as much or more than Mitsubishi, although none made this function quite so explicit. Other Japanese software factories tended to allocate engineering time either to building reusable packages, patterns, or subroutines, or to reviewing existing or newly written software for potential reuse, but without creating a special group for re-engineering.[33]

THE FIFTH GENERATION COMPUTER PROJECT (1982-1991)

Although publicity waned after the initial years, Japan's Fifth Generation Computer Project had already made an important impact on the world's artificial intelligence community by the late 1980s, stimulating research in Japan as well as in the U.S. and Europe.[34] Since the project architects hoped to develop a new type of hardware and software that would revolutionize the way people interacted with and used computers in the future, its broad goals made the Fifth Generation Computer Project both intriguing and unlikely to fulfill its more dramatic expectations. In addition to encouraging basic research in an important area of computer technology, the project settled on a few specific technical targets and promised limited but concrete results by the termination date in 1991. As of late 1989, however, Japan had announced no plans to continue the venture, suggesting some dissatisfaction with the results so far.

MITI initiated the project in 1982, after two years of study, as a 10-year program starting with 50 researchers (90 in 1989) and housed in the newly created Institute for New Generation Computer Technology (ICOT) near

downtown Tokyo. The schedule called for three phases: study of existing knowledge in the fields of logic processing and parallel computing, and the development of prototype hardware and software systems (1982-1984); construction of small-scale subsystems for logic processing and parallel computing (1985-1988); and completion of a full-scale prototype (1989-1991).[35] The finished computer was expected to build both inference and knowledge-based functions into the hardware, thus facilitating extremely fast processing speeds, while the basic software controlled the hardware and provided a platform for a knowledge database-management program and applications.

The term "fifth generation" referred to the evolution of circuit components, although the major difference with the new computer lay in its architecture -- how it processed data, instructions, and other information. The first four generations consisted of computers built with vacuum tubes (1950s), transistors (1960s), integrated circuits (1970s), and then very large-scale integrated (VLSI) circuits (1980s), that processed data or instructions one at a time in a sequential fashion, following the design of the mathematician John von Neumann. The fifth generation would also use VLSI chips (of the latest variety) but deploy them in a different way.

The premise of the research held that von-Neumann architectures limited the capabilities of computers and that significant progress in artificial intelligence, expert systems, knowledge processing, automatic programming, and other advanced applications required moving away from conventional algebraic, sequential instructions and data sets. ICOT worked to perfect a machine that processed information in the form of "predicate logic" statements or inferences, and did this in a parallel rather than a sequential fashion (i.e. parcelling out pieces of a program to different processors that acted on the instructions or data simultaneously and then combined the results), much as the human brain

appeared to funciton. Existing computers already used forms of parallel processing, but primarily with a small number of processors and with conventional data and instructions merely broken down into pieces. The new hardware would incorporate a thousand parallel processors and process information at a calculation speed considerably faster than existing computers.

The software specifications called for several complementary functions: problem-solving inference capabilities, to perform deductive and inductive inferences; knowledge-base management technology, to express, collect, store, and retrieve various types of information required by the inference functions; intelligent interfaces, to allow people and the computer to converse in natural languages; and intelligent programming capabilities, to enable "persons without specialized knowledge to write programs easily" (Figure 8.4). Each of these objectives constituted a modular subsystem of the basic software. Working groups, organized under several larger laboratories, conducted research in each area (Figure 8.5).

Pioneering such technology was expensive and, not surprisingly, the Fifth Generation had the largest budget among Japan's cooperative efforts in computer technology, with approximately 50 billion yen allocated over ten years, paid for entirely by the Japanese government. This amount constituted merely half MITI's original proposal, since MITI had anticipated (but did not receive) contributions from private firms. Hitachi President Katsushige Mita accepted the presidency of the venture and eight companies -- Fujitsu, Hitachi, NEC, Toshiba, Mitsubishi, Oki, Matsushita, and Sharp -- eventually agreed to send researchers to the project, at the government's expense. Companies declined to contribute financially, and only the company that agreed to build the hardware, Mitsubishi Electric, seemed to display much enthusiasm for the project.

The lack of enthusian reflected several factors. The risky and difficult

nature of the research presented a major difficulty in that it seemed to have no immediate commercial applications. Another characteristic of ICOT that probably made the effort uncomfortable for the participants was that, unlike in previous cooperative arrangements, which tended to have a small staff developing plans and then contracting work out to individual firms, the Fifth Generation called for most research to occur in common laboratories with personnel on full-time assignments. ICOT sponsored some important research at individual firms, but the structure required strict and common technical targets, and this made it difficult for any one firm to use funds to subsidize internal R&D or seize the advantage in capitalizing on research results, if any proved commercially viable.

In the initial phase during 1982-1984, ICOT researchers examined existing technologies on knowledge processing, synthesized the results, and successfully built a personal sequential inference (PSI) machine to serve as a tool and work station for research. This had approximately 100 processing elements, about one-tenth the number of the envisioned final system. They also experimented with a design for an operating system utilizing a logic language rather than a conventional computer language.

During 1985-1988, the researchers studied how to use and control groups of the PSI machines (Multi-PSI) for the actual goal -- knowledge and inference processing in a parallel mode. This required creation of a parallel hardware architecture as well as extension of an existing logic language to make it suitable for programming in a parallel fashion. The researchers also began work on basic tools and techniques for building a knowledge-processing system (KIPS) and a knowledge-base subsystem (an advanced relational database called Delta) that took advantage of the parallel architecture.

The basic software consisted primarily of parallel control functions as well

as logic or inference rules that allowed programs to act upon information stored in the relational database. To write the core software, ICOT initially chose PROLOG, a language developed in the early 1980s to support programming in mathematical logic. Writing a program in PROLOG requires constructing rules or hypotheses, as well as objectives or conditions, that a programmer wants to test, and providing data on which the rules can operate. It is a powerful language but proved difficult to learn and few programmers had experience with it.[36] In addition, the authors of PROLOG had designed it for sequential rather than parallel processing.

After some criticism and reflection, ICOT explored alternatives to PROLOG and developed an extended low-level version specifically for parallel processing called Flat Guarded Horn Clauses (FGHC). This proved suitable for specifying the interface between the hardware and the software needed to process information in parallel. Using the computer then required the development of problem-solving inference programs and knowledge-base management software. ICOT employed a system description language developed in the first stage, Extended Self-Contained PROLOG (ESP), to create object-oriented modules and subroutines (macros) for applications that could run in a sequential mode. The relational database used predicate-logic inferences (rules) to perform particular functions or carry out specific instructions, rather than processing data sequentially or simply finding and matching identical words or pieces of data.

The pilot tool for software development consisted of a sequential inference machine that used a version of sequential instruction processing, as in von-Neumann architectures, modified for parallel processing. But, as of late 1989, ICOT had made only limited progress toward building tools that assisted in intelligent programming or automating software development. Some advances came in methods for object-oriented modular programming using ESP and

parallel-programming languages, and early experimental work on a software-development consultation system for parallel programming found some interest among developers of telephone switching systems. But most tool work centered on limited goals, such as theorem proving and mathematical verification techniques, including a computer-aided proof system. Still under development were Argus, a tool for program synthesis from high-level descriptions, as well as a software knowledge-management system to support library management, program and document generation, and other functions involved in developing logic programs. Work on intelligent interface software concentrated on studies of Japanese grammar and syntax, as well as semantic and contextual analysis, fwith most of this R&D located at a subsidiary project, the Japan Electronic Dictionary Research Center.

A few groups outside ICOT proper pursued applications. One potentially useful tool consisted of a programming-support system produced at Fujitsu. This included an English-like specification language mechanically translated into predicate logic formulas, and a logic-based system to retrieve reusable software modules, stored by function, from a modules library. The library also stored specifications for each module, coded in PROLOG, which the tool compared with requirements to identify functionally equivalent modules reusable for particular parts of a new program. These capabilities resembled conventional reuse-support systems in use for several years but added superior retrieval and verification capabilities. Earlier reuse-support methods located modules by matching specifications or code, whereas the PROLOG system made it possible to identify modules with similar functions even if the specifications did not match in a conventional search process. In addition, another capability of the tool, which supported reuse as well as maintenance, was an "explanation generator." This analyzed code and produced English-like explanations of the

program logic by comparing the code with preexisting templates (skeletons) of explanations stored in a separate database.[37]

Plans for ICOT's final stage of research during 1989-1991 remained vague, in part because work had not proceeded as quickly as desired. In particular, hardware development remained one or two years behind schedule, although researchers still expected to finish the hardware and the basic software by 1991, in addition to exploring techniques for knowledge processing, natural-language processing, and a few experimental applications, such as expert systems.

Unlike Sigma, ICOT did not distribute or license technology as commercial products. Rather, the Fifth Generation Project was in the business of basic research. Individual participadnts had to transfer technology to their parent organizations and pursue commercial applications. Some companies did introduce tools that processed or utilized PROLOG, as in Fujitsu's case, although their market remained unclear. In fact, many Japanese laboratories now contained PSI machines made by Mitsubishi Electric, although few researchers outside of ICOT projects appeared to utilize them. Perhaps the major benefits of ICOT would not exceed the stimulation of basic research. In fact, the project's directors encouraged this within and outside its membership by establishing an AI Center in 1986 to monitor activities of Japanese and foreign firms in the field and by organizing annual conferences to disseminate research results and promote information sharing.

U.S. experts who examined the progress of ICOT through 1987 as part of the Japanese Technology Evaluation Program (JTECH), an effort supported by the U.S. National Science Foundation, made several observations regarding the project's objectives and achievements.[38] Most important, they concluded that the researchers had made significant progress in areas of AI such as speech and

image processing, language translation, and expert systems, at least matching U.S. efforts in these areas and "even teaching us a lesson in the speed of development and smooth industrial coupling of these commercially-directed efforts." This seemed true even though many of the results required special hardware, Japanese did not use the machines widely, and many results remained far from commercial application without much more R&D.

The evaluation team expressed a concern that the reliance of the project on logic programming, even with the invention of a new version of PROLOG for parallel processing (FGHC), presented both benefits and limitations. On the one hand, this focus gave the project clear direction and made it likely to meet basic technical targets (even if society did not quickly advance to new uses of computers). On the other hand, ICOT did not directly address promising areas of AI research, such as programming in LISP (a more common language than PROLOG that processes data and functions in the form of lists of symbolic expressions) or experimenting with neural networks (groups of many small-scale parallel processors that mimic how the human brain processes information). While Japanese companies pursued these and other technologies in their own laboratories, the Fifth Generation represented a significant effort and potential diversion from more practical technologies. The eventual value of ICOT thus depended heavily on how useful logic programming turned out to be, and this remained difficult to predict.

Nevertheless, and despite uncertainties over the future of logic programming, the U.S. experts appeared unanimous in their praise for the project's "superb software engineering design work." In the related area of super-computer hardware and software development, the panel found the hardware to be "world-class" and "the software work competitive with, if not superior to, the best quality output in the United States," even though project

planners dropped initial plans to develop new VLSI technology to go along with R&D in parallel architectures, programming, and AI.

COMPARABLE U.S. AND EUROPEAN EFFORTS

Comparisons with U.S. and European cooperative projects reinforce the conclusion that Japanese software producers and researchers were not only in the mainstream but at least close to the forefront in research on standards as well as advanced technologies related to software development. In the U.S., the most prominent example of a cooperative effort was the Microelectronics and Computer Corporation (MCC), founded in 1983 and located in Austin, Texas.[39] This R&D consortium had a staff of about 400 in 1989, an indeterminate lifespan, and a budget of $70 million per year. Membership (the shareholders) included leading U.S. producers of electronic equipment, components, and materials: 3M, Advanced Micro Devices, Bell Communications Research, Boeing, Control Data, Digital Equipment Corporation, Kodak, Harris, Hewlett-Packard, Hughes Aircraft, Lockheed, Motorola, National Semiconductor, NCR, General Electric, Rockwell, and Westinghouse. Research centered on four broad areas: (1) software technology (productivity and quality enhancement tools and methods); (2) semiconductor packaging and interconnection technologies (substrate materials, chip attachment, cooling and manufacturing methods); (3) VLSI/CAD systems (design support for very large integrated circuits); and (4) advanced computer architectures (divided among three laboratories--AI/Knowledge-Based Systems, System Technology, and Human Interface).

In software production, the specific R&D topics resembled work in Sigma, ICOT, FASET, and other Japanese projects as well as the laboratories of NEC, Toshiba, Fujitsu, Hitachi, NTT, and Mitsubishi. All were trying to create tools,

methods, and concepts to support an integrated design environment that included a full range of tools, including reuse support and automatic code generation. But its combination of theoretical studies, such as on the design process, knowledge processing, and coordination among large teams, with empirical research on projects at member companies, distinguished MCC's research.

Much of the effort in software technologies research at MCC concentrated on requirements specification, which usually required a great deal of time and expertise. Work on this theme included examining design decisions, rapid prototyping and simulation technologies, traceability of design steps, knowledge representation schemes (especially for "fuzzy" knowledge not easily expressible as, for example, 0s or 1s), and reuse of designs. The reuse work included tools incorporating expert-system techniques to analyze existing code and specifications in order to extract the underlying architecture, which could then be deposited in a database as design components for future reuse or maintenance. Other areas of research covered generic design representations, which could be compiled into different languages, as well as tool integration through platform standardization, and group coordination and management, through highly integrated and automated project-management tools and databases. These appeared especially useful for building distributed, embedded systems (software encased in hardware, with the hardware spread in more than one location) in multiple teams.[40]

As with any cooperative effort, where members were likely to have disparities in skills, objectives, and resources, MCC encountered problems. Member companies have disagreed on research agendas and thus supported different projects, with licensing rights to research results dependent upon what work each funded. This structure restricted coordination and technical sharing,

even in areas developing complementary technologies, such as VLSI and software.[41] Members were also supposed to provide many of the personnel but they did not always send their best researchers to the venture, leading MCC management to hire its own staff. In 1988, for example, only about 30% of researchers came from member firms. The drawback was that MCC researchers had to market their organization to shareholders on a continual basis, while shareholders had to make extra efforts to transfer technology back to their organizations.

The U.S. Department of Defense had a longer history of promoting research on computer hardware and software. Many of the results have benefitted the world industry -- the Multics time-sharing system, the Ada language, very-large scale integrated circuits, and various other tools and techniques, as well as basic research. In contrast to Japan and MCC, however, a common theme in defense research has been the focus of research and applications on military uses, thus limiting the total impact of cooperative ventures on the U.S. commercial sector. Nonetheless, in the 1980s, the defense department seemed to shift somewhat and exhibited more interest in basic problems in software engineering and potentially general solutions, in response to the growing complexity and expense of software for modern weaponry and other defense as well as informaiton systems.

For example, the Department of Defense in 1982 initiated STARS (Software Technology for Adaptable, Reliable Systems) as a multi-year industry, government, and university effort, with annual budgeting of around $60 million. This included the establishment of a Software Engineering Institute at Carnegie-Mellon University in 1985, where a staff of 250 researched new software tools and methods as well as evaluated factory concepts, much like the Sigma Project. In addition, the U.S. Department of Defense Advanced Research Projects Agency

(DARPA) directly sponsored several projects that overlapped with the technical themes being explored in the Fifth Generation Project and FASET, as well as Japanese corporations, besides making grants to U.S. universities for research in every major area of computer hardware and software technology.

Of particular prominence among the DARPA projects was the Strategic Computing Initiative, a $600-million, 5-year effort begun in 1983-1984. This brought together university, government, and industry researchers to study parallel architectures for symbolic computing, advanced microelectronics, and new hardware, with the objective, to an extent inspired by the Japanese Fifth Generation Project, of integrating vision, speech recognition and production, natural-language understanding, and expert systems, especially but not exclusively for military applications.[42] Compared to ICOT, however, progress in meeting research targets seemed slow, except for parallel-processing architectures.[43]

Major European electronics firms and governments had their equivalents of Sigma and the Fifth Generation, as well as the Strategic Computing Initiative and STARS. In all cases, similar to the Japanese and U.S. programs, the Europeans hoped to advance and diffuse basic knowledge in AI and other technologies, as well as make tools and methods available to a broad range of producers. In contrast to the recent Japanese initiatives, the European efforts seemed less focused, in part because the Europeans tended to fund efforts promoted by individual firms and give companies the right to commercialize the results of their R&D, rather than allowing firms to work, in effect, as subcontractors under a joint project.[44]

The European Strategic Program for Research and Development in Information Technologies (ESPRIT), begun in 1984, probably attracted the most attention in Europe, spending $1.5 billion on more than 200 projects. The

research included 47 projects devoted to software technologies -- knowledge engineering and expert systems, advanced computer architectures, and improved user-machine interfaces, similar to the Fifth Generation, as well as applied tool and methodology development, similar to Sigma. Several groups worked on method and tool integration as well as reuse technology for a software-factory environment, with an analogue to the Sigma tool set, PCTE (Portable Common Tools Environment), based on UNIX V. The main firm behind this initiative, Bull of France, offered PCTE on its work stations. Other firms followed, including GEC and ICL in the United Kingdom, Nixdorf and Siemens in Germany, Olivetti in Italy, and Sun Microsystems in the U.S.

Another cooperative program, the EUREKA (European Research Coordination Agency) Software Factory Project (ESF), worked on developing a tool set and integrated environment resembling PCTE but tailored for specific applications such as real-time software development and complex business programming. The development group consisted of Nixdorf, AEG, ICL, and several other firms in Germany, the U.K., Norway, and Sweden. Individual countries had other efforts exploring similar tools and techniques, with perhaps the largest consisting of Britain's Alvey program, modeled after the Fifth Generation in objectives but resembling ESPRIT in organization, with 2000 researchers from universities and companies working on 200 separate projects.[45]

SUMMARY AND EVALUATION

In contrast to the technology and the market, government direction and subsidies, including cooperative inter-firm projects, played a very small role in promoting the factory approach and supporting technologies in Japan. This is not to say that the Japanese government did not try to do more. Various

agencies sponsored cooperative efforts between the 1960s and early 1980s aimed at promoting tools, techniques, and concepts effectively used in factory environments. Yet none of the government-led projects seemed to have anywhere near as much impact on practice as the initiatives started and completed at individual firms. By the mid-1980s, the situation had begun to change slightly. Old and new standards still competed for acceptance, and software continued to come in many sizes and shapes -- maintaining a complex, fragmented industry of uncertain dimensions. But it seemed clearer what constituted good practice and where the key challenges in standardization or R&D remained. As a result, Japanese and other firms started to cooperate more actively and, as it seemed in the case of Sigma, more effectively.

Cooperation clearly proved necessary to further standardization. Japan especially exhibited a great need to spread good tools and techniques to the hundreds of small software houses that did programming work for larger software producers and other customers. Standardization and networks, such as with Sigma, helped make this possible. Even projects that failed to meet objectives at least familiarized companies with software-engineering concepts and tools, as well as with packages and operating systems such as UNIX. But, while Sigma appeared likely to be an effective environment for software development, firms still had to experiment with more advanced technologies, and cooperation seemed useful to complement to efforts at individual firms. FASET, TRON, and the Fifth Generation, in addition to company laboratories, provided a mechanism to explore basic technologies as well as potential applications.

In the short term, standardization around UNIX and Sigma work stations promised to help small firms raise their level of tool support. At the same time, these or other standards would probably delay the Japanese from moving to newer technologies as they appeared. TRON provided a good example, since

it offered a higher level of integration for different types of hardware and software. But while Japanese companies were introducing TRON products, particularly for industrial real-time settings and educational applications, they also maintained much larger commitments to UNIX, proprietary operating systems, or IBM-compatibility.

In parallel computing and logic processing, Japanese government officials and researchers focused their efforts and created a fascinating project for a fifth-generation computer, but bet perhaps too heavily on a narrow aspect of artificial-intelligence technology and had difficulty maintaining the interest of major Japanese firms. FASET seemed more ambitious than Sigma technically and more practical than ICOT in pioneering a critical area -- producing executable requirements -- but lacked strong participation from key companies, who had their own R&D projects on the same theme.

One might also argue that the sheer variety of activity in Japan served as much to fragment precious engineering and financial resources as it helped push forward the state of computer technology and the capabilities of individual firms. But Japanese managers appeared to recognize this, and companies tended to limit their participation in government-sponsored projects. In the long term, however, as international comparisons indicate, Japanese firms seemed well prepared for present and future competition, covering nearly all major areas of standardization, management, and research, and exploring these areas relatively thoroughly.

Table 8.1:Japanese Cooperative R&D Projects in Software Technology

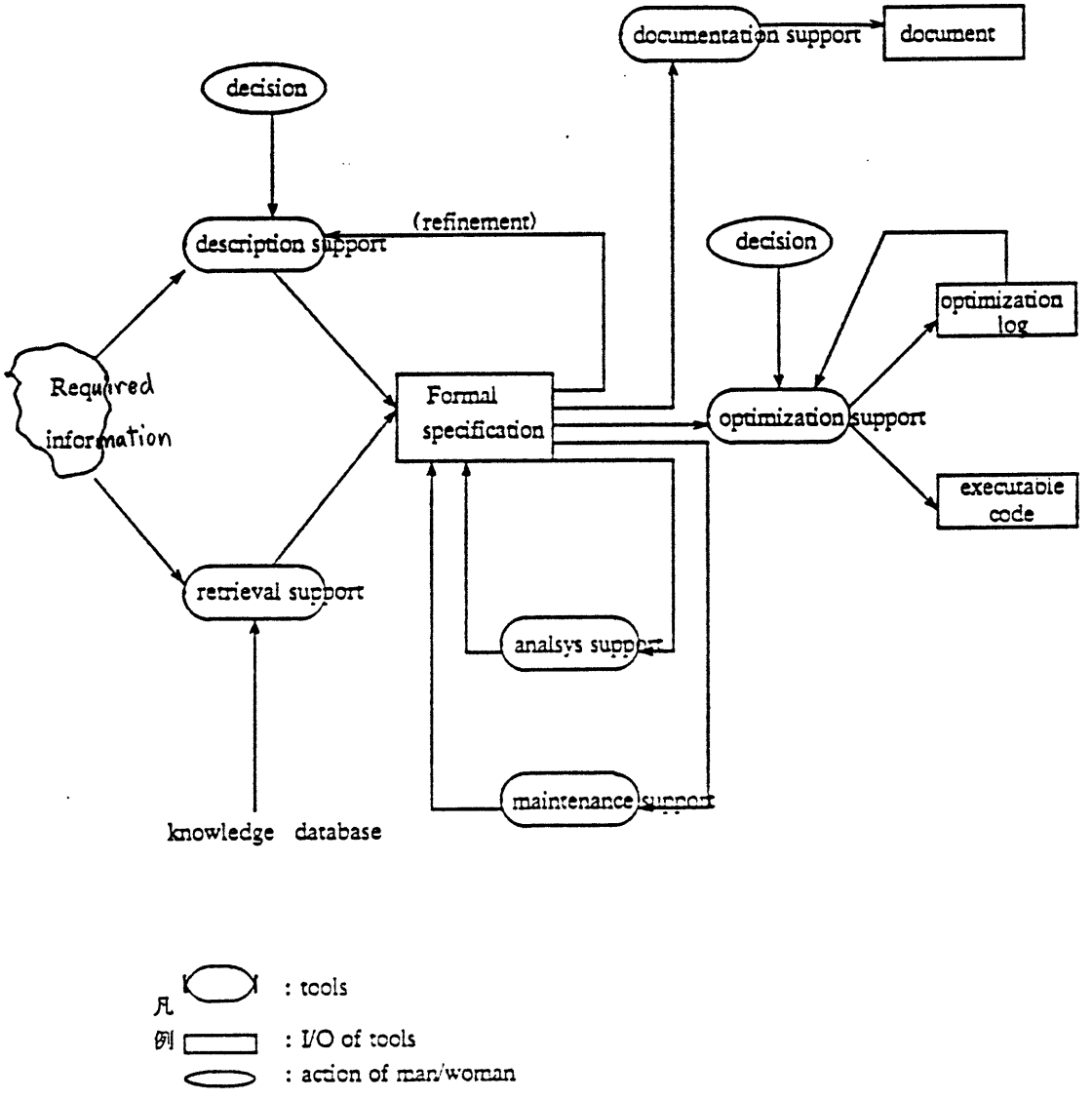
Note: In 1989 currency, 1 Billion Yen = Approximately $7 Million

Period	Project/Organization (Total Yen Funding)	Objectives and Outcomes
1966-72	Japan Software Company (2 billion)	Common development language and basic software for different architectures. Complete failure.
1970-82	IPA Package Effort (10 billion)	70 packages developed. Very limited usage.
1971-80	PIPS Project (22 billion)	Pattern-information (graphics) software, mainly for Japanese language processing. Several products commercialized. Links with Fifth Generation Project.
1973-76	Software Module Project (3 billion)	Applications development. Little coordination. Complete failure.
1976-81	Production Technology Project (7.5 billion)	Automated and integrated factory tool set and modularization techniques for batch environment. 20 discrete tools finally developed by individual firms.
1981-86	Software Maintenance Engineering Facility Project (5 billion)	Interactive, UNIX-based tool set for maintenance and development. Improved experience level of Japanese firms with UNIX.
1984-	TRON Project (Company Funds)	Development of a standardized architecture and operating system for multiple levels and types of computers. Some products announced. Promising idea despite competition from other standards.
1985-89	Interoperable Database System Project (1.5 billion)	Network to link work stations using OSI protocols. Improvement of interface standards likely.
1985-89	FASET Project (2.2 billion)	Development of CASE tools for automated code generation from formalized specifications. Promising goals but limited participation.

1985-90	Sigma Project (25 billion)	Development of UNIX-based support tools as well as reusable code and packages, for a national network. Major dissemination of existing practical technology.
1982-91	Fifth Generation Project (50 billion)	Development of knowledge (logical-inference) processing and parallel computing hardware and software. Major long-term advances possible in Japanese AI capabilities. Short-term potential for software automation and reuse support. Limited commercial applications, however, and lukewarm support from major companies.

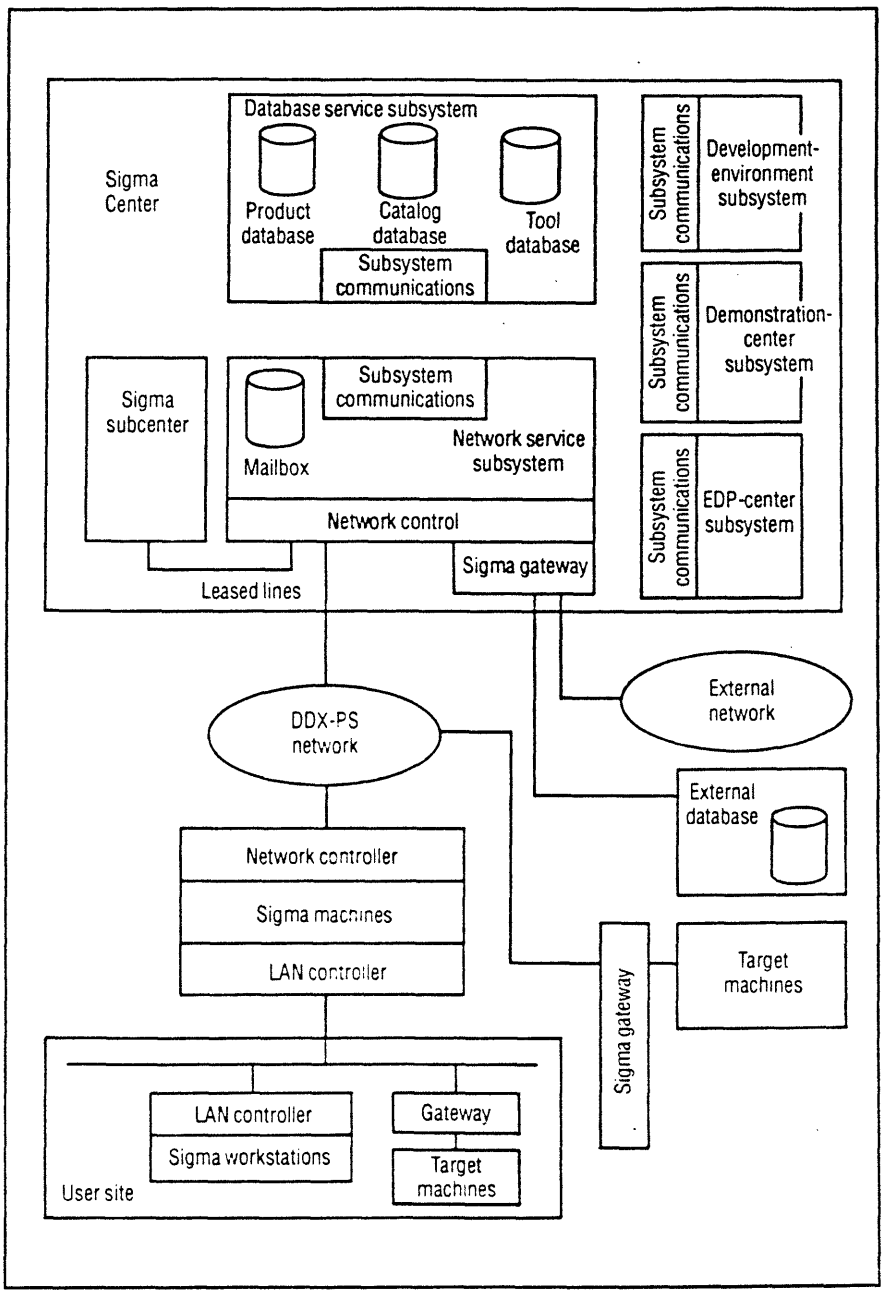
Source: See citations in Chapter 8.

Figure 8.1: FASET PROJECT DEVELOPMENT ENVIRONMENT



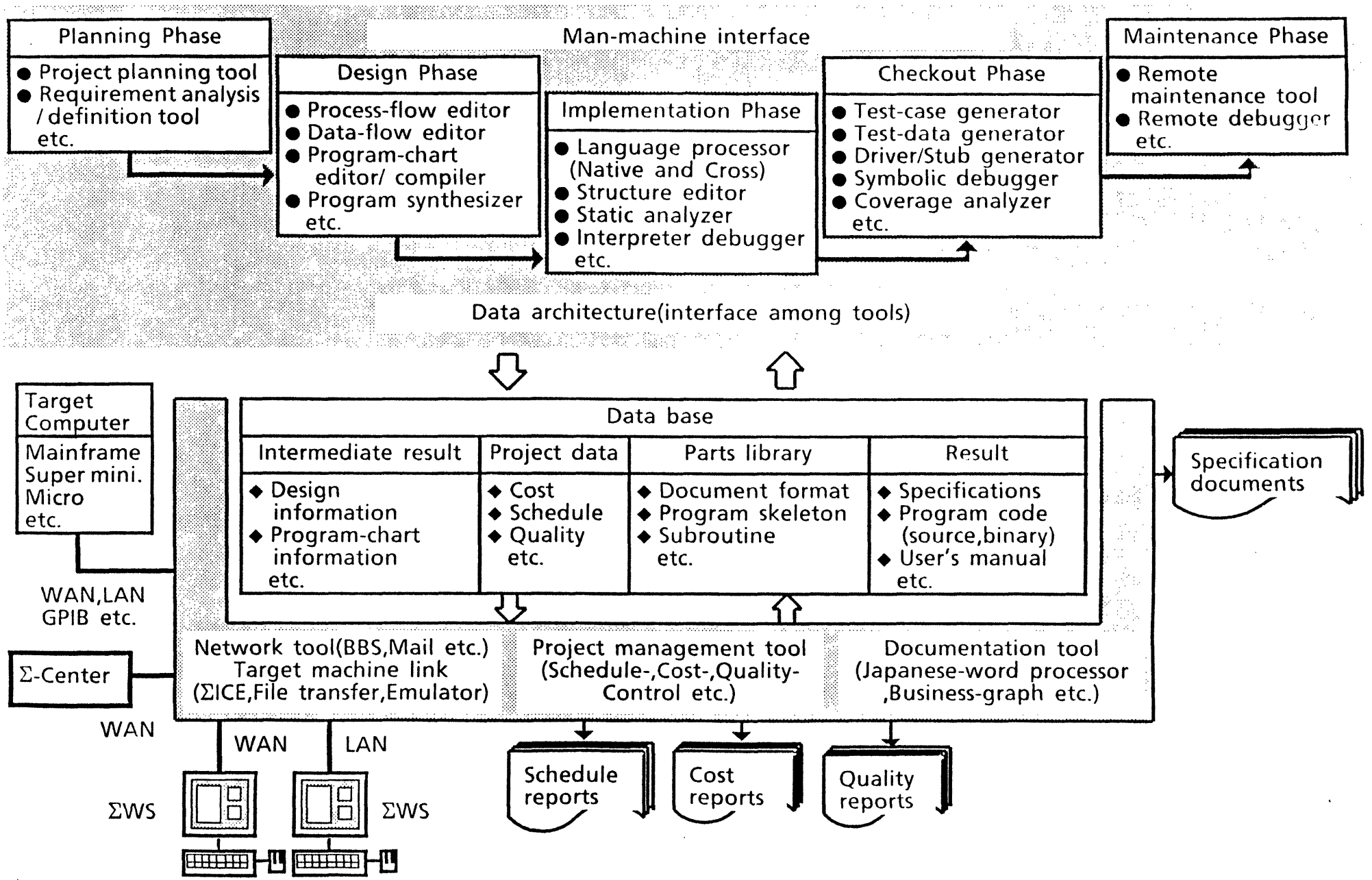
Source: "FASET -- Formal Approach to Software Environments Technology-- Overview of FASET Project," p. 3.

Figure 8.2: SIGMA SYSTEM CONFIGURATION



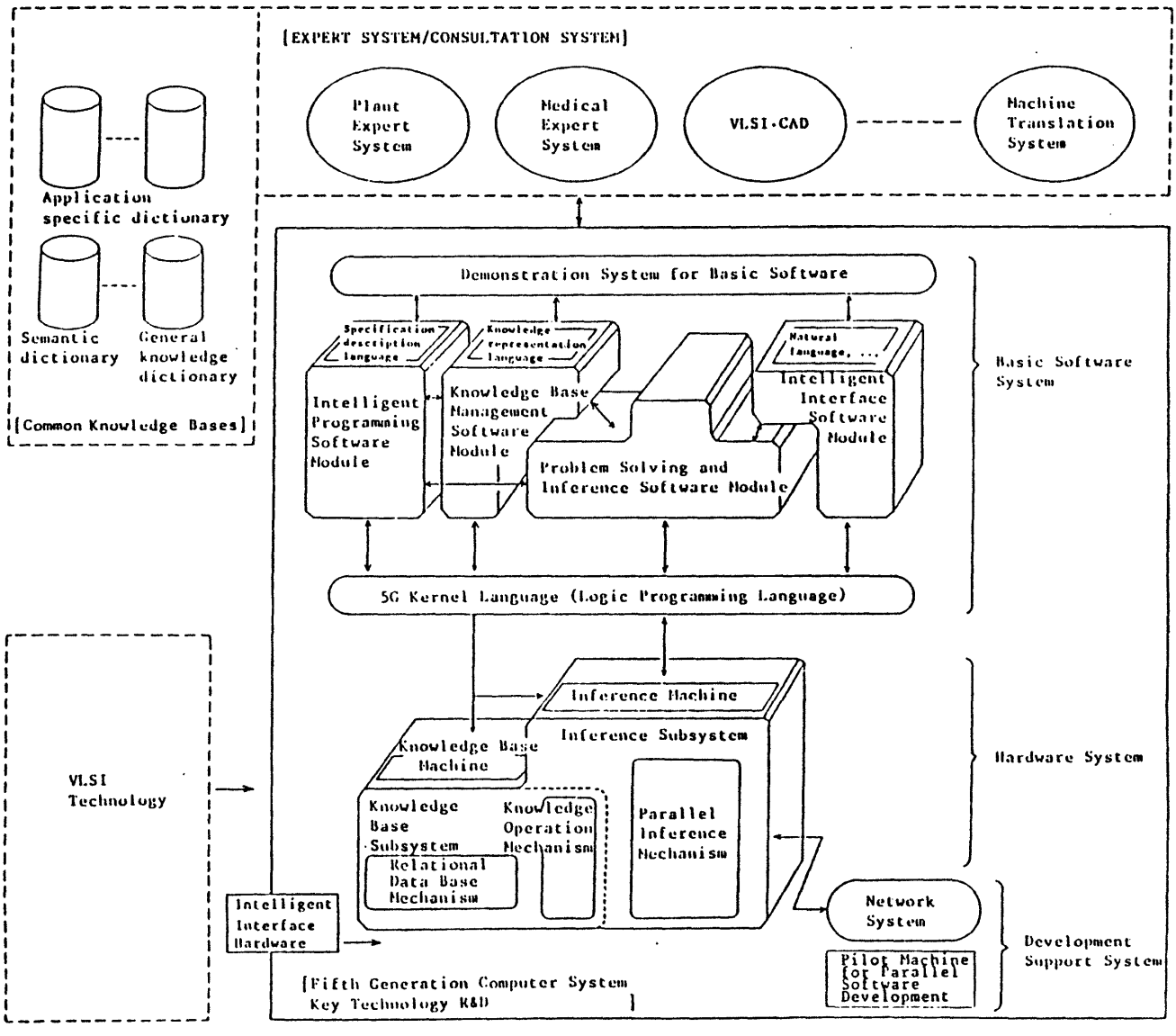
Source: Akima and Ooi, p. 15.

Figure 8.3: SIGMA SOFTWARE-DEVELOPMENT ENVIRONMENT



Source: IPA 1989, p. 18

Figure 8.4: FIFTH-GENERATION COMPUTER SYSTEM CONFIGURATION



Source: ICOT 1986, p. 24.

Figure 8.5: FIFTH-GENERATION PROJECT WORKING GROUPS

Parallel Software

Artificial Intelligence Foundations

Computer Games (Go, Shoji)

Natural Language Processing System

Japanese Generalized Phrase Structure Grammar

Speech Understanding System

Computer-Aided Proof

Term Rewriting System

Japanese Specification Language

Intelligent Programming System

Knowledge Base Machine

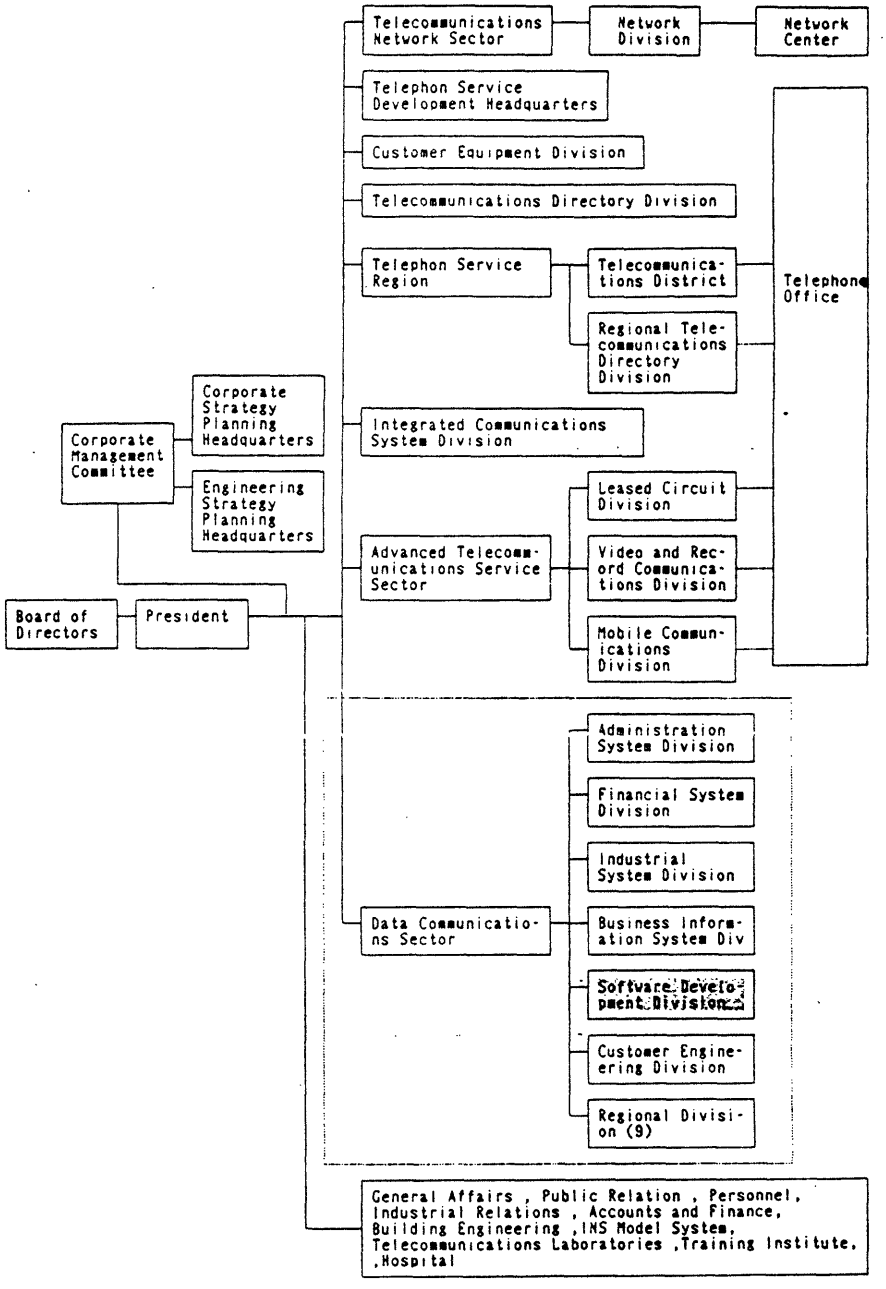
Parallel Inference Machine and Multi-PSI
(Personal Sequential Inference) Machine

Knowledge System Shell

Knowledge Acquisition Support System

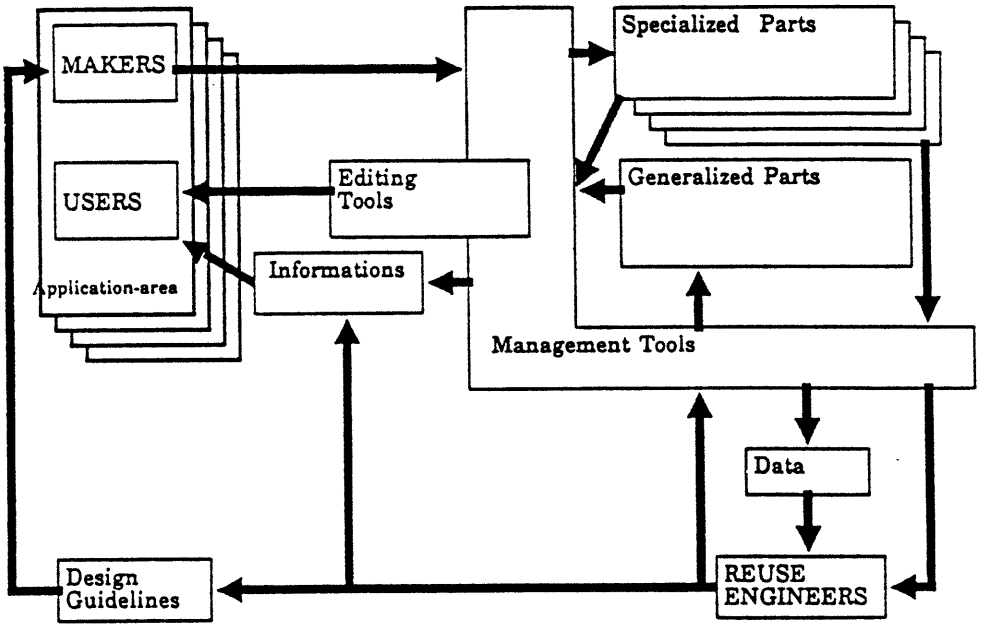
Source: ICOT 1986, p. 30.

Figure 8.6: NTT ORGANIZATIONAL CHART



Source: Nippon Telegraph and Telephone Corporation, "Software Development Division Outline," July 1986, p. 4.

Figure 8.7: MITSUBISHI'S SOFTWARE REUSE SYSTEM



Source: Mitsubishi Electric Corporation, "Software Reuse -- Our Approach," Kamakura, Computer Works, 1986.

NOTES

1. A comprehensive discussion of these various efforts, especially in hardware, is Marie Anchordoguy, **Computers Inc.: Japan's Challenge to IBM**, Cambridge, MA, Harvard University Press/Council on East Asian Studies, 1989.

2. Anchordoguy, Chapter Two.

3. Robert Arfman, "The Japanese Software Industry: A Comparative Analysis of Software Development Strategy and Technology of Selected Corporations," Cambridge, MA, Unpublished Master's Thesis, M.I.T. Management of Technology Program, May 1988, pp. 32-36.

4. Anchordoguy, Chapter 4.

5. Kouichi Kishida, "Technology Transfer Aspects of Environment Construction," Tokyo, Software Research Associates, unpublished and undated manuscript (ca. 1986), pp. 4-6. See also Anchordoguy, Chapter 4.

6. Kishida, pp. 6-9; Arfman, pp. 35-36, 60.

7. Arfman, p. 37.

8. See Joho Sabisu Sangyo Kyokai, ed., **Joho sabisu sangyo hakusho 1987** (Information services industry white paper 1987), Tokyo, 1987.

9. Thomas R. Howell et al., "Japanese Software: The Next Competitive Challenge," Arlington, VA., ADAPSO, the Computer Software and Services Industry Association, January 1989, p. 38; Joint Software Development Corporation, "FASET -- Formal Approach to Software Environments Technology -- Overview of FASET Project," unpublished outline, January 1987.

10. The best source of technical information on TRON, and the basis for the discussion in this section, is a special issue of **IEEE Micro**, April 1987, edited by Ken Sakamura. Particularly useful is the lead article, Ken Sakamura, "TRON," **IEEE Micro**, April 1987, pp. 8-14. The April 1988 issue of **IEEE Micro** also contains two articles describing further technical progress. For general discussions of the TRON Project, see Anchordoguy 1989, Chapter Five; Ken Sakamura, "Japan's New Strategy in Computer Software," **Electronic Business**, 15 November 1986, pp. 82-84; Miyoko Sakurai, "Support Swells for TRON Realtime OS Project in Japan," **Electronic Engineering Times**, 1 December 1986, p. 27.

11. This section is based primarily on Noboru Akima and Fusatake Ooi, "Industrializing Software Development: A Japanese Approach," **IEEE Software**, March 1989, pp. 13-21. Other sources include Arfman, pp. 43-61; Information and Technology Promotion Agency, **Sigma News**, Vol. 1, April 1986; and Anchordoguy, Chapter 5.

12. Arfman, pp. 60-61.

13. Information Technology Promotion Agency, "Sigma Project," Tokyo, 1989, p. 7.

14. Akima and Ooi, p. 13.

15. Interview with Fusatake Ooi, Senior Engineer and Director, Project Management Division, Sigma System Project, 7/18/89.

16. Akima and Ooi, p. 17, 19.

17. Arfman, pp. 52-58.

18. Anchordoguy, Chapter 5.

19. Ooi interview.

20. See Shimoda Hirotsugu, **Sofutouea kojo** (Software factories), Tokyo, Toyo Keizai Shimposha, 1986, pp. 50-64.

21. Interviews with Rikio Onai, Senior Manager, and Ryoichi Hosoya, Executive Manager, NTT Software Laboratories, Nippon Telegraph and Telephone Corporation, 17 July 1989; and Kenshiro Toyosaki, Department Manager, Software Division, Nippon Telegraph and Telephone Corporation, 9/3/87.

22. Mikio Aoyama et al., "Design Specification in Japan: Tree-Structured Charts," **IEEE Software**, March 1989, pp. 31-37.

23. Toyosaki interview.

24. Nippon Telegraph and Telephone Corporation, "Software Development Division Outline," July 1986.

25. Takenaka Ichiro et al., "Kokan sofutouea-yo sogo seisan shien shisutemu" (Integrated support system for switching systems software production," **Kenkyu jitsuyoka hokoku**, Vol. 36, No. 6 (1987), pp. 799-809.

26. Shimoda, pp. 198-204; Nippon Telegraph and Telephone Corporation, **NTT Software Laboratories**, Tokyo, 1989.

27. Nippon Telegraph and Telephone Corporation, "NTT Electrical Communications Laboratories," Tokyo, NTT Public Relations Group, 1987, pp. 18-19.

28. NTT Software Laboratories, "HD System," Undated document (received July 1989).

29. **NTT Software Laboratories**.

30. Interviews with Naoharu Miyakawa, Manager, Software Engineering Strategy, Engineering Department, Mitsubishi Electric; and Kenzaburo Akechi, Manager, Software Engineering Section, Production Administration Department, Computer Works, Mitsubishi Electric, 10/28/87.

31. Akira Takano et al., "A Software Development System: Solon," Kamakura, Information Systems and Electronics Development Laboratory, Mitsubishi Electric Corporation, July 1987.

32. "Tsutomu Ohkawa and Naoharu Miyakawa, "Software Development Office Environment in Mitsubishi Electric Corp.," Mitsubishi Electric Corporation, 1987.

33. Mitsubishi Electric Corporation, "Software Reuse -- Our Approach," Kamakura, Computer Works, 1986; and interviews with Miyakawa and Akechi.

34. One of the first publications to bring wide attention to the project was Edward A. Feigenbaum and Pamela McCorduck, **The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World**, New York, Signet, 1983, 1984.

35. This discussion is based primarily on the following sources: Institute for New Generation Computer Technology, "Fifth Generation Computer Systems Project," Unpublished manuscript, Tokyo, October 1986; Kazuhiro Fuchi, "Hop, Step, and Jump," **Fifth Generation Computer Systems Project: Report on ICOT's Research and Development in the Intermediate Stage**, Tokyo, Institute for New Generation Computer Technology, 1988, pp. 1-5; Takahashi Kurozumi, "Present Status and Plans for Research and Development," **Fifth Generation Computer Systems Project: Report on ICOT's Research and Development in the**

Intermediate Stage, pp. 7-19; and interviews with Dr. Koichi Furukawa, Deputy Director, Research Center, and Dr. Kazunori Ueda, Senior Researcher, Institute for New Generation Computer Technology, 19 July 1989.

36. For a non-technical but useful description of Prolog and comparisons with LISP and other languages, see John E. Savage, Susan Magidson, and Alex M. Stein, **The Mystical Machine: Issues and Ideas in Computing**, Reading, MA, Addison-Wesley, 1986, Chapter Nine.

37. Hideki Katoh, Hiroyuki Yoshiba, and Masakatsu Sugimoto, "Logic-Based Retrieval and Reuse of Software Modules," Unpublished and undated manuscript, Kawasaki, Fujitsu Ltd.

38. M. Denicoff et al., "Japanese Technology Evaluation Program: JTECH Panel Reporting on Advanced Computing in Japan," McLean, Va., Science Applications International Corporation, December 1987, pp. 3-5. The chairman of this panel, M. Denicoff, was on the board of directors of Thinking Machines Corporation, a leading AI firm. Other panel members included faculty and staff members from M.I.T., SRI International, Stanford University, and New York University.

39. This discussion of MCC is based primarily on Janet Marie Kendrick, "Managing Cooperative Research for Fifth Generation Computer Development: A Comparison of Japan's M.I.T.I. and U.S. Microelectronics and Computer Technology Corporation Projects," Cambridge, MA, Unpublished Master's Thesis, M.I.T. Management of Technology Program, May 1988. Other major sources on MCC include William H. Murphy, "The Micro-Electronics and Computer Technology Corporation," Boston, MA, Unpublished Doctoral Dissertation, Harvard Graduate School of Business Administration, May 1987; and Merton J. Peck, "Joint R&D: The Case of Microelectronics and Computer Technology

Corporation," **Research Policy**, Vol. 15, May 1986, pp. 219-231.

40. This overview of MCC efforts in software is based on Microelectronics and Computer Technology Corporation, "Software Technology Program," Technical Report # ILO-008-89, Spring 1989, Video Tapes 1 through 3. The presentations on which this discussion is based were by Les Belady (Software Technology Program -- Research Overview), Ted Biggerstaff (Overview, Information Representation-Reuse/Recovery), and Bill Curtis (Process, Methods, Tools).

41. Karen Fitzgerald and Paul Wallach, "Next-Generation Race Bogs Down," **IEEE Spectrum**, June 1987, p. 32.

42. Kenneth Flamm, **Targeting the Computer: Government Support and International Competition**, Washington, D.C., Brookings, 1987, pp. 72-75; Feigenbaum and McCorduck, pp. 91-92, 271-276.

43. Fitzgerald and Wallach, p. 31.

44. This discussion is based on Gregory Michael Toole, "ESPRIT and European Software Capability: An Analysis of Cooperation in Software Technology R&D," Cambridge, MA, Unpublished Master's Thesis, M.I.T. Sloan School of Management, May 1989.

45. Fitzgerald and Wallach, p. 33.