INVITED PAPER *IEICE/IEEE Joint Special Issue on Autonomous Decentralized Systems*

A Study of the Service Industry—Functions, Features and Control

Chitoor V. RAMAMOORTHY†, ***Nonmember***

SUMMARY We study the evolution and the dominance of the service based functions and their distinguishing features. As the service industry matures, its functions bear many similarities with the software development processes, such as intense man-machine interaction, knowledge intensive activities, flexibility in the organization, control and execution of tasks. In this paper we discuss wide range of interconnected topics, emphasizing the multi-faceted nature of service functions. These include the evolution of service industry and their products, the consumerization of high tech products based on their large-scale adoption and the consequent creation of implicit requirements; the technology transfer processes; the error proneness due to intense and prolonged interaction with computers and some methods of mitigating error incidence. We argue that by proper 'humanization and personalization' of interactive systems and by the use of teams of computer supported professionals, we can prevent such errors. We discuss some useful team types, models of their behavior and their control aspects. As the cost of communications shrinks like due to the Internet, we conclude that a fully decentralized system control provides a flat, flexible, and fair and friction-free organization for large team based service systems.

key words: *service engineering, service industry, software engineering*

1. Introduction

The service industry, whether it is travel, leisure, entertainment or finance, involves personalized activities requiring interaction and intervention between humans and machines. The U.S. Bureau of Labor Statistics estimates that more than two thirds of all workers in U.S. are involved in service functions at present and their numbers are increasing rapidly (Fig. 1(a)).

The dictionary defines 'service' as an organized system of appliances, products, personnel and other resources to supply activities needed to satisfy a public or private need. Insurance, banking, catering, lodging, travel and entertainment activities are traditionally considered as service industries. TV repair service, product maintenance services, and even massage service in recent times are listed as service functions. However, the goals and the meaning of the service functions have broadened. In modern parlance, service is the work performed directly or indirectly to satisfy the needs required by customers. Service employees often supported by such resources as telephones, messaging devices, TV, computers, etc. that help support their interaction with customers. The service activities are generally initiated by customers and generally controlled by them.

Since the industrial revolution in Europe manually intensive functions have given way to mentally or intellectually intensive activities. As a part of this transformation, we have seen an increasing dependency on the high technology, actually information technology, mostly on PC's, and networks. The manual chores have reduced and are replaced by automated devices and robots. Software agents and tools support the mental activities.

Service functions are delivered or discharged by service providers at the request of customers. Service providers who cater to customer needs and requests satisfy the preferences, needs and conveniences sought by customers. Services must be easy to call on, responsive and dependable. They must be competitive in costs, easy and reliable to use. Service offerings must be up to date. They must pour out new products, keeping up with customer's insatiable frenzy for functional novelty and technological curiosity. The service providers individualize/customize the service packages and products to suit their customers, collectively, selectively and individually, where possible.

The service providers often may work at home like their clients, reminiscent of the ancient craftsman of pre-industrial Europe, who practiced his craft from home. Service providers generally work in small teams—some co-located and some distributed in desperate places (virtual teams) but all work collaboratively or cooperatively as members of a single team. 'Small service firms with fewer than 100 workers account for one half of the American workforce and are responsible for 2 out of 3 new jobs. These service firms seldom have more than 3 layers of hierarchy—The workers are provided with more variety and responsibility (through self governing teams and the like).' (The Economist, Jan. 29, 00)

Service functions are major components of every industry. Examples of service function-dominated industries include finance, travel, telephone and communications, power and other utilities, health care, entertainment etc.

These mentally (knowledge) intensive activities are shown to be error prone if the operator or user is stressed emotionally or physically. Studies by Andersen

Manuscript received January 18, 2000.

†The author is with Computer Science Division, University of California, Berkeley, CA, USA.

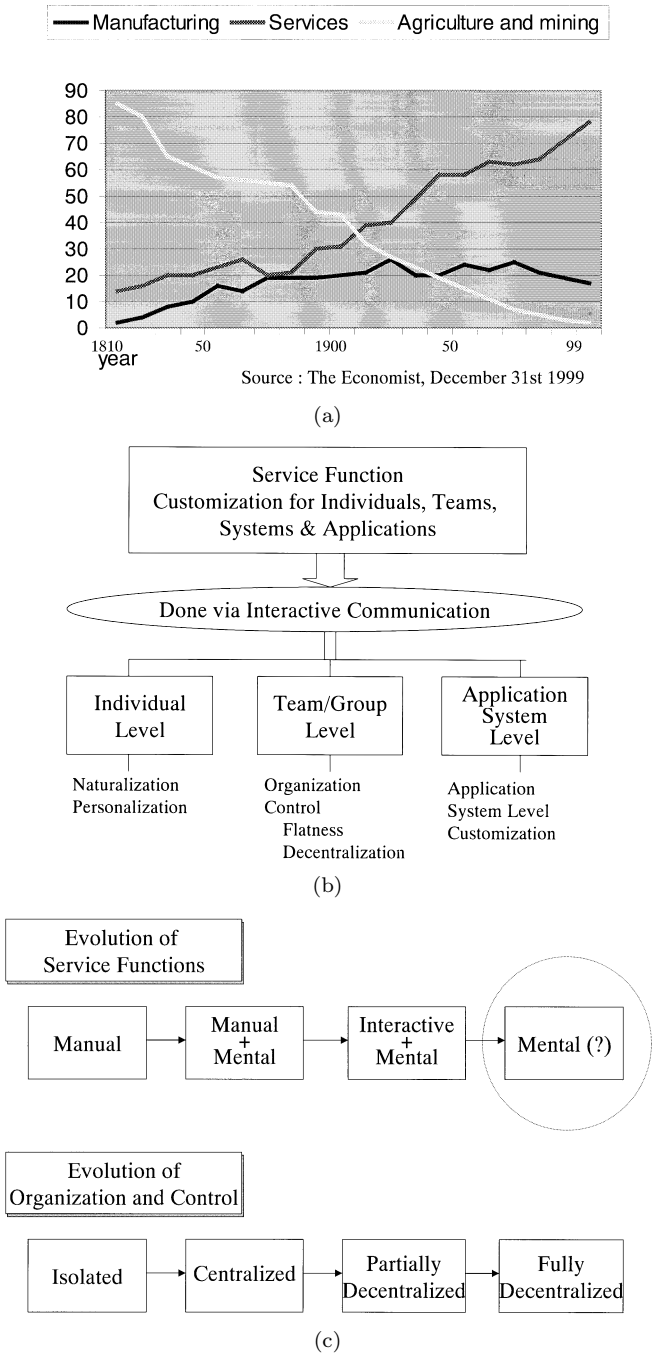


Fig. 1 (a) The growth of service industry functions in US. (b) Service function. (c) Evolution of service functions and service organizations.

Consulting and others have shown that about 40–80% of the failures may be due to improper human reaction due to unnatural circumstances, including lack of knowledge, stress and confusion created by information overload and abundance of useless information (commonly known as data smog).

Service functions aim to satisfy and facilitate the goals of their customers. In doing so, service providers design their service functions (also called, service tasks) so that the customer feels comfortable and convenient in using them. The service providers try to precondition, specialize or personalize their offerings to satisfy the customer's needs, desires, preferences and fancies. This personalization or preconditioning is based on or derived from user's requests, perceived needs and goals primarily through interactions. The interactions are between humans (with their support resources like computers, networks, etc.), between teams of humans and the application environment. Customers and users communicate their service needs through their interactions. Preconditioning of service functions may also be based on other aspects such as community, environment etc., but we shall consider not them in this discussion.

The dictionary defines interaction as 'a two way communication between source of information and a user who can initiate or respond to queries.' Every service function has a 'convenience' dimension, i.e., adaptability to help users in facilitating towards their goals, but also be a silent monitor who observes the user's action so that it could serve the user better in the next sequence of tasks. Therefore preconditioning is based on ongoing dialog (interactive session) between the user and the servicing system.

We classify the interactions into three categories: 1) interactions between service functions and individuals. 2) interactions between service functions and teams and 3) interactions between service functions and customer applications.

By personalizing and humanizing our service products we enhance users' convenience and reduce the incidence of errors and thereby improve their productivity and quality of their work.

In the case of interactions between and amongst teams, we can show that decentralized form of organization and control facilitates the effectiveness of teams. This is somewhat similar to personalization of service functions and service products for the individual customer. Decentralization of teams provides a flat organization (or something close to it, like the 3-layers of hierarchy as quoted from The Economist above) with the assumption that as communications costs become negligible there would be lesser need for rigorous hierarchical or centralized control. Therefore there would be more encouragement for individual initiatives and exchange of ideas during decision making and all of which facilitating sharing, learning and creation of knowledge.

The third type of interaction is between the service function and the application objectives. Here the service products are preconditioned to provide high performance, reliability and safety to that specific spectrum of applications. We shall discuss these in the subsequent sections.

The technology evolution has seen manually intensive tasks being replaced by a combination of mentally and manually intensive tasks which in turn being re-

placed by a combination of mental and interactive tasks Fig. 1(c). Currently, some of the services are possibly performed by individuals at their homes. Figure 1(c) also depicts the evolution of team organization and their control. The service system moves from an isolated cluster of non interacting entities, towards a centralized, then to a partially decentralized and finally to a fully decentralized fully connected entity. When the last stage of evolution is reached, the system is organized as a flat (one layer) organization where each component or entity may act by itself in full agreement with other members of the system by mutual consultation or rules of behavior protocols. We shall defer our discussion to later sections.

In this paper, we study, very briefly, the evolution of the service industry and the role played by automation We shall also touch on the product improvements by customization, humanization and personalization that support service functions.. We shall also comment on the process of technology transfer, and the evolution of information technology products that support the service functions. We show that this evolution has created a new set of requirements, called the implicit requirements.

Since many service activities involve teams of people and machines, we shall study some significant trends in the organization and control of service teams. Lastly we develop some models for team-oriented systems and comment on their usefulness. We shall show that decentralization is the growing trend of team-based service systems of the future.

2. Distinguishing Features of Service Functions

The distinguishing characteristics of service functions include the following:

Human (customer) needs driven
Knowledge intensive, high mental effort
Automation intensive to reduce manual effort
Human interaction intensive
Information technology intensive and
Team based.

We shall next comment on some of these.

Since they cater to human needs, the service functions are human-machine interaction intensive, knowledge intensive or manual.. They are mostly information (software) technology driven. Teams execute most of the functions. Teams could include clusters of human professionals and computer resources.

There exist several parallels between the service industry and the software industry. They both are highly knowledge oriented and are heavily dependent on collaborative efforts between humans and their networked computers. Collaboration is needed because the size, complexity and intellectual intensity of the service tasks. Just like software development operations, the service operations tend to be flexible, easily modifiable and requiring to variable lengths times for completion. Also, the modern day service functions are focussed on facilitating the goals their clients and these goals can change as the services are being rendered. By contrast, some service functions may involve manually intensive operations, whereas the software functions do not.

2.1 Human Needs Driven

The human needs can be of three types:

Personal—humanistic and individual.
Professional
Societal (community)

One major personal need is to make the machines react more like humans so that the operator or user is always comfortable, and not stressed, surprised or confused at any time particularly in an emergency. We call this the 'humanization' process. Individualized personal needs may depend on our disabilities, such as eye and hearing impairments, left handedness, short attention spans etc. They may also depend on our personal preferences and styles. We call this 'individualization.'

Professional needs of service functions call for excelling in the job, and improving one's productivity and performance at work. This generally requires customizing work equipment and the work place to fit the application and the employee so that the performance of the system and the productivity of the professional are improved.

Societal needs of the service functions emanate from community and environmental consciousness and responsibilities, thereby eschewing altruism—befitting the theme of 'The Three Musketeers,' namely, 'all for one and one for all.'

2.2 Knowledge-Technology Intensive

Service activities generally involve choosing among options, and solving problems with the help of computers, technology and knowledge. By knowledge, we mean primarily technology oriented scientific knowledge. Davis and Boskin [1] have conjectured that scientific knowledge grows exponentially, doubling every seven years. It raises the technological and educational levels with its growth. By Ross Ashby's law [2], this has the rising tide effect; that is, the rising tide (of knowledge) lifts all the boats (improvements in functionality and quality of service). Specifically, as customers become more educated and familiar with new technologies, they demand better, faster and more intelligent services and products. Instead of walking or requesting a rickshaw to take her from San Francisco to Los Angeles, the customer likes to fly in a jet plane at the cheapest fare.

Knowledge growth implies a parallel growth in technology from which it is derived. This is evident from the semiconductor (primarily the microchip) technology's exponential growth as postulated by the Moore's law, which stipulates that chip performance doubles every eighteen months while the cost remains the same.

The streams from the knowledge generation process flow across the knowledge-technology-services-products chain, or the technology transfer chain—all the way from knowledge creators, technology developers, service providers, product developers, to fat, thin, and famishing clients and users—as products and services.

The driving forces in technology belong to three categories namely, the current technology, the supportive and integratable (collaborative) technologies, and the so-called disruptive technologies. We follow similar arguments like those of Clayton Christensen ('The Innovator's Dilemma. When New Technologies Cause Great Firms to Fail,' Harvard Business School Press, 1997). As the services and their related products grow like in the semiconductor industry, the supportive and integratable technologies will help to sustain or boost the current technology by enhancing its native performance and capabilities. Some examples include the copper wire technology, which has enhanced the semiconductor microchip performance, and the laser technology, which supports the established semiconductor technology by providing important computer adjuncts like printers and fiber optic networking. Thus the supportive and integratable technologies lengthen the life (strangle hold?) of the current technology beyond its physical limits because a changeover could mean another Y2K problem, (or will it be a fizzle or a Chernobyl?—that will be the question!) but more massive disaster. A disruptive technology such as the one based on superconductivity or quantum physics may have a devastating effect on the current technology. An example from the olden times is the uprooting of the vacuum tube and the magnetic core memory technologies by the invention of the transistor, which heralded the current semiconductor industry.

2.3 Human Interaction Intensive

There exist many types of human interactions; viz., amongst humans, between humans and machines, and within and across teams of humans and machines. We will be concerned with two primary types of interactions, one of which is knowledge intensive and the other, heavily oriented towards simple keyboard interactions and bookkeeping. Knowledge intensive interactions are those in which humans have to think and interact with the computer, like when one composes a report or a letter on the computer. We shall simply label this as a knowledge intensive (K) activity since mental activity is primary and machine interaction is secondary. On the other hand, when someone does pure transcription like copying a letter onto a terminal for transmission, this is considered as a simple machine interaction, or simply an interactive bookkeeping operation (IB).

2.4 Information Technology Intensive

The service functions depend on very heavily on information (computer and communications) technology (IT) with the software discipline as the major player. Every aspect of the service function from customer requests to service delivery, and service evaluation by customer satisfaction-metrics is IT driven. There are striking similarities between software development processes and service functions. These include their knowledge dependency, their collaborative development and the ease and flexibility in modification and execution. Just like the software activities, the service functions show great variability of interactions, namely, human-human, human-machine, team-team etc. The two disciplines depend on decisions and choices made by users and developers but always facilitating the goals the clients would want to achieve.

2.5 Automation Intensive

Automation is the replacement (or substantial reduction) of manual, mental and human interaction efforts by automated systems, computer and knowledge based mechanisms. Automation includes the use of intelligent robots to tend the manual service tasks, and intelligent software agents to take care of intellectual or mental functions. As we shall see later, the evolution of service industry and its functions parallels the growth of the automation technologies. Automation has produced in its wake momentous improvements in productivity, performance, and product and service quality. But most significantly, automation has been very agile and resilient in accepting new ideas and new technologies rapidly—thus accelerating the creation of new product families and facilitating their use.

Manual (physiological) effort is tiresome and accident-prone. Automation reduces manual effort and thereby helps prevent human induced failures. Similarly, computers, knowledge-based software tools and decision support systems relieve intense mental (psychological) effort and mitigate error incidence in mentally intensive activities.

2.6 Teams and Teaming

Teams of people supported by machines who work collaboratively or cooperatively perform service functions. In serial chains, or assembly lines, teams work on a task in a sequential fashion. Teams can also be structured to perform parallel operations functions in the form of

parallel threads. We shall discuss about teams in a later section.

3. Service Functions

Service functions are embedded in all industrial, business and economic functions, and indeed, in every human endeavor. Included amongst them would be the administrative services, and bookkeeping activities that are prominent in business and industrial enterprises, and the maintenance services that are primarily associated with complex manufactured products. We can classify the activities into four principal categories. The first category is mostly manual (M). The second is mostly man-machine interactive but requiring no intensive mental effort. We include in this the bookkeeping and administrative activities which generally require simple interactions with computers. We shall label them as IB (Interaction-Bookkeeping). The third category is mostly mental or knowledge intensive activity (K) which includes human to human discourses, man-machine interactions requiring intellectual effort like playing chess with the computer or composing a letter on the computer terminal. Human thinking and decision making are central in this type of activity. Examples include planning, searching, selection and finding closest matches to the desired routing, and developing cost effective schedules subject to customer's desires The fourth type of activity is associated with networked transactions (NT). This would involve accessing websites, databases and performing data-processing functions as well as Internet and web-based activities. The interaction based activities (IB), the knowledge based activities (K) and manual activity (M) take variable lengths of time for execution. The execution times for network-transaction (NT) type of activities will depend on the vagaries of database processing times, delays due to network congestion and transmission. But fortunately, they can be estimated reasonably well. Bookkeeping and administrative operations or the IB type of activities can be done in parallel with other types of service functions with which they are generally overlapped and interleaved.

We shall illustrate the service activities by means of a simple example. The example does not include any manual operations. A customer decides to take a trip and interacts with his travel agent to discover the best choices for the journey. The customer provides a list of specifications and constraints such as the places he likes to visit, preferred travel dates, his budget etc. This is a knowledge intensive (K) activity. We shall call this Step 1. The travel agent consults an air line reservation system to provide a list of feasible options. The data entry part is an interaction-bookkeeping (IB) operation. This is Step 2. Next, the travel agent accesses the airline reservation system and she receives a list of feasible travel choices. This (Step 3) is a networked transaction-based (NT) operation Then the customer makes his choice. This is a knowledge intensive (K) operation. This is Step 4. The agent enters the customer's choice into the networked airline reservation system. This is Step 5 and is a data entry (interaction-bookkeeping, IB) operation. The reservation system updates its records and confirms the customer's itinerary. This is Step 6 and is a network transaction (NT). The customer then pays for the fare using his credit card. The travel agent enters the payment information into the system. This is Step 7 and is an IB operation. The reservation system enters this into its file. This is Step 8 and is a network transaction (NT). The agent confirms the itinerary and prints it out. This is Step 9 and is an interaction-bookkeeping (IB) activity. We have broken down this service transaction into a number of simple steps.

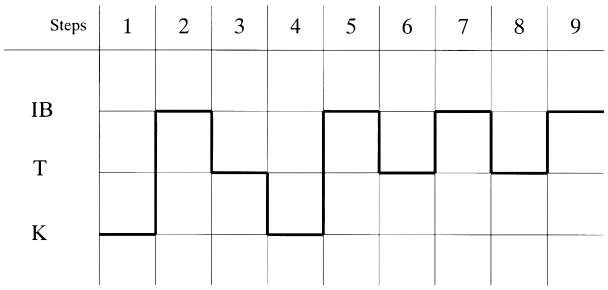


Fig. 2 The sequence of service operations in air line reservation example.

We shall illustrate the sequence of activities by a simple Gant-chart like sequence diagram (Fig. 2).

As an example of service using the Internet-Web domain, consider a typical transaction of the on-line bookseller, amazon.com. When the company accepts a book purchase order, it performs several interactive-bookkeeping (IB) and Network-Transaction (NT) operations. We shall discuss only a few. It performs a network transaction (NT) on the customer's credit card account, transmits the book dispatch order (NT) to the warehouse. It also requests UPS to do the pick-up and delivery to the customer's address. This is an NT type activity. The delivery of the book by UPS is a mostly manual activity (M). We will not go into details of this function. We can represent all the service components in this 'e-tailing' transaction using the activity elements we discussed above. In parallel, the company may also record customer's book preferences, tastes for topics and authors for its future business. It may also monitor the performance of their service personnel. These are all parallel bookkeeping (IB-type) operations.

4. Evolution of Service Industry

The Austrian born American economist, Schumpeter traced the growth and the impact of technologies over the last two centuries. (Figs. 3, 4). The figures show the waves of innovation of dominant technologies over the

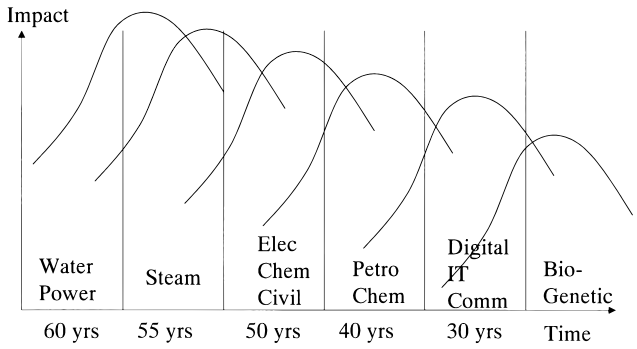


Fig. 3 The growth of new technologies (Europe).

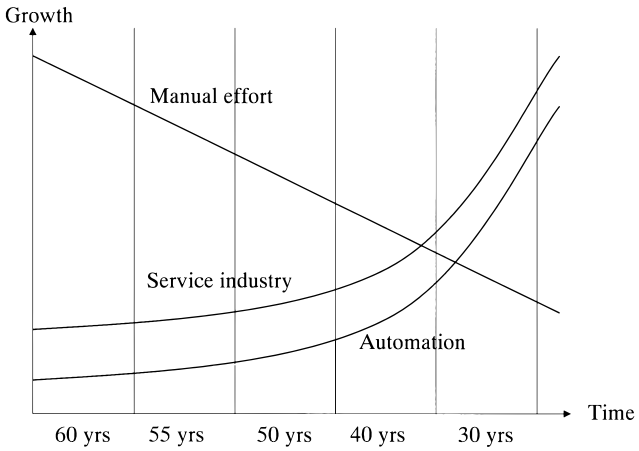


Fig. 4 The growth of service industry and automation.

last two centuries. We have included in it the information technology industry, encompassing the computer, communication and artificial intelligence technologies. Each technology growth wave rises out fast, matures, and then declines, just when the next technology innovation wave starts to move up and takes hold. The interesting fact is each technology spawns new industries less and less manually intensive than its predecessors and accepts more automation as it matures and therefore requires less manual labor than it started out with. This also implies a tremendous growth in productivity. Figure 4 shows the marked decline of manual effort and the growth of the service oriented functions and the growth of automation as a function of time. One can attribute the decline of manually intensive functions to the surge of automation. Service industries seem to have benefited by the advances in the information technology, which is, by far, the most important enabler of manual and mental automation. Economies of U.S., Japan, Singapore, and Western Europe seem to fit well with the Schumpeter's model.

The rapid growth of the service component of the industries implies an increase in activities largely dependent on manual, and mental interactions with machines. This, unfortunately, may provoke human generated errors. Product design and development is complex and mentally intensive activities partly because it is cross disciplinary. Products and product families have to be periodically enhanced by adding new features and upgrading the technology to meet the challenges of competition. At the same time, the design and the operational life of the product must be lengthened and the costs must be kept low. These activities, particularly in the software area, come under the banner of maintenance. These could be horrendously detailed and mentally taxing, therefore become bug ridden and veritable sources of future problems.

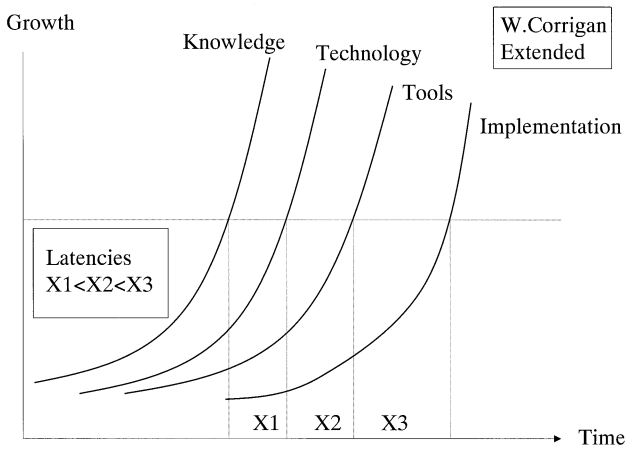


Fig. 5 Knowledge technology transfer (e.g., semiconductor industry).

5. Knowledge Utilization and Technology Transfer

The delivery of services depends upon products, procedures and human interactions. Service products are produced by technology. Human needs, knowledge and experience create technology. Knowledge, in our context, implies a specific aspect of the discipline that is needed to satisfy a product technology. The ultimate aim of technology innovation is new product creation to satisfy some human need. Knowledge utilization and technology transfer is an important part of this process, which we consider as a very important service function that supports human progress. It consists of the following four broad phases: a) knowledge generation, b) technology derivation to satisfy a broad array of human needs and c) new product family specification, and the development of design methodologies, and the related computer-aided tool development and d) product implementation, and manufacturing. For the sake of brevity, we have omitted important steps like testing, product quality control, marketing etc. Figure 5 illustrates the essential phases of this process and displays the processes associated with the growth of knowledge, technology, tool development, and implementation, over time. We have extended the graphical depiction of Dr. Wilfred Corrigan, CEO of LSI Logic,

and we have augmented it by adding the knowledge generation (creation) phase and introducing the concept of phase latency. We define latency as the time difference between two successive phases during a specific instance of growth of a technology (Fig. 5). For example, the latency between knowledge creation and the corresponding technology development at some specific instance of growth is x1 (Fig. 5). Latencies, in practice, cannot be measured precisely, but only estimated based on experience. As shown in the Figs. 5 and 6, the latencies in the semiconductor industry between successive phases seem to follow a transitive relationship, which is x1 < x2 < x3. To phrase this in words, the time to transfer knowledge into technology (x1) is shorter than the time to develop support tools from technology (x2), and which is shorter than time to use the tool to develop the product (x3). However, this may not be true for all technologies. Sometimes, innovations in one technology may have to wait for a long time for an enabling technology to evolve before marketable products can be realized.

6. The Kozmetsky Effect

Prof. George Kozmetsky of the University of Texas, Austin, an internationally acclaimed business leader and a National Medal of Technology recipient, has shown the existence of a strong interaction between knowledge and technology growths. According to him, the rapidity of the knowledge growth exerts a strong synergistic attraction on the technology growth and pulls technology growth curve towards it (Fig. 6). In other words, the technology transfer latencies get reduced or compressed with a maturing technology. Or, the technology transfer time gets shortened with a mature technology. As alluded earlier, there are two reasons for this. First, knowledge and technology interact vigorously, thus accelerating the corresponding tool development. The knowledge technology curves get closer and closer to the tool development and implementation curves. Secondly, as knowledge saturation in a particular technology takes place due to physical limits, we only encounter incremental advances but no big break-through. Thus the tools and the manufacturing processes may not change significantly In effect, the technology progresses ever so slowly and incrementally. Due to this convergence of the knowledge-technology transfer curves the phase latencies shrink and may ultimately coalesce. This implies that the technology has reached a saturation point and progress will be difficult, like trying to squeeze water out of a desert rock. We shall call this convergence of knowledge-technology-transfer phases as the Kozmetsky Effect. This creates an inflection point in the technology growth curves similar to the Dr. Grove's inflection points [3] in the growth curves of hi-tech enterprises. (The Economist, Dec. 31, 99 defines the inflection point as 'an event with potential to change competitive landscape so fundamentally so nothing can be same again.')

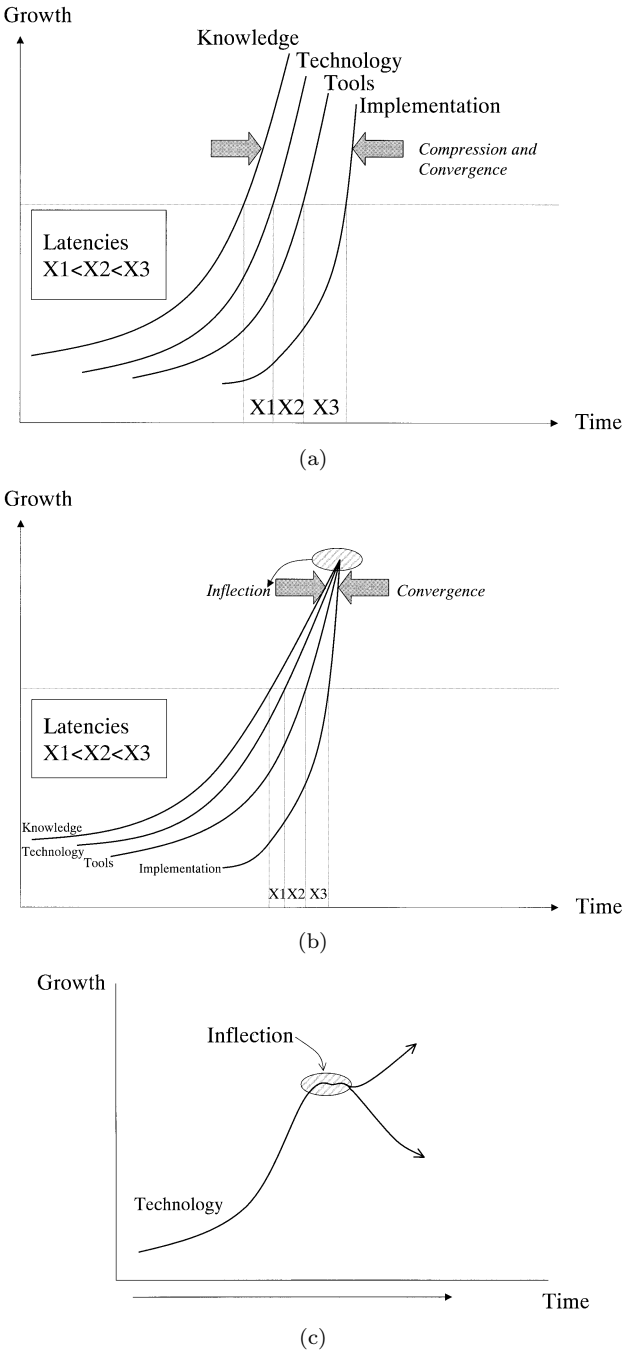


Fig. 6 (a) The compression of the technology transfer phases (the Kozmetsky effect). (b) The convergence of technology transfer phases. (c) Technology growth inflection point.

Tight marketing deadlines and product release schedules may hurt the quality of the final system (product or service) because of the haste associated with the development. In the service industry, this can be inadequate preparation in the new training courses or maintenance procedures. In the software industry,

tight deadlines are often blamed for inadequate product testing, and therefore, for the dismal quality and failures of the product in the field caused by residual errors still remaining in the system. This may explain why our high tech products undergo several incremental corrections—humorously called revisions, versions and releases—over their product lifetime.

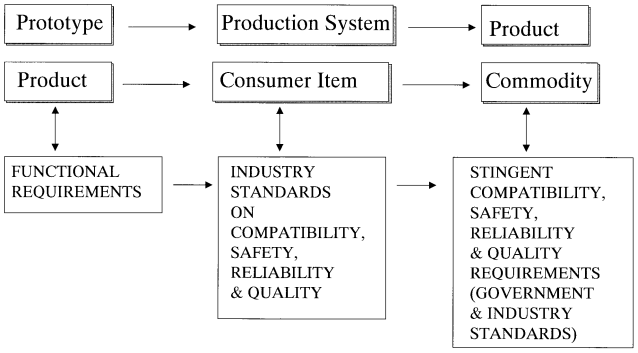
We conclude this section by commenting on two major characteristics of knowledge-technology synergism. First as stated earlier, knowledge doubles every seven years, i.e., knowledge grows exponentially [1]. Secondly, technology too may be similarly growing in an exponential fashion. Moore's law, associated with the semiconductor technology, hypothesizes that the performance of semiconductor chips doubles every 18 months while the cost remains the same, and therefore, follows an exponential growth. One sees the similarity and the relationship between these two empirical laws. But Moore's law cannot be sustained indefinitely due to physical limitations Thus, as the technology growth reaches its peak and begins to slow down, a new fledgling technology may take over. We can justly say that 'old technologies, like old soldiers, never die, but they are just left behind'—a saying attributed to Dr. Andrew Grove of Intel.

7. High Tech System Evolution

In this section we shall consider the evolution of high tech service products like microprocessors or application programs that are the harbingers of the new service era. These products are initially produced in small quantities and sold at high costs. They are considered as 'commercial products.' They may become very popular, and create a large consumer demand. These are, then, produced in large quantities and sold cheaply. They are called 'consumer items.' Consumer items are implicitly subject to stricter safety; reliability, interoperability (open-systems) requirements and environmental constraints set up by the industry than a commercial product. If these products become essential needs in our lives, they become 'commodity items.' These items are mass-produced and would be available in many retail stores. Commodity items are subject to both to industry standards and to governmental and environmental regulations. The implicit requirements will be stricter than those of similar consumer items. An important class of consumer items is the digital appliance. This is a simple, generally, single function-based, easy to use, inexpensive but reliable digital device. Examples of digital appliances are pocket calculators, digital thermometers, or digital watches and organizers. A consumer item reaches the status of a commodity when its use is widespread, and its need becomes essential, like clean air, water, food grains, gas and electricity—items that we cannot live without. In essence, it fulfills a major service function. A long list of products have received the commodity status in the high technology field, such as digital watches, cellphones, TV, VCR's, PC's etc. (Do you remember the time when you misplaced your TV remote controller, and how much inconvenience you had until you found it.)

When high tech products reach the consumer or commodity item status, they will be sold in large quantities in very competitive markets, their manufacturers feel intense pressure on profit margins. The customers will often focus on price, while taking the things like reliability, safety, quality, etc., for granted. Thus, as the evolution proceeds from a system prototype stage to a salable product, then to a consumer item, and then to a commodity item, the consumer expectations rise, creating stricter implicit requirements in their wake. The designers must remember and factor these during the development of architecture of the product family. These implicit requirements are distinct from the explicit requirements which are generally the functional requirements initially used by the designers during the first product creation. A good example of implicit requirement is the electromagnetic radiation standard imposed on TV sets, computers and cell-phones. Product designers have to foresee these and create product family infrastructures to easily accommodate these evolutionary (implicit) requirements during the life cycle of the product family. It is somewhat similar to requirement imposed on high-rise building designers in Missouri and Oregon to consider the possibilities of seismic catastrophes even though there never was a major earthquake in those states. (The lack of a spare tire is no concern if you do not get a flat, so said Alan Greenspan.) Figure 7(a) summarizes these findings.

We shall extend the evolution of implicit requirements by considering the some new trends in software product family design by referring to Fig. 7(b). Here the rows refer to the commercial product types and the columns refer to the various standards and regulations in the order of their strictness. We summarize what is depicted in the figure as follows: The initial company (proprietary) product is based on the core functional requirements plus some company preferred quality and performance characteristics. When the product becomes a consumer item, it is subjected to industry standards in addition. When it becomes an open systems product, it is regulated by the industry for interoperability. When it becomes an open software source system, like LINUX, its design and code is open to all users and developers, who have a say in any future modification, change and enhancement. It is thus regulated by the industry and its developers and users. When it reaches the status of a commodity item, it is governed both by industry and government standards and regulations. Ultimately, the products may be subject to community's environmental and ethical concerns. There may come a time when we may require every service



(a)

Service Product Evolution

	Company Standards	Industry Standards	Industry Inter-operability Standards	Developer Community Requirement	Government Regulation	Environmental & Ethical Obligations
Commodity Item	1	1	1	1	1	1
Open Software Source	1	1	1	1	1	-
Open System	1	1	1	1	0	0
Consumer Item	1	1	-	-	0	0
Proprietory Product	1	1	0	0	0	0

Standards, Regulations, & Responsibilities

(b)

Fig. 7 (a) Product evolution and implicit requirements. (b) Extended view of product evolution and implicit requirements.

system and software developer to take an oath like the physician's Hippocratic Oath which says, 'I shall do no harm,' or equivalently, 'I shall not introduce any malicious viruses and activities.' This possibility is indicated in the last column.

8. Surety Engineering

The implicit requirements, as we have seen in the previous section, grow more stringent as the product evolves from a simple prototype to a consumer or commodity item. Of course, the nature of these requirements will depend on the applications. Sandia National Laboratories developed a conceptual model based on 'Surety engineering' [4], which helps us to classify applications based on the nature of the requirements. We have extended and in some cases modified their ideas, to fit our aims. 'Surety is the methodology by which any system can be designed to operate exactly as planned, every time in every circumstance. Surety is a way of approaching just about any systems problem that considers why and how a system fails, then anticipates and prevents such failures' [4]. Products based on these concepts are guaranteed or assured to be safe, reliable, secure over specified lifetimes, under pre-defined conditions of use. Surety engineering concepts are particularly attractive for specifying the implicit requirements of high tech services and their products, as they evolve from purely lab prototypes to marketable products, then to consumer items (or appliances) and then ultimately reaching the star status of commodities or necessities of life.

Our extended version of surety engineering distinguishes five levels of product safeguards. Each level includes the functions of its lower level predecessor, the highest level being the most stringent. This follows the original concept of Sandia, 'the greater the consequences, the higher the surety level.' We shall use Sandia's examples in the following discussion.

The lowest level, Level 0, personifies products that are cheap (often called throwaway or non-maintainable items) for which the manufacturer provides little or no guarantee or safeguard except it works as intended, at least at the time it is sold. Cheap information appliances correspond to this category.

The next level or Level 1 safeguards are primarily reactive. When the system recognizes or detects a system failure, it reacts to it, and it may take some corrective or evasive action to mitigate the effects of the failure. It is suitable for applications, where a failure would not create catastrophic or collateral damage. The system is based on very reliable design and uses very reliable components and where failures are highly unlikely and infrequent. Most importantly, it is assumed that the system failure is recognizable after it happens, repairable and the system operation is restorable, without creating undue inconvenience. Level 1-surety measures may include consumer protection under product liability warranties, insurance against property damage, etc.

Level 2 surety is proactive. Its purpose is to avoid failures as much as possible by taking precautions and instituting surveillance measures like metal detectors at airports to prevent sky-jacking and rigorous training of pilots and crew. It would include the measures outlined for Level 1 surety because it provides all lower level safeguards.

Level 3 surety is preventive. Its purpose is to predict and lower the odds of a potential system failure by careful redesign of the system. Because the automobile accidents kill, manufacturers incorporate seat belts and airbags, not just as add-ons but as integral parts of total vehicle design. It would also include the safeguards required of Level 2 surety.

Level 4 surety is applied to systems where failure is not acceptable or unthinkable, that is, surety safeguard is fundamental. To ensure absolute surety, the design is made highly reliable and fault tolerant so that safe predictable behavior is assured under anticipated failure conditions. Examples of applications fitting this category include nuclear power plants and weapons systems.

The surety levels which we associate with implicit

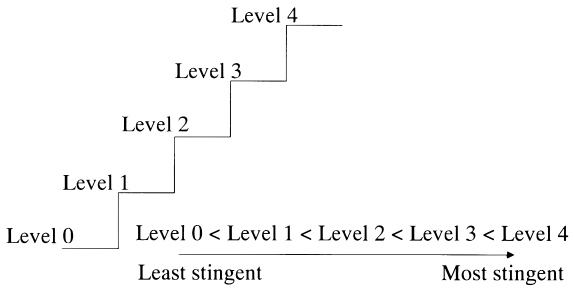


Fig. 8 Surety levels.

requirements on services and products follow a transitive relationship, in which the implicit requirements in Level 0 are included in Level 1, which are also included in Level 2 and so on, through Level 4. This is portrayed in Fig. 8. The modeling of the utilization of high tech services and products illustrates the fact that as these systems or services move up in use and popularity, their providers and manufacturers have to satisfy increasingly stringent but unwritten surety conditions. When Intel Corporation introduced the Pentium microprocessor, it initially ignored a design error in its floating-point divide command. The company felt the resulting numerical error was very negligible and the frequency of use of the operation was small. But after an academic user discovered the error, a large number of customers demanded an immediate remedy, even though it did not affect their applications. Intel realized that its product had become a consumer item where trivial errors are not tolerated. It redesigned and reissued a corrected version of the Pentium at a cost upwards of $200 million.

9. Errors in Service Engineering Activities

Service activities require intensive interaction amongst people and machines. It is very vulnerable to intentional and unintentional errors. In the following sections, we shall explore ways of mitigating such errors. As stated in the beginning studies of Andersen Consulting and others have shown that about 40–80% of the failures may be due to improper human reactions.

To help reduce accident and error incidence resulting from man-machine interactions, designers have considered two general methods applicable to a broad range service applications. One is the process of humanization, personalization and application specialization. And the other is the use of teams—groups of professionals (usually a small number) working together—to accomplish the tasks. We shall discuss these in the subsequent sections.

10. Humanization, Personalization and Application Customization

We work long hours in intense interaction and close working relationships with their computers. We prefer to treat these intelligent machines as human-beings and expect to receive human-like responses from them We can 'humanize' the machines to imitate human-like reactions under selected circumstances, so that their human users are not confused or stressed. This built-in 'naturalness' makes it easy for us to understand, learn and use them better. We shall call this 'humanization.' Examples of this include providing voice or visual advice, guidance, and warning, before or when an operator inadvertently commits an error. The messages could carry emotional content! These are common in PC publishing software.

Individuals differ from one another by gender, age, education, and physical and mental ability. One can 'personalize' the target service system to conform to individual user's disabilities, preferences, and fancies. We call this 'personalization.' Examples of personalization are TV remote controllers for left handed people, telephones with volume and tone control for the hard of hearing, and use of large letter fonts in computer displays for those with impaired vision. Thus 'humanization' of the machine makes it easy to use since its responses will be less confusing to the humans, whereas 'personalization' individualizes the system to the particular human user to compensate for physical disabilities or to support his/her preferences

'Customization' adapts a general-purpose computer or a machine which is generally, a cheap, mass-produced consumer or commodity item to perform more effectively and efficiently in a specific class of applications. This is generally done by adding specialized instructions, or ROM-based firmware or software adjuncts. Intel's Pentium MMX microprocessor comes with 57 special instructions to support multi-media applications. Application customization is very common in the PC domain, particularly in visual graphic support, in database accelerators, and in sound cards for enhancing audio performance.

In summary, we 'humanize' the machines so that their responses are natural and human-like. We 'personalize' the machine responses to fit the user's individual preferences or disabilities. We 'customize' or precondition general-purpose machines to perform specialized application-based tasks efficiently and effectively.

10.1 A Layered Hierarchical Model for Application Customization, Humanization and Personalization

To develop system architecture based on the ideas outlined in the previous section, we shall propose a layered model much like the one used in communication protocols generally known as the ISO Model. In systems engineering, complex designs are often modeled and constructed by hierarchical layering. Each layer is associated with a set of service processes. The layer

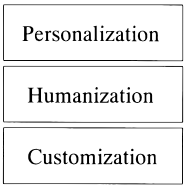


Fig. 9 Layered hierarchical model.

at the lowest level interfaces and interacts with the fast electronic and physical elements. The top most layer generally interfaces with applications and users. Through a series of layers, the top-level (the highest level) tasks are translated into machine recognizable electrical signals at the bottom-most layer [5].

Our model consists of four layers (Fig. 9). The lowest layer will be the general-purpose microprocessor or micro-controller. The next higher layer will be the application customization layer, which preconditions the machine to perform specific application oriented tasks efficiently. This is similar to the enhancements and the add-on packages (or boards) achieve speed-up in database accelerators or graphical visualization systems. The next higher layer will be the humanization layer, which helps to the system to react in a humanistic fashion. This could be also be dependent on the application tasks but it will focus on general human performance and reactions The top or the fourth layer is the personalization layer which tries to help out in the personal disabilities or caters to user's preferences. Individual personalization is left as an option to be exercised by the user and therefore, some manual controls must be provided where warranted.

10.2 Comments on Humanization

The purpose of humanizing a computer or a machine is to make its interactions as natural and human-like as possible to its users. By humanizing, we try to reduce the circumstances where the human user will be surprised, uneasy, inconvenienced, confused or stressed. Computer and other intelligent machines have become our constant companions. We crave for human-like relationships with them. We expect human-like interactions with machines and we apply subconsciously social rules when we interact with them, such as politeness and respect. We prefer a female voice for soothing after making a mistake and a male voice to admonish us before committing one. Unfortunately, our current systems are not designed to be humanistic and behave in a predictable way. This is a likely source for a large fraction of machine operator's mistakes. The majority of our responses during man machine interactions fall into one of the following categories:

a) Responses based on inherited and built-in natural instincts and reflexes-often called the knee-jerk responses. These are fast responses where no thinking is required.
b) Responses based on acquired knowledge, experience, education and training. As knowledge accumulates and evolves over time, our habits and responses tend to change. These are thoughtful or deliberate responses, which require introspection.
c) Responses to new and unanticipated situations which have to be dealt without prior training, preparation, experience and knowledge. While responses to these may create risk but they also provide exposures to new learning experience. These require research and more exploration or knowledge-gathering.

For the first two categories, we can respond effectively using our built-in instincts, experience and knowledge. For the last category, we are left in the dark. People may look over their neighbor's responses and do (copy) the same, much like the way we follow the main traffic flow when we are lost in a fast freeway. Or succumb to Turtle's casino effect—the addictive habit of continuing to gamble even if one has lost heavily. Or, moving in the direction of increasing illumination when we are lost in the dark, or doing what may be considered as politically correct, even though it is against our conscience or moral judgement.

We next summarize or quote some eminent social scientists on the man-machine interactions without comment. These include Profs. Sherry Turkle of MIT, Byron Reeves and Clifford Nass of Stanford, W.C. Frederick of University of Pittsburgh and journalists like Thomas Pitzinger Jr. of the Wall Street Journal [6].

Information technology has changed the way we interact with the world at large and with the machines in particular. People have become the connoisseurs of information. One can go on-line, learn more about the product one plans to buy, make a rapid price comparison between different stores and then make a purchase. There is an innate yearning for the human touch that we believe will be provided by the computer. There is a lack of predictability when we deal with machines. "The future has so much uncertainty about it that we need to prepare our children to see the future in a different light." Prof. Turkle tells about the Furby doll, a fuzzy computerized mechanized doll that talks, blinks, sleeps and asks to be fed. The actions of this doll are totally unpredictable. Compare this with a Barbie doll that behaves predictably. Playing with the Furby doll makes the child get used to unpredictability and uncertainty. The child will not be uncomfortable or ill prepared in the future, when dealing with unexpected circumstances or interactions with machines whose behaviors are not predictable.

All living things harbor an impulse to economize, to accomplish more with less. Economizing process is only way to survive, grow, develop and flourish. 'The genes that create us have programmed us for business-

the foundations of which are trade, technology and division of labor.'

11. Teams and Team Evolution

Since service functions are enabled and executed by teams, we shall briefly comment on some pertinent characteristics, specifically their types and functions. The word 'team' is defined in the Webster's dictionary as a number of persons associated in some joint action like a team of experts, or 'draft animals' harnessed together to draw a vehicle. P. Senge [7] explains the meaning of a team as, 'an alignment of people when it functions as a whole. When a team gets more aligned, a commonality of direction emerges, the individual energies harmonize.' Teams in our discussion are made up of people supported by machines to achieve certain goals during some specified period of time in a cooperative or collaborative fashion. Teaming includes the processes of planning, organizing, controlling and executing the required tasks by teams.

Teaming is a primitive (herd) instinct inherited by man from his animal ancestors, practiced by predators, hunters and scavengers amongst animals, and used by street gangs and the military amongst the humans. In ancient days small teams (typically from 2 to 7) worked together as craftsmen at the home of the merchant-craftsman. This is the forerunner of home-office and telecommuting concepts. Their organization was tight knit and hierarchical. The 1770's marked the start of industrial revolution in Europe funneled by the steam engine technology. It used the division of labor, and the work place shifted from the home to the factory. Here the teams employed anywhere from 200 to 2000 people. With the growth of assembly lines in U.S. (thanks to Henry Ford), the teams became more specialized and manually intensive—using more brawn than brains. In the future, with the advances in information and communication technologies and the ever-spiraling inconvenience of rush hour traffic grid-locks, telecommuting and home-offices will become popular, where teams will virtually work together from the more pleasant surroundings of their homes at different physical locations. Lo and behold, our ancient craftsman has returned!

As the technologies begin to interact, the systems and their designs become more complex. Teaming becomes tools for design, implementation, prototyping and design verification. Team members learn from each other, complement and supplement each other's expertise and share the developmental burden they have been very effective in the detection and correction of errors before the designs are implemented. These activities are done by groups of people since no one single person can do them alone. Teams, when properly organized often produce innovative and bold ideas.

Gersting and Ives of Anderson Consulting (Solution Integration Magazine, July 1999), state that 87% of failures and costly mistakes are caused by lack of knowledge on the part of designers, operators and users. Several of the catastrophic disasters are traced to operator errors during unexpected and unanticipated events. Teams of knowledgeable operators, users and experts can, with proper computer aided support can help reduce this. Oftentimes, the critical decisions made by properly qualified teams are superior than those made by a single human. Poole [8] provides arguments and instances taken from safety critical applications to support the inherent error detection and error prevention effect of teams. Quality Circles used in software development in Japan utilize well-chosen teams of professionals for early detection, location of errors in large software systems. The IBM's Clean Room software development procedures advocate the multiple application of inspection teams for the same purpose.

A well-coordinated group of people was found to be very effective when working with large complex computer-communications systems in FAA, US Navy's Nuclear Submarines and aircraft carriers.

A cardinal advantage of any team-based activity is the learning experience gained by its members. Cross-disciplinary team members provide both supplementary and complementary support in accomplishing a complex intellectual activity. We shall cite a recent example.

The chess match that culminated on May 11, 1997 pitted Chess Champion Garry Kasparov against a human-machine team that included the IBM supercomputer, Deep Blue. The Champion lost The news media came to the conclusion that the machine has defeated a human champion, ignoring the fact that the computer and its programs were all developed by a superb group of cross-disciplinary experts and mathematicians. Indeed one can say that a team of humans with support from machines can solve more complex problems than it is possible by single human alone. A machine by itself cannot solve a given structured problem unless it is provided with intelligent software tools encapsulating the chess-players' knowledge and tactics (in the form of algorithms). In the case of the Chess Championship, the Deep Blue's programs (developed by several teams) looked 2000 steps ahead on all possible moves of the opponent, before it made its move. The machine also used the playing style and tactics of the champion, Kasparov, from his several previous games as inputs. One can conclude that a team of human specialists augmented by powerful computers and programs can out-perform a single human expert.

One of the assumptions we make is that a computer cannot be smarter than its designers—who are human beings. A person with computer resources can be 'superior' to a person without such support. A team of humans properly supported can perform better than a single person with a machine. Ultimately, groups of teams could be more efficient than a single team. These

are just empirical observations and it is good remember the context and the implicit assumptions that we made. In essence, we can hypothesize empirically from these observations the following:

> Machine < Human < (Human+Machine) < Teams of (Human+Machine)
> where the symbol "<" implies "inferior to" or 'not as good as' relation, and the symbol "+" refers to "logical or," or, "either one or both" relation.

In a simplistic way, the Turing Test (1950) sought to establish whether a device possesses human-like intelligence or not. If the test results on both the machine and human show that one cannot the distinguish two results-the results from the machine matches to those of the human-then one can conclude the machine possesses human-like intelligence. Turing provided little or no details on how such a test can be conducted and the results are to be evaluated. H. Dreyfus, from a phenomenological basis and John Searle from a linguistic and cultural basis (Chinese Room Argument) question the fundamental premise of the test.

11.1 Team Types

Team type depends on the main purpose for which the team is formed. It defines the organization and the nature of dependency amongst the team members. The criteria one can use in classification are the team's purpose the nature of its collaboration, and its organizational and control philosophies. These govern the nature of interaction and dependency amongst its members We shall only discuss some specific but important types of teams used in the service functions namely, the collaborative, co-operative, consensus seeking, hierarchical and virtual teams.

Collaborative team members work together in small groups, very closely, i.e., cohesively, to achieve the best outcome for their intended goal. Generally the goal of the task is well defined, and the team members are chosen because of their motivation with the objective, perceived potential and their competence in that area. Collaboration creates a synergy such that their total contribution is greater than the sum of their individual contributions. The team members are tightly coupled who work together very intensively and strive to achieve the very best outcome. A parallel concept from software engineering of this is cohesiveness, which implies that the components (members) of the team are very dependent on each other, and work together very closely and effectively.

Cooperative team members support, and interact with each other in a loose or a loosely coupled fashion. Their participation may be sporadic or may work together on an 'as needed' basis because of their specialized expertise. They may be located at different places. Two collaborating teams may be cooperating and therefore will be interacting with each other in a loosely coupled fashion. Example of a cooperating team would be a group of consultants, who work with each other in some specific phase or problem of a large project because of its expertise in that activity.

Consensus developing teams are organized to create an acceptable or a negotiated solution for the critical issues. When a team is tackling a large number of trade-off issues, the team members may not all agree on any one of proposed solutions. To break the deadlock, it is usual, for the team to agree only to the least contentious but most agreeable compromise solutions. This may not be the best, but it would be acceptable all around. For the sake of time and expense, all or a majority of the team members may go for this 'least common denominator' or 'politically correct' solution. Practices such as the quality circle method of evaluating the quality of software follow this principle because of numerous trade-off issues. Consensus teams contain members from a diversity of interests, and always strive to find the best workable common ground. Another simple example is a team of jurors in a legal trial.

Virtual teams may be considered as simulated representation of real teams. Political pollsters use this method to gauge the public support for several office seekers. They provide the freedom to experiment and study the results about activities of people and machines by computer simulation. Irreversible and actions which either take a long time or too dangerous for realistic evaluation can be studied in this virtual world scenario. Virtual teams may consist of software agents, representing specialists (expert systems), or monitors and observers who record critical events during simulations. Currently the phrase 'virtual team' also is used to denote a geographically dispersed group of people forming a team.

Hierarchically organized team has a command and control type of relationship amongst its members. A member in a lower hierarchy reports to or obeys the orders of those in the next higher level. A common form of the model is an inverted tree, the root node is the commander and the leaf nodes are soldiers. The nodes in the intermediate level represent officers. Hierarchical teams are tightly coupled. Examples include military command and control systems, gangster teams, drug cartels and the common master-slave systems.

We shall next comment on other types of teams, whose dependency objectives may be useful for service type operations.

Reciprocal dependency teams understand each other's goals and try to eliminate conflict, competition or confrontation by working around them. The conflict may be about their need to share some common resources and therefore requiring them to rearrange their activity schedules to avoid conflicts. In another type of dependency, called mutual dependency, each team needs the other's help to achieve its desired

goal. Benevolent dependency-based teams try to help the overall society and encourage others to do likewise.

Altruistic teams sacrifice their goals for the common good of the society without any direct benefit for themselves. This type of behavior is observed amongst animals that sacrifice themselves to save their herd. This typifies the moral: 'All for one and one for all.'

11.2 Cohesion and Coupling in Teams

Similar to the terminology used in software engineering, cohesion is the property that attracts, or binds two entities together to produce a new one with different characteristics. It is a dependency relationship and in our case, produces tight collaborative interaction between team members. This tight relationship overshadows their individual behaviors. It is a synergy producing relationship or fusion. Cohesion exists between humans and their computers. A team of people working in an assembly line all participating in a single task exhibit cohesion. On the other hand, coupling is a relationship, which preserves, in some way the individual properties of the constituents, like a team of consultants all working on different aspects in a loosely coupled fashion. It is a dependency relationship between larger aggregates. An analogous, often-cited concept is from chemistry. The atoms form molecules and molecules combine to become compounds. We shall give an oft-cited example. Sodium (Na), an explosive element combines with Chlorine (Cl), a highly toxic element to form common salt (NaCl), which is neither explosive nor toxic. NaCl is a molecule or a simple compound. This illustrates the important property that molecules formed from cohesive combination of atoms may not exhibit the properties of their constituent elements. Compounds produced from the interaction of molecules may retain the characteristics of their constituent molecules. Cooperating teams show loose coupling whereas collaborating teams show tight coupling, or cohesion.

11.3 Evolution and Control of Man-Machine Teams

In this subsection, we shall model the evolution of teams particularly those where the team members are heavily computer supported as in the service industries. The four models use simple directed graph structures.

1) In the first model, each node represents a team, and is independent and isolated. There are no links or arc emanating from the nodes. Nodes have little or no communication capability, that is each node is isolated from others. Each node makes its own decisions. (They can be ripe merger or takeover candidates). We shall label this as a totally isolated team model. The nodes are like islands of isolation developing themselves over time-very much like the ancient clans who have their own languages, religions, and cultures which are local and specific, uninfluenced by their neighbors. This is shown in Fig. 10(a). Here the cost of communication across teams is zero, and there will be no fear of virus invasion or security breaches.

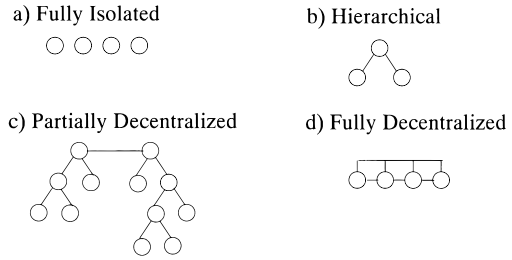


Fig. 10 Team types.

2) The second model centralized model which portrays a top down hierarchy, in which one node, the root node of the tree acts as a controller or director receiving information from its subordinates and giving directives to its subordinates. This models a centralized enterprise, where the head office receives status information from the branch offices and sends out directives or commands to the subordinate nodes to carry out. Military organizations, which use the command and control regimen, and street gangs are examples of the centralized, tightly coupled hierarchical organization. This model is illustrated in Fig. 10(b). This model is suitable where the communication costs are high and therefore communication is kept to a minimum. This philosophy is also used in the obedience training schools for dogs. This system works well when the number of branch offices (dogs) is few, and there is no need for real time updating of the information at several sites. The down side of this organization is there is no interaction amongst the nodes, the head office can micromanage the operation of its branches.
3) The next model (partially decentralized model) is illustrated in Fig. 10(c). The sub-trees under the nodes B and C are organized like the centralized model. The nodes (teams) B and C report to the centralized node A. We can consider nodes B and C as regional head offices. The enterprise is still controlled by a single headquarters node A. By partially decentralizing the system, the computational burden on the central node is lessened. This model provides the next step towards expansion or evolution of the centralized model described above. The communication requirements can be high depending on the size (number of nodes), and the structure of the system.
4) The fourth model is the fully decentralized model and is shown in Fig. 10(d). This model assumes full connectivity between the teams or nodes and the inter-node communication is assumed to be

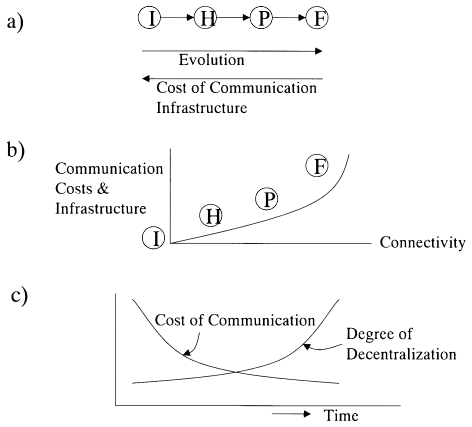


Fig. 11 The effect of decreasing cost of communications and increasing decentralization of control.

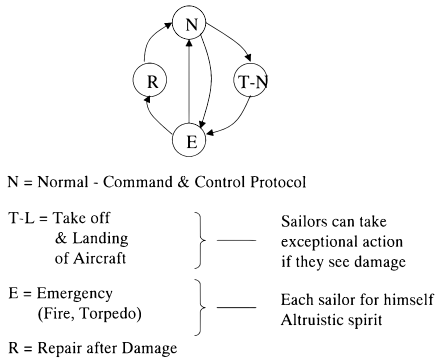


Fig. 12 Finite state machine model of air craft carrier control and organization.

instantaneous with negligible (e.g., the Internet) cost. Ideally, the teams can have total access to each other's information databases at all times. Each node makes functional and administrative decisions rapidly and collaboratively with others based upon a set of agreed-upon rules, protocols and principles. The designers of the system must carefully check and validate these rules and roles to avoid conflicts, inconsistencies, and ambiguities to thwart any security breaches and catastrophes that could occur during faults, malfunctions or virus invasions on the system.

We envision that a realistic system may portray any one of the above organization models or combinations thereof. A flexible or a reconfigurable system can switch between these configurations to adapt to the changes in the application, communication environment and other conditions.

We infer through our classification and discussions, that as the communication and computational costs go down, the system evolves from an unlinked, isolated and independent enterprise units (Fig. 11(a)) to a centralized or hierarchical configuration (Fig. 11(b)), then to a partially decentralized configuration (Fig. 11(c)) and finally to a fully decentralized configuration (Fig. 11(d)). This is very similar to the cultural evolution through globalization that we see around us. Once not very long ago, the world was littered with kingdoms, nationalities and irreconcilable differences due to non-interaction and lack of communication. But now it is giving place to a democratic and cooperative world—towards a fully decentralized system—which is made possible by the advances in education, information and communication technologies.

12. A Finite State Machine Model of the System and an Actor Model of the Component

In this section, we model the organization and control of man-machine teams. We show how this organization and its control structure can be changed by different circumstances. Every state in this model is a meta-state-an ensemble of states. A finite state machine (FSM) model provides a conceptual framework for the rest of the discussion. The behavior of the team is represented by the finite state machine. The entity or team member is modeled as becomes an actor, who assumes a given role during that state and acts according to a given set of rules of behavior. The team member is a service provider. When the system is in that state the entity follows the rules of behavior assigned to its part or role. The play is personified by the state. When the finite state machine of the system moves to a different state, the play changes. The actors, the components, entities or team members assume different roles in a different play and act their part accordingly. To reiterate, when the system state changes (a new play is initiated) the team member assumes a new role and acts under a different set of rules. When the system encounters a different situation, the system state changes (the system reconfigures), the team members assume new roles and the system changes its organizational and control structure.

We shall provide an example first and then create a finite state model from it. The example is taken from Poole [9]. A navy aircraft carrier has many sailors and officers in it. We will use the sailor as the model of an actor. We represent the states of the sailor with a finite state diagram as shown in Fig. 12. State A is one where the sailor does normal routine operations assigned to him. These are based on standardized procedures. In this state the sailor follows the command and control protocols and procedures. The command and control procedures and protocols prevail. The organization is hierarchical, i.e.; the command orders propagate through a chain of command in a top down fashion. The responsibilities and the duties are as prescribed in the command and control rulebook. Ranks and authority are recognized and strictly enforced. The primary concern here is performance efficiency. The State B

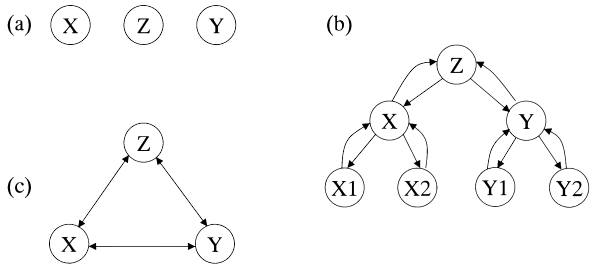


Fig. 13 A model of financial institution.

in Fig. 12 exemplifies the aircraft launch and recovery operations. These operations are most stressful to the personnel and sometimes quite dangerous. Every man and machine have well-defined functions and the sailors and officers are trained for this. The primary concern here is the safety of the individuals, the aircraft and the ship. Ranks and commands can be ignored for the sake of safety. If the sailor sees that a plane's landing gear is not in proper position during its landing, he can forget the protocol and wave off the plane not to land or abort the landing.

The third state, State E, represents an emergency or an unanticipated situation on the aircraft carrier. This could be due to a fire on the deck; a missile hit on the ship etc. Even though the personnel of the carrier may have been drilled for several types of emergency, most of the accidents cannot be anticipated. In such situations, each sailor acts for himself as well as for the total benefit for all in the carrier. The emphasis, in such instances, is one should not panic but one should exhibit an altruistic, predictable and orderly behavior. Collective safety and altruistic spirit takes precedence over one's own personal safety, rank and protocol. For simplicity, state E may also include subsequent repair and reconfiguration of the damaged system.

We have shown above is an idealized version of the states the aircraft carrier and the role of one its components, a sailor, as they go through their life cycle. A finite state system models the states of the carrier. The actor model fits the component, the sailor. We will not discuss the state transition conditions since they are application dependent.

12.1 An Example of Evolution of a Financial Institution

This is a legacy application, in which initially the financial institutions X, Y, and Z were in geographically different locations with little or no communication between them (Fig. 13(a)). The institutions merged their enterprises, with Office Z becoming the head-office. The central data files for the whole enterprise, X, Y and Z are located in Office Z. There is data (transaction) update activity between X, Y and Z. The headquarters node Z controls the system. It has access to the data files at the branches. The headquarters node Z has the leadership role. This is an example of a centralized (hierarchical) system. When more branch offices are created the enterprise establishes several centralized systems each controlled by a regional head-office as shown in Fig. 13(b). The regional head offices are again as a tree with a single root node serving as the enterprise head-office. This organization the old hierarchical structure is maintained and therefore old procedures used in the previous system can be re-used. This is the example of a partially decentralized system. When the communications costs go way down and become negligible, all the files are accessible from any node or office of the enterprise (Fig. 13(c)). Any high-level decision can be made and administered at any node or office at any time. In essence, each node can make decisions with the instantaneous participation of other nodes. Each node acts in conformance with the rules and policies governing the whole network. This represents what we called earlier, the democratic or decentralized way of control. It is also an example of decentralized autonomous control that empowers the teams in an industry to perform their very best efficiently in rendering services to their customers.

12.2 Comments on the Models

These models are implementable and simulatable for an arbitrary system. The actors or components are man-machine teams. For example, the team members are tightly coupled and hierarchical in one state, but change as the circumstances warrant. They may become partially or fully decentralized as the situation unfolds. Within a team the members are collaborative, whereas the teams within a system state may be loosely coupled. The behavioral rules of the actors within teams and the rules that govern inter-team interaction must be consistent, and complete but also non-conflicting and unambiguous. We shall conclude this section with an analog taken from digital design. Consider the design of a time-multiplexed subsystem of four flip-flops. We wish to reduce the hardware by time-sharing them in many computer functions. We can design an incrementing counter, a right shifting register and a complementation register using the same four flip-flops. The designer designs a system that when a shift command signal is given, the subsystem performs one position open-ended right shift and later when an increment command comes in, it increments the contents of the four flip flop register by one, and so on. In other words, during a shift command the flip-flops behave as a shift register. Under the increment command the subsystem behaves like a counter. The flip-flops change their role according to the command they have to execute. This is very much like the same actor performing the part of Hamlet in one play, and the role of Othello in another but still working for the same playhouse.

13. Conclusions

Service functions have matured, and have reached the most dominant position by entwining themselves with the advances of information and business technologies. Software engineering has been the technology catalyst, a behind-the-scenes player but the major enabling discipline in all service sectors. Recent data from the Bureau of Labor Statistics indicate that more than two-third of all jobs in US involves service functions. In this paper, we have traced the evolution of the service industry. We have overviewed the distinguishing features of its functions. These vary with the application and the industry. Generally they involve interactions amongst teams of professionals and machines. Faults and errors inadvertently or intentionally introduced during the performance of service functions can create failures and breakdowns in critical systems.

Transfer of knowledge and technology into the building of commercial products and services (also known as technology transfer) is an important service function. We decompose it into phases, which include the development of appropriate technology from scientific knowledge, the creation of tools (mostly software), and then, designing, simulating, prototyping, assembling and implementing the product.

As we trace the evolution of innovative technologies over the last two centuries, we note that automated machinery and processes are replacing the manually intensive activities. These automated tools and machines have benefited very greatly by advances in information technology and artificial intelligence. Automation always improved the productivity and reliability of products, and services. It also eliminates a large portion of manual and mental interaction errors as well, which currently account for 80% of all failures in critical systems. Such errors are the result of improper user/operator's interactions with the machines. We argue that by proper humanization and personalization of machines and computers, the failure incidence can be greatly reduced. The use of human teams with computer support had been found to be very effective as well in reducing errors in design and operation. Interaction amongst team members help their rapid learning, and towards sharing and creation of knowledge.

The type, organization, and control of man/machine teams amongst other things depend on the application. The Finite State Machine and Actor models fit well in portraying an important class of teams. During a particular state of the system, the teams (components) or actors follow a specified set of rules of behavior and operation. During the next state, the actors follow a different set of rules of operation and behavior as if they are doing in a new role in a different play. The state transitions depend on the application events and circumstances. These models are helpful in the simulation and analyses complex man-machine systems.

We consider four distinct types of team organizations, namely, isolated, centralized or hierarchialized, partially decentralized and fully decentralized. When the communication and computer processing costs are negligible, fully decentralized disciplines appear to be most congenial to the rapidly evolving technologies and economies.

There exist lots of research issues worth looking into. We shall touch upon a few. The performance evaluation of service functions and activities depends on customer satisfaction. We need to establish the criteria, the measurement metrics to detect trends of what customers desire to develop suitable data-mining approaches. Currently we are studying representations of service activity work elements, like the one we have used in this paper. We have combined the ideas reminiscent work of Gilbert and Taylor of the time and motion study and operations research fame and the ideas of computer programming to represent a service activity as a program entity and perform an entropy analysis similar to the ways Shannon did for information theory. These will be subjects of future papers.

Acknowledgements

The author owes a deep debt of gratitude to Prof. G. Kozmetsky for his invaluable advice and encouragement. Also Prof. R.T.Yeh provided valuable insights while Profs. Kane Kim and Kinji Mori helped in clarifying several ideas discussed in this paper. Dr. R. Paul of the Dept of Defense provided valuable comments on surety engineering ideas. Last but not the least Dr. T. Kwon, his colleague at the University of California, Berkeley provided invaluable advice and comments in the overall organization. The writer expresses his deepest appreciation to these friends.

References

[1] D.S. Davis, "J. Boskin, 'The coming of knowledge based business'," Harvard Business Review, Sept.–Oct. 1994.
[2] K. Weick, "Ross Ashby's principle of requisite variety," in The Social Psychology of Organizing, Addison Wesley, Reading, MA, 1979. (also, R. Cooper, "Debunking the myths of new product development," Research Technology Management, July/Aug. 1994, 1999.)
[3] A. Grove, "Only the paranoid survive," Doubleday, April 1999.
[4] T. Dellin, "Confidence by design," Sandia Technology, Sandia National Labs, vol.1, no.1. Winter 1999. Also M.C. Paulk, B. Curtis and M.B. Chrisis, "Capability maturity model for software Version 1.1," Software Engineering Institute CMU/SEI-93-TR, Feb. 24, 1993.
[5] G. Held, Data Communication Networking Devices, John Wiley & Sons, 1986.
[6] T. Petzinger, Jr., "A model for the nature of business. It is alive," Wall Street Journal, Feb. 2nd, 1999.
[7] P.M. Senge, The Fifth Discipline: The Art and Practice of Learning Organization, Bantam Doubleday Dell Publishing

Group, 1990.
[8] R. Poole, "When failure is not an option," IEEE Engineering Management Review, Spring 1999.
[9] D.M. Amidon, Innovation Strategy for the Knowledge Economy, The Ken Awakening, Butterwoth-Heinemann, 1997.

Chitoor V. Ramamoorthy earned his two undergraduate degrees in physics and technology from the University of Madras, India, two graduate degrees in mechanical engineering from the University of California, Berkeley and a Master's and Ph.D. from Harvard University in Applied Mathematics (Computer Science) in 1964. His education was supported by Honeywell Inc.'s Computer Division, Waltham, MA with whom he was associated till 1967, last as Senior Staff Scientist. He then joined the University of Texas, Austin as a Professor in Electrical Engineering and Computer Sciences. After serving as Chairman of the Compute Science Dept., he joined the University of California, Berkeley as Professor of Electrical Engineering and Computer Sciences in 1972, a position that he still holds. He has supervised 73 Ph.D. students, who include Vice Chancellor of University of Texas System, Deans, Dept Chairmen, Chair Professors, the CEO of Lucent Technology's largest subsidiary and including, most recently, the President Elect of the IEEE Computer Society. He has held the Control Data Distinguished Professorship at the University of Minnesota and the Grace Hopper Chair at the U.S. Naval Postgraduate School, Monterey, California. He was also a Visiting Professor at the Northwestern University and Visiting Research Professor at the University of Illinois, Urbana-Champaign. He is a Senior Research Fellow at the ICC Institute of the University of Texas, Austin. He has received the IEEE Computer Society's Group Award in Education, the Taylor Booth Award for Education, the Richard Merwin Award for Outstanding Professional Contributions, the Golden Core Award and the IEEE Centennial Medal. He is a Fellow of IEEE and of the Society of Design and Process Sciences, from which he received he R.T. YEH Distinguished Achievement Award in 1997. He also received a Best Paper Award from the IEEE Computer Society in 1987. Three international conferences were organized in his honor as well one UC Berkeley Graduate Student Research Award and two International Conference/Society Awards have been established in his name. He served as the Editor in Chief of the IEEE Transactions of Software Engineering, and the founding Editor in Chief of the IEEE Transactions of Knowledge and Data Engineering. He is also founding Co-Editor in Chief of the International Journal of Systems Integration and of the Journal of the Society of Design and Process Sciences. He served in several capacities in the IEEE Computer Society including the First Vice President and the Governing Board Member. He served on several Advisory Boards of the Federal Government and Academia including U.S. Army, Navy, Air Force, the DOE'S Los Alamos Lab, the Universities of Texas, California, Toronto, and State University System of Florida. He is one of the founding directors of the International Institute of Systems Integration in Campinas, Brazil, supported by the Federal Government of Brazil and for several years, a member of International Board of Advisors, of the Institute of Systems Science of the National University of Singapore. He has published more than 150 papers and co-edited three books. He has worked on and holds patents in computer architecture, software engineering, computer testing and diagnosis and data bases. He is currently involved in the research on models and methods to assess the evolutionary trends in information technology.