Automation Handbook

Roxie Corley

First Edition, 2012

ISBN 978-81-323-3514-6

Published by:
University Publications
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter- 1

Introduction to Automation

KUKA Industrial Robots being used at a bakery for food production

Automation is the use of control systems and information technologies to reduce the need for human work in the production of goods and services. In the scope of industrialization, automation is a step beyond mechanization. Whereas mechanization provided human operators with machinery to assist them with the muscular requirements of work, automation greatly decreases the need for human sensory and mental requirements as well. Automation plays an increasingly important role in the world economy and in daily experience.

Automation has had a notable impact in a wide range of industries beyond manufacturing (where it began). Once-ubiquitous telephone operators have been replaced largely by automated telephone switchboards and answering machines. Medical processes such as

primary screening in electrocardiography or radiography and laboratory analysis of human genes, sera, cells, and tissues are carried out at much greater speed and accuracy by automated systems. Automated teller machines have reduced the need for bank visits to obtain cash and carry out transactions. In general, automation has been responsible for the shift in the world economy from industrial jobs to service jobs in the 20th and 21st centuries.

Advantages and disadvantages

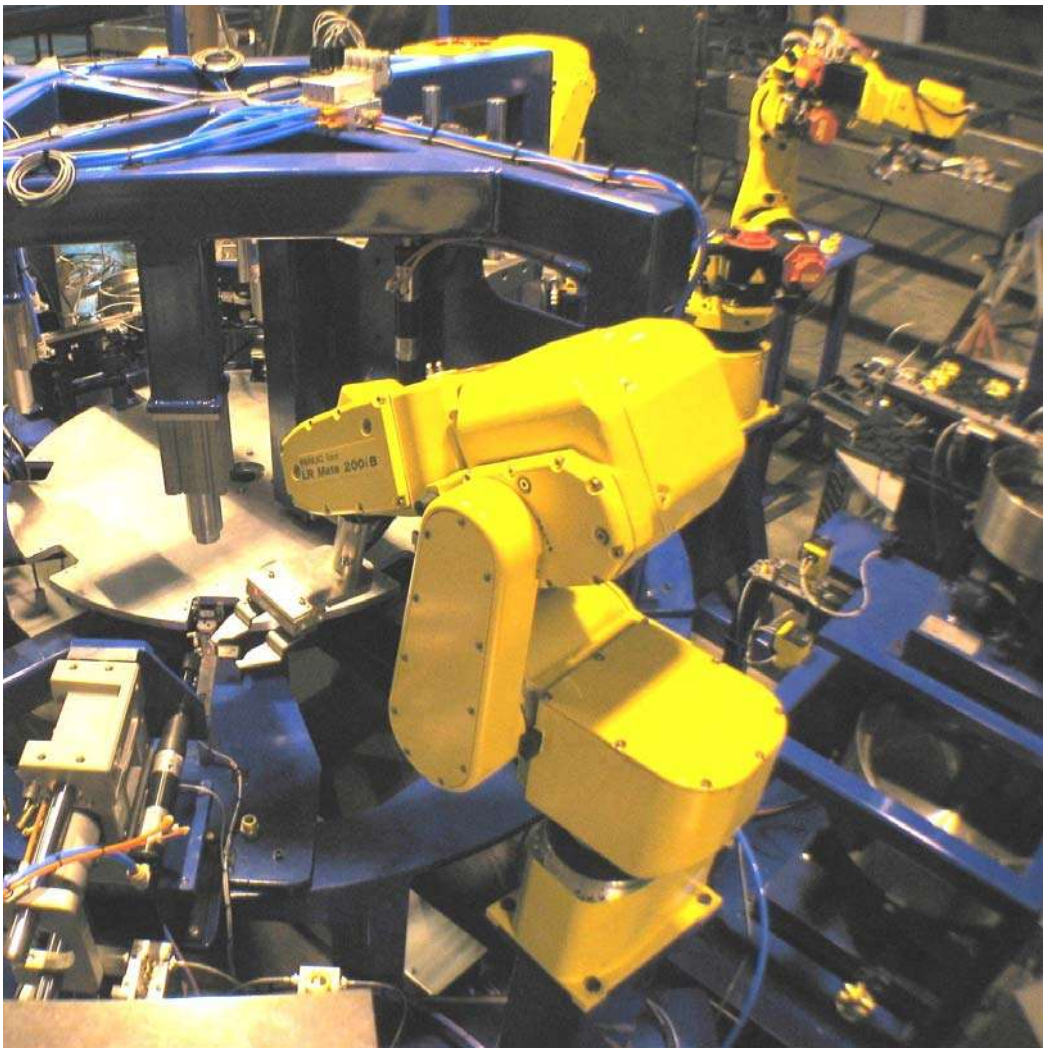
The main advantages of automation are:

- Replacing human operators in tasks that involve hard physical or monotonous work.
- Replacing humans in tasks done in dangerous environments (i.e. fire, space, volcanoes, nuclear facilities, underwater, etc.)
- Performing tasks that are beyond human capabilities of size, weight, speed, endurance, etc.
- Economy improvement. Automation may improve in economy of enterprises, society or most of humanity. For example, when an enterprise invests in automation, technology recovers its investment; or when a state or country increases its income due to automation like Germany or Japan in the 20th Century.

The main disadvantages of automation are:

- Technology limits. Current technology is unable to automate all the desired tasks.
- Unpredictable development costs. The research and development cost of automating a process may exceed the cost saved by the automation itself.

- High initial cost. The automation of a new product or plant requires a huge initial investment in comparison with the unit cost of the product, although the cost of automation is spread in many product batches.



Relationship to unemployment

Multivariate effect

Most people consider it common sense that automation has the potential to foster unemployment, because it obviates human work by transferring tasks to machines. However, the translation of that potential into observed effect has largely not happened in the two centuries during which it has been continually predicted. After many decades of automation development and dissemination, the net macroeconomic effect has been generally positive—automation has been part of a general trend of economic growth worldwide; standards of living have risen in many places; and automation has never yet been shown to have induced any widespread structural unemployment. The main

explanation for this is that, so far, job losses in any one particular economic niche have always been more than offset by job gains in other niches. As the lowered unit cost of goods and services (which the automation made possible) gave consumers more purchasing power to devote to other goods and services, new jobs sprang up in the production of those goods and services. Thus each time that automation has freed up human resources, those resources have been redeployed by market forces (although it did not always happen without turbulence in the lives of individual workers).

One of the earliest promises of automation was to allow more free time, without any threat of income reduction. This effect has been seen in many individual facets of life (for example, the automatic washing machine has made laundry less time-consuming; engine control units have reduced the amount of automotive downtime; the automatic dishwasher has made dishwashing less time-consuming), but the net outcome of modern life in developed economies remains a state of hurry and busyness, mostly because rising living standards have brought rising expectations in direct relation. (Each time-saving improvement has made room for a new aspiration to take its place.)

Automation also does not imply unemployment when it makes possible tasks that were unimaginable without it (such as exploring Mars with the Sojourner rover). Likewise

with fields where the economy is already fully adapted to an automated technology, and the jobs were lost long enough ago that the displacement was long since absorbed by the workforce (as with the continually advancing automation of the telephone switchboard, which eliminated most telephone operator jobs and kept many more from ever existing in the first place).

Today automation is quite advanced (relative to just a few lifetimes ago), and it continues to advance with an accelerating pace throughout the world. Although it has been encroaching on ever more skilled jobs, the general well-being and quality of life of most people in the world (where political factors have not muddied the picture) have improved. Clearly a multivariate effect has been at work (something much more than just the obvious idea that automation has the *potential* to cause unemployment). In fact, the idea that automation posed an *imminent* threat to employment, first articulated in 1811 by a group of textile workers known as Luddites, has proven to be so fallacious over the ensuing two centuries that economists call the imminent-threat idea the *Luddite fallacy*. Today the eternal fallaciousness of the Luddite premise is a mostly undisputed principle of economic theory, because it has proven true empirically time and again.

There is some concern today that the economy's ability to continue absorbing ever-increasing automation without experiencing significant structural unemployment may be heading toward an upper limit—that is, that we are approaching a point where the Luddite premise will no longer be entirely fallacious, because the relationship of humans to machines that made it fallacious is changing. In this view, the empirical strength of the eternal-fallaciousness idea is only a reflection of the parameter values of the environment thus far. In other words, the idea is undoubtedly an excellent explanation of the past, but whether it can accurately predict the future is an independent problem. Like an investment prospectus, proponents of this view caution that "past performance is no guarantee of future results."

Timeline of concerns about automation's relationship to unemployment

Early in the Industrial Revolution

Historical concerns about the effects of automation date back to the very beginning of the Industrial Revolution, when a social movement of English textile machine operators in the early 19th century known as the Luddites protested against Jacquard's automated weaving looms. The Luddites destroyed a number of these machines, which they felt threatened their jobs.

Later in the Industrial Revolution

The development of the American system of manufacturing disgusted many skilled machinists at a time when the very definition of being a machinist included a core element of skilled toolmaking and fitting on a craft basis. Innovations of this system included increasing reliance on jigs and gauges and on machine tools that built more of a process into the tool's movements (such as turret lathes and screw machines). These

innovations continually turned skilled work into semi-skilled or unskilled, contributing to vast migrations of laborers across borders and oceans. However, despite this transformation, there were always other economic niches for skilled workers to go to, given enough searching. Recessions interfered with employment, but no foundational aspects of structural unemployment were caused by automation itself.

During the Machine Age

As in the preceding century, the period of 1880 to 1940 saw no underlying automation-induced structural lack of new economic opportunities for skilled workers to go to, given enough searching, although the Great Depression caused a tremendous disruption to employment. The foundational *potential* for full employment had not been lost, as would later be shown by the post–World War II economic expansion and other economic miracles.

During the 1950s through 1990s

The postwar development of new automation technologies using electronics, servomechanisms, and digital computers stoked a new wave of fears similar to the old

Luddite ones. Among the working class and labor unions, there was stiff resistance to loss of employment through automation, including contract clauses won in hard-fought contract negotiations that mandated alternate employment for any workers whose positions were eliminated by automation. These clauses seemed a great victory for union workers at large corporations in developed nations, but because they had no effect at smaller, nonunionized companies or in developing nations, those corporations faced withering competition that shrank their market shares until their workers' gains eventually undermined their own success. However, the salvation for employment rates damaged in the industrial sector (secondary sector of the economy) came from the service sector (tertiary sector), which absorbed all of the workers that automation displaced elsewhere. For example, many manufacturing jobs left the United States during the 1990s but were offset by a one-time massive increase in IT jobs at the same time. And in some cases the freeing up of the labor force allowed more people to enter higher skilled managerial jobs and technically specialized jobs, which are typically higher paying. Therefore, fears of unemployment due to automation were generally dismissed as just another instance of the Luddite premise, which had proven fallacious time and again over many decades. Given this obvious empirical contradiction of the premise, people who nevertheless returned to it were usually viewed by the mainstream as cranks misled by quixotic leftist political bias. For example, works by scholars including David F. Noble and Jeremy Rifkin were often respected but discounted. At worst, they were mocked with the disparaging label "neo-Luddite". Noble even wrote a later book titled *Progress Without People: In Defence of Luddism* to try to further explain why the Luddite premise should not be laughed out of academia.

Post-market musings

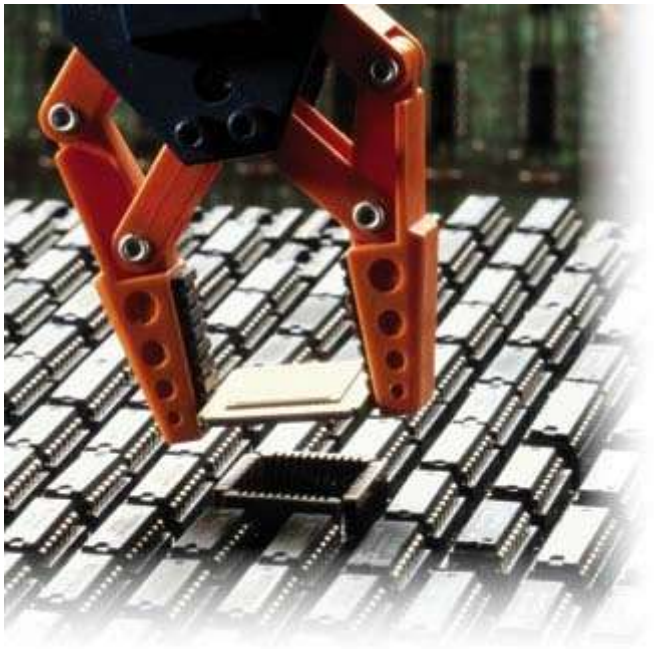
Rifkin's *End of Work*, published in 1995 and written by a non-engineer, predicted automation-induced unemployment despite having a rather hazy idea of how IT would evolve over the next decade. (The book mentioned the Internet once in passing and the

World Wide Web not at all. Its IT focus was mostly on robotics.) Also hazy was Rifkin's explanation of any solution to the problem. The book's subtitle called the solution a "post-market economy", but its concluding chapters did not clearly lay out how such an economy could be engineered, leaving readers to conclude that a non-market solution involving a planned economy was implied between the lines.

In terms of political economy implications, there was no clear differentiation at the time between the ideas of authors like Noble or Rifkin (on the one hand) and traditional leftist agitation (on the other hand). To the extent that readers could ask "What point is this guy getting to?" and answer the question with "socialism" or "a welfare state", they dismissed these authors.

During the 2000s and 2010s

Since the 1990s, the possibility has been raised again in even an apolitical, technocratic way that the Luddite premise (that automation creates unemployment) was only fallacious in the absence of highly advanced and ubiquitous automation, which until recently was mostly out of reach technologically. This would explain why it has always been fallacious until now, but also why it might not always remain so. For example, Marshall Brain, Martin Ford, and others have suggested that exponentially accelerating information technology (IT) may ultimately result in widespread structural unemployment, because an implicit assumption underlying the "eternally fallacious" idea (that lots of regular humans will always find ways to do service work that machines can't do) will itself be fallacious as IT advances. They suggest that, unlike in the 20th century, when the tertiary sector absorbed all of the workers that the automation of the secondary sector expelled, the tertiary sector now also faces depopulation via automation; its employment will shrink, not grow, and this time there is no other sector to backstop the process by absorbing the displaced workers. The high unemployment rates of the late-2000s recession have brought the idea of structural unemployment back into mainstream attention, as observations are made about positions that require extensive specialized skill and experience standing long vacant even while general unemployment rates above 9% (and horror stories of fruitless job searches) would seem to suggest that such vacancies ought to be scarcer. The idea that automation has finally advanced to the point that the Luddite premise is no longer entirely fallacious is one of the components of some theoretical explanations for the string of jobless recoveries in developed economies in recent decades. Expectations that the (already eroding) fallaciousness will fall off sharply in coming decades underlie the fear of structural shift.



Writers such as Rifkin, Brain, and Ford often suggest that the structure of the economy will have to shift to a basic income because its present structural foundation (trading labor for income) will no longer be an available option. They often include an element of civic obligation, such that able people must somehow contribute civically in order to receive the basic income. The labor-market economy (trading labor for income) already achieves that outcome today (because working for income generally produces civic value in various ways, directly and indirectly), but the argument is that advanced automation will decouple the linkage that makes that possible. Thus the same result (trading civic value for income) would have to be driven by different forces—either non-market ones, or via a new kind of market. The non-market idea seems infeasible given the generally abysmal performance record of planned economies. But the idea of engineered new markets leaves room for the disciplining and motivating powers that make capitalist markets capable of positively shaping human behavior where government alone is usually unable.

New-market engineering

Brain and Ford's books, in stark contrast to Rifkin's, came later and were written by engineers with extensive under-the-hood knowledge of modern production methods, computer hardware and software, and the Internet's underpinnings. They explicitly reject non-market solutions as unworkable and instead suggest new kinds of markets. Rather than being "post-market" proponents, such authors could be called "new-market" proponents. They vigorously distance themselves from socialism or welfare states—generally seeking to keep a market economy with private enterprise, which they believe cannot be preserved *unless* its foundation is modified from its current structure. Thus, quite contrary to being anti-market agents (as critics might suppose them to be), they believe themselves to be *salvaging* markets from destruction. They envision creating consumer purchasing power by some other mechanism than the traditional labor market

as we have known it so far, in order that free markets may continue to provide the invisible hand component of production-possibilities decisions. In other words, they believe that market forces are necessary to generate allocative efficiency, and they believe that without a structural modification that (at least partially) decouples purchasing power (and consumer confidence) from employment determined by the traditional labor market, there will be a systemic market failure, which they seek to avoid.

Just as new-market advocates are pro-market and pro-private-property, they are also very much non-Luddite (in fact, exactly opposite of Luddite) in the respect that they *like* technology—they don't *hate* it. They want it to continue advancing as robustly as ever. They simply feel that income and purchasing power must be decoupled from human participation in production. (The decoupling does not have to happen all at once; it could start small and gradually increase.) If that happens, then they essentially do not have any problem with technology or automation, per se. In contrast to old-style welfare, they do not feel that income should be unconditional, or equal, or "free" (given out "for nothing"). They believe that people should have to work for it (in a new sense of the word "work"), in the respect that they are given incentives to do positive things, like take classes, read books, conserve environmental resources, and so on. People would be paid to do civically valuable things, and if they chose not to do those things, they would not be paid. In this way, new-market advocates align themselves with human nature, which generally requires selfish motivations and incentives to shape behavior, and with the market's invisible hand, which is needed to make the right production-possibilities decisions (because the idea that individual human managers, or groups of them, are capable of making those decisions correctly with zero invisible-hand assistance has been empirically discredited).

Wage-recapture market variant

Ford's main new-market mechanism would be to create a tax that recaptures most (not all) of the value that firms and their customers gain from eliminating wages, then use the tax revenue to pay people for doing civically valuable actions—that is, pursuing activities, such as higher education or environmental preservation, that have positive externalities. The main reason for paying these "wages" need not be their altruistic or environmentalist components; the main reason is simply to prevent the market economy from collapsing due to noncirculation of value (that is, the lack of adequate trade which

would occur if lack of consumer purchasing power and confidence left no way for an adequate mass market to exist). Ford points out that the tax could not take *all* of the gains away from the corporations and their customers, because this would destroy the natural incentive to innovate that a market economy needs to be sustainable. The value would be split between the innovators, their customers, and the rest of the population, because leaving out any of that trio would wreck the sustainability of the model. (The leaving out of the third leg is what is causing today's economic pathologies and promising tomorrow's, in the view of new-market engineers.) Ford's idea is an earnest market effort because it preserves the invisible hand as the maker of production-possibilities decisions for goods and services. However, it does rely on human planning (via a technocratic government agency in each country) to make the production-possibilities decisions for civic actions. The latter is viewed as unfortunate but necessary due to the lack of an alternative.

Mirror-image market variant

Another idea for a new-market mainspring which solves the aforementioned "lack of an alternative" problem is a "mirror image" idea, which has an even more private-sector approach in which the invisible hand helps make even the civic-actions production-possibilities decisions. In this model, the government does not collect a wage-recapture tax at all. Instead of enforcing tax payment, it only enforces payment of a new-style "wage" directly from corporations to consumers that looks to us today like something we might label "mandatory philanthropy", but which would actually be a true market wage of a new type. In today's old market, money flows from consumers, through (partially automated) companies, past the eyes of the government enforcement sentry (but not through its hands) as wages, into the hands of workers (who are also the consumers, thus completing the cycle of value recirculation). In the new market (mirror-image variant), money would flow from consumers, through (highly automated) companies, past the eyes of the government enforcement sentry (but not through its hands) as [new-style] "wages", and into the hands of [new-style] "wage" earners, who are paid the "wage" for civically valuable actions. (They are also the consumers, thus completing the cycle of value recirculation).

In this model, the decisions about what the civic actions are can be made by the invisible hand, because each "mandated philanthropist" gets a large degree of authority in what actions their "philanthropy" (which is actually [functionally] a new-style "payroll") will or won't pay for. Many such paymasters functioning simultaneously could constitute the "buyers" in a market for civically valuable actions (with mass-market "workers" as the "sellers"). There would still be *some* regulation involved, because, for example, it would be illegal to base the "payroll" decisions on race, color, religion, creed, gender, sexual orientation, ethnicity, disability, marital or veteran status, and so forth. To decide which "workers" were on a given "payroll", there might be a clearinghouse to randomly match the two, rotating assignments every several years. Or perhaps the businesses that run the "payroll" could even "hire" the "workers" themselves, in which case workers would compete for "jobs" by showing off how "productive" they could be in doing the civic actions (another level of invisible hand yet again). The "mirror image" name comes from the idea that this variant of new market is a very free market where the invisible hand remains just as powerful as it was in the 1945-2008 economy, but with many mirror-

image aspects (which are visualized above by the amount of quotation marks that are necessary to signify mirror-image senses for words that were always [up till now] widely known only in their non-mirror-image senses).

The axis of reflection in the mirroring seems to be, at root, a "polarity shift" from where human individuals can add value only by *doing production* (from within production systems) to where they can also add value by *avoiding hurting production systems* (from outside). The hurt-avoidance comes from such civic actions as providing goods-and-services demand via consumption (which the system requires in order to stay running) instead of failing to consume (because of lack of income); ensuring the sustainable supply of energy and environmental resources to the production systems (by avoiding *over*consuming those); and by ensuring the supply of people educated enough to provide the few humans that the production systems will need in the future, by pursuing education and cognitively enriching pastimes. The humans that the systems need will be few, but those few will need to be highly intelligent, talented, and educated, constituting a human resource that might be endangered if the general population does not act as a "farm team system" for it by valuing education and self-education as a civic action. An analogy is provided by sports' relationship to general life. Few humans are talented and practiced enough to play professional sports, but the professional teams rely on a system that filters such scarce people out of the general population via little league/pee wee programs, high school play, college play, farm-team play, etc. People in the general population are not considered inferior human beings (versus the pro players) because of their lack of pro talent. They are valued as the fans and ticket-buyers that make the pro system economically viable. And a small fraction of them grow up to become pros themselves.

In today's old market, governments enforce the payment of wages by having outlawed their nonpayment (i.e., slavery); by levying tariffs on cheap competition from countries that kept their nonpayment (slavery) (that outlawing has now been global for many decades); and by attempting to minimize their underpayment (i.e., wage slavery, a sharply cheapened value of work [with elites and their customers keeping the money]). In the new market (mirror-image variant), governments enforce the payment of "wages" by outlawing their nonpayment (i.e., evading the "payroll"); by levying tariffs on cheap competition from countries that kept their nonpayment (non-participating countries); and by attempting to minimize their underpayment (i.e., "wage" slavery, a sharply cheapened value of civic actions [with elites and their customers keeping the money]).

One of the inherent challenges of the mirror-image variant is that various forms of dressing up corporations' financial self-interest in a specious cloak of civic virtue would inevitably arise. This would be a "washing" form of marketing and operations that included greenwashing and analogous washing in other domains of life (e.g., education, infrastructure). It seems unlikely that this can be entirely negated; instead, it would have to be perennially pruned by social censure and regulatory oversight. However, no other system is without its chronic weaknesses, either. For example, the classical variants of capitalism (implemented thus far) have scored poorly on various tests, such as environmental sustainability and (potentially) the employability of the average human (as that was traditionally defined) as automation grows pervasive. Twentieth-century variants of communism fared even worse in environmental sustainability, and also failed economically in average standard of living and politically in individual freedom. The wage-recapture new-market variant, with its technocratic decisions on how to spend the revenue, holds promise to minimize the corporate "washing" problem, yet it also holds risks of failing on allocative efficiency and market-driven innovation, which the mirror-image variant mitigates. As elsewhere in reality, each choice has pros and cons, rather than any choice being perfect. The "washing" problem may be the mirror-image analog of classical capitalism's tendency to exaggerate needs (for example, a maker of antibacterial soaps encouraging the populace to fear microbes to an irrational degree). Both are forms of conflict of interest that cause "chronic irritation" to a socioeconomic system but need not be "fatal" to it if given adequate "medical management". The washing problem may be less systemically injurious than the allocative inefficiency problem, just as the "exaggerated needs" problem of classical capitalism was less systemically injurious than the allocative inefficiency problem of central economic planning.

In choosing the decider of production-possibilities decisions (whether of goods, services, or civic actions), the invisible hand is generally preferred to committees of humans because it has proven to be superior at the decision making (except for regulatory issues such as race-color-religion-etc and the "washing" discussed above). In the future it will also be necessary to ask what role artificial intelligence might possibly have in making those decisions, and whether humans would allow it. Perhaps artificial intelligence, like human intelligence, will share the role with the invisible hand but be barred from usurping the entirety of it.

Implementations

Regarding the chances of any new-market ideas being implemented, there are both significant barriers and significant drivers, with a net potential of perhaps "even chances". Ford discusses many of these barriers and drivers. The barrier side includes (a) natural cultural conservatism that powerfully resists systemic changes; (b) the powerful influence of laissez-faire ideals, which would resist any engineered systemic change to markets (especially *anything* involving a tax); (c) the fact that early implementation by individual countries faces an immediate threat from the export and offshoring competition of countries that *haven't* yet implemented; and (relatedly) (d) the all-or-nothing problem, which may occur if a new system would work well but only if the switch from old to new was an off-on switching rather than a gradual evolution. However, on the driver side there are powerful forces that may answer all of the barriers. Foremost would be a dawning realization by economic elites that they have a choice between a new market with prosperity, or the old market spiraling into near-total failure. Globalization so far has not threatened the wallets of economic elites (only those of average workers), and has in fact enriched the elites thus far; but the changing parameter values of the economic system as automation advances would alter that runtime environment and transform it into a new one, where even the elites' wealth would be threatened by a market failure that killed their businesses and reduced asset values throughout the economy. Realizing these options, elites might actually switch from opposing new markets to actively supporting their implementation (including addressing the competition between countries whose policies differed). The all-or-nothing problem does not have to occur if an implementation is engineered such that extremely profitable, extremely automated industries began piloting new markets while other industries continued with an old-market status quo for quite some time. In this model, the early adopters voluntarily become leaders, and the pilot projects would act simply as economic stimulus on the broader economy (although a type of stimulus much more effective than old-style stimulus, whose efficacy seems to be eroding because it relies on the Luddite premise being a total fallacy as opposed to shifting by degrees out of total fallaciousness). The overall transition in this model could actually be quite painless, as a generally prosperous economy changed gradually over decades from mostly-old-with-some-new to mostly-new-with-some-old. The selfish motivation of the early-adopter leaders would be the aforementioned choice faced by economic elites. They would choose to stimulate the broader economy because that result would ensure their own continuing strong sales and growth by preserving a runtime environment of general prosperity for them to operate within, without which depression or malaise would occur.

Given the aforementioned choice faced by economic elites, those in the private sector might even choose to pursue the new market without government involvement. But the private sector faces two hurdles that would make it difficult: the natural competition between firms (which is necessary and thus must be protected by competition law), and legal obligation to maximize shareholder value. The traditional definitions of shareholder value evolved in an earlier era whose commercial environment had different parameter values due to lack of advanced automation. Those traditional definitions would bar new-style payrolls. But in the face of market failure without them, perhaps a case would emerge for an updated definition. Competition is the other hurdle. Companies are barred by competition law from agreeing to limit competition, and even if they weren't, individual companies generally cannot make the first move of increasing expense without being killed by competition from rivals who don't. This is why "a level playing field" would have to be created by policy, or to use a different analogy, "a high tide that lifted all boats equally". This is directly analogous to existing minimum wage laws. Individual companies generally could not survive in the market if they volunteered to self-enforce minimum limits on wages (in the absence of any laws requiring them). There *is* breathing room for above-market wages (e.g., to attract superior talent) at some companies in some industries who enjoy a relatively high level of imperfect competition; but most companies in most industries face competition too close to pure to survive the attempt. In this sense, the mandated value recirculation (whatever anyone calls it, from "wage recapture" to "new-style wages") is as unremarkable and non-novel an idea as any legislative or regulatory mechanism in commerce. For goals that make long-term systemic balance possible but cannot be pursued by the self-interest of individual market players, these mechanisms provide a path by forcing all competitors to play the game by the same rules. Existing examples include employment standards (e.g., child labor laws, minimum wage laws), environmental protection, and financial regulation (to prevent bubbles and thus crashes). These exist in perennial tension with the forces of pure capitalism; thus the extremes perform checks and balances on each other. Businesses usually fight for inadequate regulation; government usually fights for excessive regulation; and a sustainable balance results. Over decades, systemic pathologies gradually push the balance point out of the sustainable range; periodic breakdowns then yield correction by counteractive forces (e.g., trust-busting [leftward correction], the Reagan revolution [rightward correction]).

Laissez-faire ideals reigned supreme worldwide for about three decades (roughly centered on the fall of the Soviet Bloc, which vindicated capitalism over central planning in many ways). In this environment, where the lesson commonly extrapolated was that pure capitalism will always be better than any mixed-economy alternatives, the prevailing theory has been that higher corporate taxes can only harm economic prosperity. The reasoning is partly that countries can simply compete to undercut each other's corporate tax rates (which is true), but also, more importantly, that only the invisible hand is capable of recirculating capital back toward the base of the economy in a successful manner (which is not to be dismissed lightly, and may in fact be true). The disparaging label for such ideas is "trickle-down economics", but many intelligent people have earnestly believed in these ideals; and the fact that their discounting has often been facile and done by imperfect opponents has only encouraged believers to stay faithful.

Widespread fervor for trickle-down beliefs (in both the public and private sectors) poses a formidable barrier to the wage-recapture-tax new-market variant. But these conventional beliefs rely on the assumption that the Luddite premise is entirely and eternally fallacious. Unfortunately, there has already been a decade of empirical evidence that low taxes, new business investment, and economic growth no longer have a sure-fire correlation to strong, "good-jobs" employment in developed economies. If the Luddite premise has been starting to shift into partial accuracy, then no amount of continued low taxes and deregulation will ever be able to produce enough trickling down to create broad-based prosperity. In that case, mandating the payment of new-style wages could recirculate value back to the base of the mass market. The promise of the mirror-image variant would be that humans need not turn to central planning for the distribution details, because as long as the "minimum wage" (referring to the new-style wages) and other employment standards are being enforced, then government's role ends there.

Other goals of automation (beyond productivity gains and cost reduction)

In manufacturing, the purpose of automation has shifted to issues broader than productivity and costs.

Reliability and precision

The old focus on using automation simply to increase productivity and reduce costs was seen to be short-sighted, because it is also necessary to provide a skilled workforce who can make repairs and manage the machinery. Moreover, the initial costs of automation were high and often could not be recovered by the time entirely new manufacturing processes replaced the old. (Japan's "robot junkyards" were once world famous in the manufacturing industry.)

Automation is now often applied primarily to increase quality in the manufacturing process, where automation can increase quality substantially. For example, automobile and truck pistons used to be installed into engines manually. This is rapidly being transitioned to automated machine installation, because the error rate for manual installment was around 1-1.5%, but has been reduced to 0.00001% with automation.

Health and environment

The costs of automation to the environment are different depending on the technology, product or engine automated. There are automated engines that consume more energy resources from the Earth in comparison with previous engines and those that do the opposite too. Hazardous operations, such as oil refining, the manufacturing of industrial chemicals, and all forms of metal working, were always early contenders for automation.

Convertibility and turnaround time

Another major shift in automation is the increased demand for flexibility and convertibility in manufacturing processes. Manufacturers are increasingly demanding the ability to easily switch from manufacturing Product A to manufacturing Product B without having to completely rebuild the production lines. Flexibility and distributed processes have led to the introduction of Automated Guided Vehicles with Natural Features Navigation.

Digital electronics helped too. Former analogue-based instrumentation was replaced by digital equivalents which can be more accurate and flexible, and offer greater scope for more sophisticated configuration, parametrization and operation. This was accompanied by the fieldbus revolution which provided a networked (i.e. a single cable) means of communicating between control systems and field level instrumentation, eliminating hard-wiring.

Discrete manufacturing plants adopted these technologies fast. The more conservative process industries with their longer plant life cycles have been slower to adopt and analogue-based measurement and control still dominates. The growing use of Industrial Ethernet on the factory floor is pushing these trends still further, enabling manufacturing plants to be integrated more tightly within the enterprise, via the internet if necessary. Global competition has also increased demand for Reconfigurable Manufacturing Systems.

Automation tools

Engineers now can have numerical control over automated devices. The result has been a rapidly expanding range of applications and human activities. Computer-aided technologies (or CAx) now serve the basis for mathematical and organizational tools used to create complex systems. Notable examples of CAx include Computer-aided design (CAD software) and Computer-aided manufacturing (CAM software). The improved design, analysis, and manufacture of products enabled by CAx has been beneficial for industry.

Information technology, together with industrial machinery and processes, can assist in the design, implementation, and monitoring of control systems. One example of an industrial control system is a programmable logic controller (PLC). PLCs are specialized hardened computers which are frequently used to synchronize the flow of inputs from (physical) sensors and events with the flow of outputs to actuators and events.

Human-machine interfaces (HMI) or computer human interfaces (CHI), formerly known as *man-machine interfaces*, are usually employed to communicate with PLCs and other computers. Service personnel who monitor and control through HMIs can be called by different names. In industrial process and manufacturing environments, they are called operators or something similar. In boiler houses and central utilities departments they are called stationary engineers.

Different types of automation tools exist:

- ANN - Artificial neural network
- DCS - Distributed Control System
- HMI - Human Machine Interface
- SCADA - Supervisory Control and Data Acquisition
- PLC - Programmable Logic Controller
- PAC - Programmable automation controller
- Instrumentation
- Motion control
- Robotics

Current limits

Many roles for humans in industrial processes presently lie beyond the scope of automation. Human-level pattern recognition, language recognition, and language production ability are well beyond the capabilities of modern mechanical and computer systems. Tasks requiring subjective assessment or synthesis of complex sensory data, such as scents and sounds, as well as high-level tasks such as strategic planning, currently require human expertise. In many cases, the use of humans is more cost-effective than mechanical approaches even where automation of industrial tasks is possible.

Applications of Automation

- **Automated Video surveillance:**

The Defense Advanced Research Projects Agency (DARPA) started the research and development of automated Visual surveillance and Monitoring (VSAM) program 1997-99 and airborne Video Surveillance (AVS) program 1998-2002. Currently there is a major effort underway in the vision community to develop a fully automated tracking surveillance system. Automated video surveillance monitors people and vehicle in real time within a busy environment. Existing automated surveillance systems are based on the environment they are primarily designed to observe, i.e., indoor, outdoor or airborne, the amount of sensors that the automated system can handle and the mobility of sensor, i.e., stationary camera vs. mobile camera. The purpose of a surveillance system is to record properties and trajectories of objects in a given area, generate warnings or notify designated authority in case of occurrence of particular events.

- **Automated Highway Systems:**

As demands for safety and mobility have grown and technological possibilities have multiplied, interest in automation have grown. Seeking to accelerate the development and introduction of fully automated vehicles and highways, The United States Congress authorized more than $650 million over 6 years for intelligent transport systems (ITS) and demonstration projects in the 1991 Intermodal Surface Transportation Efficiency Act (ISTEA). Congress legislated in ISTEA that “The secretary [of transportation] shall develop an automated highway and vehicle prototype from which future fully automated intelligent vehicle-highway systems can be developed. Such development shall include

research in human factors to ensure the success of the man-machine relationship. The goal of this program is to have the first fully automated highway roadway or an automated test track in operation by 1997. This system shall accommodate installation of equipment in new and existing motor vehicles." [ISTEA 1991, part B, Section 6054(b)].

Full automation commonly defined as requiring no control or very limited control by the driver; such automation would be accomplished through a combination of sensor, computer, and communications systems in vehicles and along the roadway. Fully automated driving would, in theory, allow closer vehicle spacing and higher speeds, which could enhance traffic capacity in places where additional road building is physically impossible, politically unacceptable, or prohibitively expensive. Automated controls also might enhance road safety by reducing the opportunity for driver error, which causes a large share of motor vehicle crashes. Other potential benefits include improved air quality (as a result of more-efficient traffic flows), increased fuel economy, and spin-off technologies generated during research and development related to automated highway systems.

- **Automated manufacturing:**

Automated manufacturing refers to the application of automation to produce things in the factory way. Most of the advantages of the automation technology has its influence in the manufacture processes.

The main advantage of the automated manufacturing are: higher consistency and quality, reduce the lead times, simplification of production, reduce handling, improve work flow and increase the morale of workers when a good implementation of the automation is made.

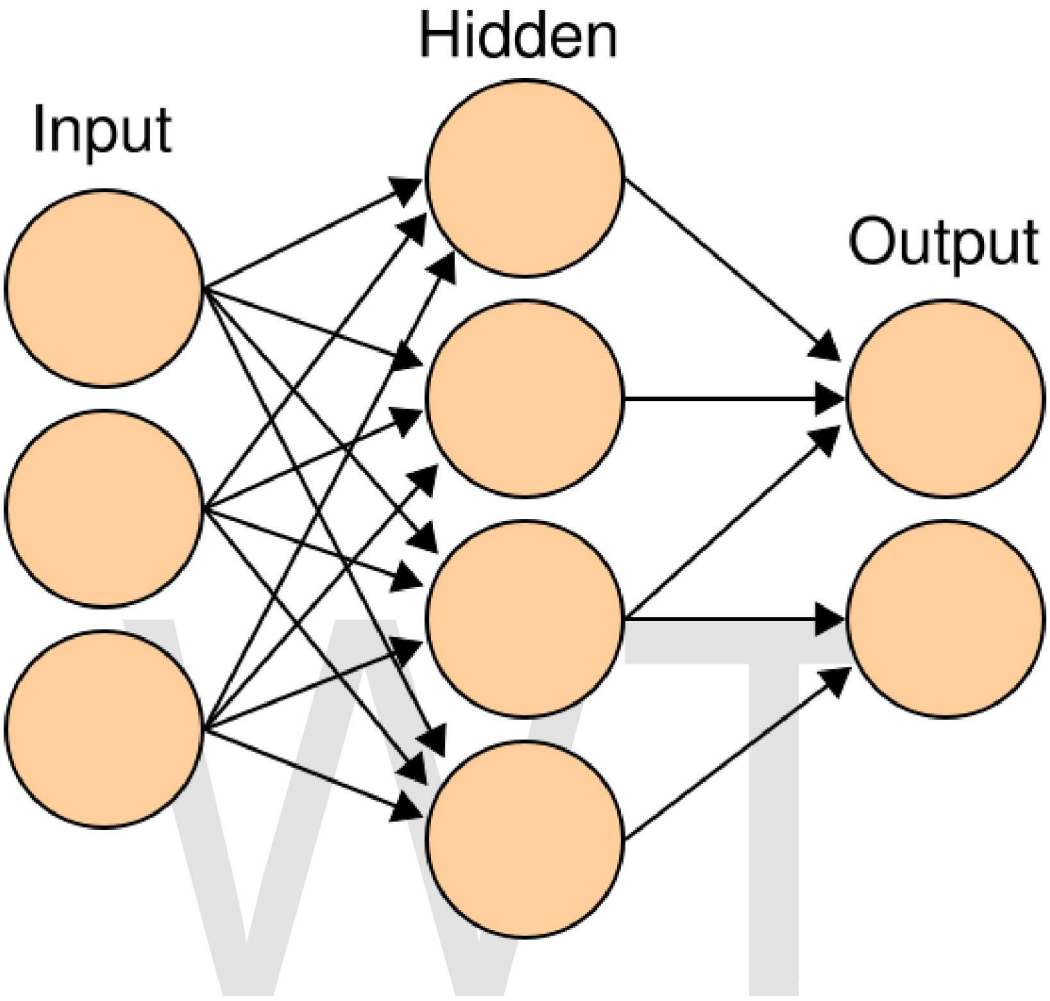
- **Home Automation**

Home automation (also called domotics) designates an emerging practice of increased automation of household appliances and features in residential dwellings, particularly through electronic means that allow for things impracticable, overly expensive or simply not possible in recent past decades.

Chapter- 2

Artificial Neural Network

An **artificial neural network (ANN)**, usually called **neural network (NN)**, is a mathematical model or computational model that is inspired by the structure and/or functional aspects of biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data.



An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in the human brain.

Background

The original inspiration for the term *Artificial Neural Network* came from examination of central nervous systems and their neurons, axons, dendrites and synapses which constitute the processing elements of biological neural networks investigated by neuroscience. In an artificial neural network, simple artificial nodes, variously called "neurons", "neurodes", "processing elements" (PEs) or "units", are connected together to form a network of nodes mimicking the biological neural networks — hence the term "artificial neural network".

Because neuroscience is still full of unanswered questions and since there are many levels of abstraction and therefore, many ways to take inspiration from the brain, there is no single formal definition of what an artificial neural network is. Most would agree that it involves a network of simple processing elements which can exhibit complex global behavior determined by the connections between the processing elements and element

parameters. While an artificial neural network does not have to be adaptive per se, its practical use comes with algorithms designed to alter the strength (weights) of the connections in the network to produce a desired signal flow.

These networks are also similar to the biological neural networks in the sense that functions are performed collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which various units are assigned. Currently, the term Artificial Neural Network (ANN) tends to refer mostly to neural network models employed in statistics, cognitive psychology and artificial intelligence. Neural network models designed with emulation of the central nervous system (CNS) in mind are a subject of theoretical neuroscience and computational neuroscience.

In modern software implementations of artificial neural networks, the approach inspired by biology has been largely abandoned for a more practical approach based on statistics and signal processing. In some of these systems, neural networks or parts of neural networks (such as artificial neurons) are used as components in larger systems that combine both adaptive and non-adaptive elements. While the more general approach of such adaptive systems is more suitable for real-world problem solving, it has far less to do with the traditional artificial intelligence connectionist models. What they do have in common, however, is the principle of non-linear, distributed, parallel and local processing and adaptation.

Models

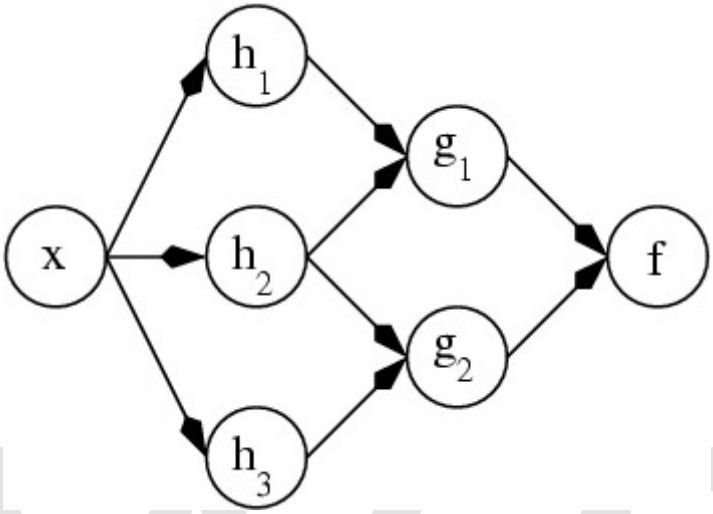
Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially simple mathematical models defining a function $f: X \rightarrow Y$ or a distribution over X or both X and Y, but sometimes models are also intimately associated with a particular learning algorithm or learning rule. A common use of the phrase ANN model really means the definition of a *class* of such functions (where members of the class are obtained by varying parameters, connection weights, or specifics of the architecture such as the number of neurons or their connectivity).

Network function

The word *network* in the term 'artificial neural network' refers to the inter–connections between the neurons in the different layers of each system. The most basic system has three layers. The first layer has input neurons which send data via synapses to the second layer of neurons and then via more synapses to the third layer of output neurons. More complex systems will have more layers of neurons with some having increased layers of input neurons and output neurons. The synapses store parameters called "weights" which are used to manipulate the data in the calculations.

The layers network through the mathematics of the system algorithms. The network function $f(x)$ is defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely

used type of composition is the *nonlinear weighted sum*, where $f(x) = K\left(\sum_i w_i g_i(x)\right)$, where K (commonly referred to as the activation function) is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$.



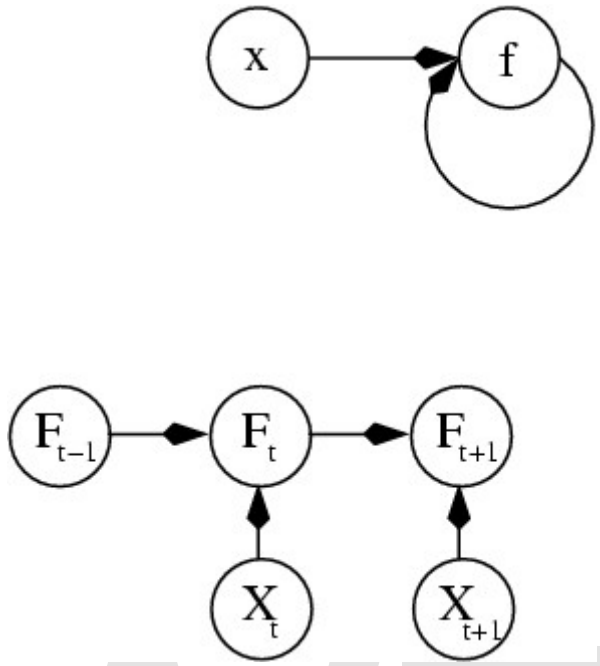
ANN dependency graph

This figure depicts such a decomposition of f, with dependencies between variables indicated by arrows. These can be interpreted in two ways.

The first view is the functional view: the input x is transformed into a 3-dimensional vector h, which is then transformed into a 2-dimensional vector g, which is finally transformed into f. This view is most commonly encountered in the context of optimization.

The second view is the probabilistic view: the random variable $F = f(G)$ depends upon the random variable $G = g(H)$, which depends upon $H = h(X)$, which depends upon the random variable X. This view is most commonly encountered in the context of graphical models.

The two views are largely equivalent. In either case, for this particular network architecture, the components of individual layers are independent of each other (e.g., the components of g are independent of each other given their input h). This naturally enables a degree of parallelism in the implementation.



Recurrent ANN dependency graph

Networks such as the previous one are commonly called feedforward, because their graph is a directed acyclic graph. Networks with cycles are commonly called recurrent. Such networks are commonly depicted in the manner shown at the top of the figure, where f is shown as being dependent upon itself. However, there is an implied temporal dependence which is not shown.

Learning

What has attracted the most interest in neural networks is the possibility of *learning*. Given a specific *task* to solve, and a *class* of functions F, learning means using a set of *observations* to find $f^* \in F$ which solves the task in some *optimal* sense.

This entails defining a cost function $C : F \to \mathbb{R}$ such that, for the optimal solution f^*, $C(f^*) \leq C(f) \forall f \in F$ (i.e., no solution has a cost less than the cost of the optimal solution).

The cost function C is an important concept in learning, as it is a measure of how far away a particular solution is from an optimal solution to the problem to be solved. Learning algorithms search through the solution space to find a function that has the smallest possible cost.

For applications where the solution is dependent on some data, the cost must necessarily be a *function of the observations*, otherwise we would not be modelling anything related to the data. It is frequently defined as a statistic to which only approximations can be made. As a simple example, consider the problem of finding the model f which minimizes $C = E\left[(f(x) - y)^2\right]$, for data pairs (x, y) drawn from some distribution $\mathcal{D}$. In

practical situations we would only have N samples from $\mathcal{D}$ and thus, for the above example, we would only minimize $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$. Thus, the cost is minimized over a sample of the data rather than the entire data set.

When $N \to \infty$ some form of online machine learning must be used, where the cost is partially minimized as each new example is seen. While online machine learning is often used when $\mathcal{D}$ is fixed, it is most useful in the case where the distribution changes slowly over time. In neural network methods, some form of online machine learning is frequently used for finite datasets.

Choosing a cost function

While it is possible to define some arbitrary, ad hoc cost function, frequently a particular cost will be used, either because it has desirable properties (such as convexity) or because it arises naturally from a particular formulation of the problem (e.g., in a probabilistic formulation the posterior probability of the model can be used as an inverse cost). Ultimately, the cost function will depend on the desired task. An overview of the three main categories of learning tasks is provided below.

Learning paradigms

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning.

Supervised learning

Supervised learning is the machine learning task of inferring a function from *supervised* training data. The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a *classifier* or a *regression function*. The inferred function should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. (Compare with unsupervised learning.) The parallel task in human and animal psychology is often referred to as concept learning.

Overview

In order to solve a given problem of supervised learning, one has to perform the following steps:

1. Determine the type of training examples. Before doing anything else, the engineer should decide what kind of data is to be used as an example. For instance, this

might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.
2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.
5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a *validation* set) of the training set, or via cross-validation.
6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

A wide range of supervised learning algorithms is available, each with its strengths and weaknesses. There is no single learning algorithm that works best on all supervised learning problems.

There are four major issues to consider in supervised learning:

Bias-variance tradeoff

A first issue is the tradeoff between *bias* and *variance* . Imagine that we have available several different, but equally good, training data sets. A learning algorithm is biased for a particular input x if, when trained on each of these data sets, it is systematically incorrect when predicting the correct output for x. A learning algorithm has high variance for a particular input x if it predicts different output values when trained on different training sets. The prediction error of a learned classifier is related to the sum of the bias and the variance of the learning algorithm . Generally, there is a tradeoff between bias and variance. A learning algorithm with low bias must be "flexible" so that it can fit the data well. But if the learning algorithm is too flexible, it will fit each training data set differently, and hence have high variance. A key aspect of many supervised learning methods is that they are able to adjust this tradeoff between bias and variance (either automatically or by providing a bias/variance parameter that the user can adjust).

Function complexity and amount of training data

The second issue is the amount of training data available relative to the complexity of the "true" function (classifier or regression function). If the true function is simple, then an "inflexible" learning algorithm with high bias and low variance will be able to learn it from a small amount of data. But if the true function is highly complex (e.g., because it involves complex interactions among many different input features and behaves differently in different parts of the input space), then the function will only be learnable from a very large amount of training data and using a "flexible" learning algorithm with low bias and high variance. Good learning algorithms therefore automatically adjust the bias/variance tradeoff based on the amount of data available and the apparent complexity of the function to be learned.

Dimensionality of the input space

A third issue is the dimensionality of the input space. If the input feature vectors have very high dimension, the learning problem can be difficult even if the true function only depends on a small number of those features. This is because the many "extra" dimensions can confuse the learning algorithm and cause it to have high variance. Hence, high input dimensionality typically requires tuning the classifier to have low variance and high bias. In practice, if the engineer can manually remove irrelevant features from the input data, this is likely to improve the accuracy of the learned function. In addition, there are many algorithms for feature selection that seek to identify the relevant features and discard the irrelevant ones. This is an instance of the more general strategy of dimensionality reduction, which seeks to map the input data into a lower dimensional space prior to running the supervised learning algorithm.

Noise in the output values

A fourth issue is the degree of noise in the desired output values (the supervisory targets). If the desired output values are often incorrect (because of human error or sensor errors), then the learning algorithm should not attempt to find a function that exactly matches the training examples. This is another case where it is usually best to employ a high bias, low variance classifier.

Other factors to consider

Other factors to consider when choosing and applying a learning algorithm include the following:

1. Heterogeneity of the data. If the feature vectors include features of many different kinds (discrete, discrete ordered, counts, continuous values), some algorithms are easier to apply than others. Many algorithms, including Support Vector Machines, linear regression, logistic regression, neural networks, and nearest neighbor methods, require that the input features be numerical and scaled to similar ranges (e.g., to the [-1,1] interval). Methods that employ a distance function, such as nearest neighbor methods and support vector machines with Gaussian kernels, are

particularly sensitive to this. An advantage of decision trees is that they easily handle heterogeneous data.

2. Redundancy in the data. If the input features contain redundant information (e.g., highly correlated features), some learning algorithms (e.g., linear regression, logistic regression, and distance based methods) will perform poorly because of numerical instabilities. These problems can often by solved by imposing some form of regularization.
3. Presence of interactions and non-linearities. If each of the features makes an independent contribution to the output, then algorithms based on linear functions (e.g., linear regression, logistic regression, Support Vector Machines, naive Bayes) and distance functions (e.g., nearest neighbor methods, support vector machines with Gaussian kernels) generally perform well. However, if there are complex interactions among features, then algorithms such as decision trees and neural networks work better, because they are specifically designed to discover these interactions. Linear methods can also be applied, but the engineer must manually specify the interactions when using them.

When considering a new application, the engineer can compare multiple learning algorithms and experimentally determine which one works best on the problem at hand (Tuning the performance of a learning algorithm can be very time-consuming. Given fixed resources, it is often better to spend more time collecting additional training data and more informative features than it is to spend extra time tuning the learning algorithms.

The most widely used learning algorithms are Support Vector Machines, linear regression, logistic regression, naive Bayes, linear discriminant analysis, decision trees, k-nearest neighbor algorithm, and Neural Networks (Multilayer perceptron).

How supervised learning algorithms work

Given a set of training examples of the form $\{(x_1, y_1), ..., (x_N, y_N)\}$, a learning algorithm seeks a function $g : X \to Y$, where *X* is the input space and *Y* is the output space. The function *g* is an element of some space of possible functions *G*, usually called the *hypothesis space*. It is sometimes convenient to represent *g* using a scoring function $f : X \times Y \to \mathbb{R}$ such that *g* is defined as returning the *y* value that gives the highest score: $g(x) = \arg\max_y f(x, y)$. Let *F* denote the space of scoring functions.

Although *G* and *F* can be any space of functions, many learning algorithms are probabilistic models where *g* takes the form of a conditional probability model $g(x) = P(y \mid x)$, or *f* takes the form of a joint probability model $f(x,y) = P(x,y)$. For example, naive Bayes and linear discriminant analysis are joint probability models, whereas logistic regression is a conditional probability model.

There are two basic approaches to choosing *f* or *g*: empirical risk minimization and structural risk minimization . Empirical risk minimization seeks the function that best fits the training data. Structural risk minimize includes a *penalty function* that controls the bias/variance tradeoff.

In both cases, it is assumed that the training set consists of a sample of independent and identically distributed pairs, $(x_i,\ y_i)$. In order to measure how well a function fits the training data, a loss function $L : Y \times Y \to \mathbb{R}^{\geq 0}$ is defined. For training example $(x_i,\ y_i)$, the loss of predicting the value $\hat{y}$ is $L(y_i, \hat{y})$.

The *risk* $R(g)$ of function *g* is defined as the expected loss of *g*. This can be estimated from the training data as

$$R_{emp}(g) = \frac{1}{N} \sum_i L(y_i, g(x_i))$$

.

Empirical risk minimization

In empirical risk minimization, the supervised learning algorithm seeks the function *g* that minimizes $R(g)$. Hence, a supervised learning algorithm can be constructed by applying an optimization algorithm to find *g*.

When *g* is a conditional probability distribution $P(y \mid x)$ and the loss function is the negative log likelihood: $L(y, \hat{y}) = -\log P(y|x)$, then empirical risk minimization is equivalent to maximum likelihood estimation.

When *G* contains many candidate functions or the training set is not sufficiently large, empirical risk minimization leads to high variance and poor generalization. The learning algorithm is able to memorize the training examples without generalizing well. This is called overfitting.

Structural risk minimization

Structural risk minimization seeks to prevent overfitting by incorporating a regularization penalty into the optimization. The regularization penalty can be viewed as implementing a form of Occam's razor that prefers simpler functions over more complex ones.

A wide variety of penalties have been employed that correspond to different definitions of complexity. For example, consider the case where the function *g* is a linear function of the form

$$g(x) = \sum_{j=1}^{d} \beta_j x_j$$

.

$\sum_j \beta_j^2$

A popular regularization penalty is , which is the squared Euclidean norm of the weights, also known as the L_2 norm. Other norms include the L_1 norm,

$$\sum_j |\beta_j|$$

, and the L_0 norm, which is the number of non-zero β_js. The penalty will be denoted by $C(g)$.

The supervised learning optimization problem is to find the function g that minimizes

$$J(g) = R_{emp}(g) + \lambda C(g).$$

The parameter λ controls the bias-variance tradeoff. When $\lambda = 0$, this gives empirical risk minimization with low bias and high variance. When λ is large, the learning algorithm will have high bias and low variance. The value of λ can be chosen empirically via cross validation.

The complexity penalty has a Bayesian interpretation as the negative log prior probability of g, $-\log P(g)$, in which case $J(g)$ is the posterior probabability of g.

Generative training

The training methods described above are *discriminative training* methods, because they seek to find a function g that discriminates well between the different output values. For the special case where $f(x,y) = P(x,y)$ is a joint probability distribution and the loss function is the negative log likelihood

$$-\sum_i \log P(x_i,y_i),$$

a risk minimization algorithm is said to perform *generative training*, because f can be regarded as a generative model that explains how the data were generated. Generative training algorithms are often simpler and more computationally efficient than discriminative training algorithms. In some cases, the solution can be computed in closed form as in naive Bayes and linear discriminant analysis.

Generalizations of supervised learning

There are several ways in which the standard supervised learning problem can be generalized:

1. Semi-supervised learning: In this setting, the desired output values are provided only for a subset of the training data. The remaining data is unlabeled.
2. Active learning: Instead of assuming that all of the training examples are given at the start, active learning algorithms interactively collect new examples, typically by making queries to a human user. Often, the queries are based on unlabeled data, which is a scenario that combines semi-supervised learning with active learning.
3. Structured prediction: When the desired output value is a complex object, such as a parse tree or a labeled graph, then standard methods must be extended.
4. Learning to rank: When the input is a set of objects and the desired output is a ranking of those objects, then again the standard methods must be extended.

Unsupervised learning

In unsupervised learning, some data x is given and the cost function to be minimized, that can be any function of the data x and the network's output, f.

The cost function is dependent on the task (what we are trying to model) and our *a priori* assumptions (the implicit properties of our model, its parameters and the observed variables).

As a trivial example, consider the model $f(x)=a$, where a is a constant and the cost $C=E[(x-f(x))^2]$. Minimizing this cost will give us a value of a that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: for example, in compression it could be related to the mutual information between x and y, whereas in statistical modeling, it could be related to the posterior probability of the model given the data. (Note that in both of those examples those quantities would be maximized rather than minimized).

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

Reinforcement learning

Inspired by old behaviorist psychology, **reinforcement learning** is an area of machine learning in computer science, concerned with how an *agent* ought to take *actions* in an *environment* so as to maximize some notion of cumulative *reward*. The problem, due to its generality, is studied in many other disciplines, such as control theory, operations research, information theory, simulation-based optimization, statistics, and Genetic Algorithms. In the operations research and control literature the field where reinforcement learning methods are studied is called *approximate dynamic programming*. The problem has been studied in the theory of optimal control, though most studies there are concerned with existence of optimal solutions and their characterization, and not with the learning or approximation aspects. In economics and game theory, reinforcement learning may be used to explain how equilibrium may arise under bounded rationality.

In machine learning, the environment is typically formulated as a Markov decision process (MDP), and many reinforcement learning algorithms for this context are highly related to dynamic programming techniques. The main difference to these classical techniques is that reinforcement learning algorithms do not need the knowledge of the MDP and they target large MDPs where exact methods become infeasible.

Reinforcement learning differs from standard supervised learning in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected. Further, there is a focus on on-line performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). The exploration vs. exploitation trade-off in reinforcement learning has been most thoroughly studied through the multi-armed bandit problem and in finite MDPs.

The basic reinforcement learning model consists of:

1. a set of environment states S;
2. a set of actions A;
3. rules of transitioning between states;
4. rules that determine the *scalar immediate reward* of a transition; and
5. rules that describe what the agent observes.

The rules are often stochastic. The observation typically involves the scalar immediate reward associated to the last transition. In many works, the agent is also assumed to observe the current environmental state, in which case we talk about *full observability*, whereas in the opposing case we talk about *partial observability*. Sometimes the set of actions available to the agent is restricted (e.g., you cannot spend more money than what you possess).

A reinforcement learning agent interacts with its environment in discrete time steps. At each time t, the agent receives an observation o_t, which typically includes the reward r_t. It then chooses an action a_t from the set of actions available, which is subsequently sent to the environment. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the *transition* (s_t,a_t,s_{t+1}) is determined. The goal of a reinforcement learning agent is to collect as much reward as possible. The agent can choose any action as a function of the history and it can even randomize its action selection.

When the agent's performance is compared to that of an agent which acts optimally from the beginning, the difference in performance gives rise to the notion of *regret*. Note that in order to act near optimally, the agent must reason about the long term consequences of its actions: In order to maximize my future income I better go to school now, although the immediate monetary reward associated with this might be negative.

Thus, reinforcement learning is particularly well suited to problems which include a long-term versus short-term reward trade-off. It has been applied successfully to various problems, including robot control, elevator scheduling, telecommunications, backgammon and chess.

Two components make reinforcement learning powerful: The use of samples to optimize performance and the use of function approximation to deal with large environments. Thus, reinforcement learning is most successful when the environment is big or cannot be precisely described. However, reinforcement learning methods can also be applied when the environment is big, but can be reasonably simulated, a problem studied in simulation-based optimization.

Exploration

The reinforcement learning problem as described requires clever exploration mechanisms. Randomly selecting actions is known to give rise to very poor performance. The case of (small) finite MDPs is relatively well understood by now. However, due to the lack of algorithms that would provably scale well with the number of states (or scale to problems with infinite state spaces), in practice people resort to simple exploration methods. One such method is ε-greedy, when the agent chooses the action that it believes has the best long-term effect with probability $1 - \varepsilon$, and it chooses an action uniformly at random, otherwise. Here, $0 < \varepsilon < 1$ is a tuning parameter, which is sometimes changed, either according to a fixed schedule (making the agent explore less as time goes by), or adaptively based on some heuristics (Tokic, 2010).

Algorithms for control learning

Even if the issue of exploration is disregarded and even if the state was observable (which we assume from now on), the problem remains to find out which actions are good based on past experience.

Criterion of optimality

For simplicity, assume for a moment that the problem studied is *episodic*, an episode ending when some *terminal state* is reached. Assume further that no matter what course of actions the agent takes, termination is inevitable with probability one. Under some additional mild regularity conditions the expectation of the total reward is then well-defined, for *any* policy π and any initial distribution over the states. Given a fixed initial distribution μ, we can thus assign the expected return ρ^π to policy π:

$$\rho^\pi = E[R \mid \pi],$$

where the random variable R denotes the *return* and is defined by

$$R = \sum_{t=0}^{N-1} r_{t+1},$$

where r_{t+1} is the reward received after the *t*-th transition, the initial state is sampled at random from μ and actions are selected by policy π. Here, *N* denotes the (random) time when a terminal state is reached, i.e., the time when the episode terminates.

In the case of non-episodic problems the return is often *discounted*,

$$R = \sum_{t=0}^{\infty} \gamma^t r_{t+1},$$

giving rise to the total expected discounted reward criterion. Here $0 \leq \gamma \leq 1$is the so-called *discount-factor*. Since the undiscounted return is a special case of the discounted return, from now on we will assume discounting. Although this looks innocent enough, discounting is in fact problematic if one cares about online performance. This is because discounting makes the initial time steps more important. Since a learning agent is likely to make mistakes during the first few steps after its "life" starts, no uninformed learning algorithm can achieve near-optimal performance under discounting even if the class of environments is restricted to that of finite MDPs. (This does not mean though that, given enough time, a learning agent cannot figure how to act near-optimally, if time was restarted.)

The problem then is to specify an algorithm that can be used to find a policy with maximum expected return. From the theory of MDPs it is known that, without the loss of generality, the search can be restricted to the set of the so-called *stationary* policies. A policy is called stationary if the action-distribution returned by it depends only the last state visited (which is part of the observation history of the agent, by our simplifying assumption). In fact, the search can be further restricted to *deterministic* stationary policies. A deterministic stationary policy is one which deterministically selects actions based on the current state. Since any such policy can be identified with a mapping from the set of states to the set of action, these policies can be identified with such mappings with no loss of generality.

Brute force

The naive brute force approach entails the following two steps:

1. For each possible policy, sample returns while following it
2. Choose the policy with the largest expected return

One problem with this is that the number of policies can be extremely large, or even infinite. Another is that variance of the returns might be large, in which case a large number of samples will be required to accurately estimate the return of each policy.

These problems can be ameliorated if we assume some structure and perhaps allow samples generated from one policy to influence the estimates made for another. The two main approaches for achieving this are value function estimation and direct policy search.

Value function approaches

Value function approaches attempt to find a policy that maximizes the return by maintaining a set of estimates of expected returns for some policy (usually either the "current" or the optimal one).

These methods rely on the theory of MDPs, where optimality is defined in a sense which is stronger than the above one: A policy is called optimal if it achieves the best expected return from *any* initial state (i.e., initial distributions play no role in this definition). Again, one can always find an optimal policy amongst stationary policies.

To define optimality in a formal manner, define the value of a policy π by

$$V^{\pi}(s) = E[R \mid s,\pi],$$

where R stands for the random return associated with following π from the initial state s. Define $V^{*}(s)$ as the maximum possible value of $V^{\pi}(s)$, where π is allowed to change:

$$V^*(s) = \sup_{\pi} V^{\pi}(s).$$

A policy which achieves these *optimal values* in *each* state is called *optimal*. Clearly, a policy optimal in this strong sense is also optimal in the sense that it maximizes the expected return ρ^{π}, since $\rho^{\pi} = E[V^{\pi}(S)]$, where S is a state randomly sampled from the distribution μ.

Although state-values suffice to define optimality, it will prove to be useful to define action-values. Given a state s, an action a and a policy π, the action-value of the pair (s,a) under π is defined by

$$Q^{\pi}(s,a) = E[R|s,a,\pi],$$

where, now, R stands for the random return associated with first taking action a in state s and following π, thereafter.

It is well-known from the theory of MDPs that if someone gives us Q for an optimal policy, we can always choose optimal actions (and thus act optimally) by simply choosing the action with the highest value at each state. The *action-value function* of such an optimal policy is called the *optimal action-value function* and is denoted by Q^{*}. In summary, the knowledge of the optimal action-value function *alone* suffices to know how to act optimally.

Assuming full knowledge of the MDP, there are two basic approaches to compute the optimal action-value function, value iteration and policy iteration. Both algorithms compute a sequence of functions Q_k ($k = 0, 1, 2, \ldots$,) which converge to Q^{*}. Computing these functions involves computing expectations over the whole state-space,

which is impractical for all, but the smallest (finite) MDPs, never mind the case when the MDP is unknown. In reinforcement learning methods the expectations are approximated by averaging over samples and one uses function approximation techniques to cope with the need to represent value functions over large state-action spaces.

Monte Carlo methods

The simplest Monte Carlo methods can be used in an algorithm that mimics policy iteration. Policy iteration consists of two steps: *policy evaluation* and *policy improvement*. The Monte Carlo methods are used in the policy evaluation step. In this step, given a stationary, deterministic policy π, the goal is to compute the function values $Q^{\pi}(s,a)$ (or a good approximation to them) for all state-action pairs (s,a). Assume (for simplicity) that the MDP is finite and in fact a table representing the action-values fits into the memory. Further, assume that the problem is episodic and after each episode a new one starts from some random initial state. Then, the estimate of the value of a given state-action pair (s,a)can be computed by simply averaging the sampled returns which originated from (s,a) over time. Given enough time, this procedure can thus construct a precise estimate Q of the action-value function Q^{π}. This finishes the description of the policy evaluation step. In the policy improvement step, as it is done in the standard policy iteration algorithm, the next policy is obtained by computing a *greedy* policy with respect to Q: Given a state s, this new policy returns an action that maximizes $Q(s,\cdot)$. In practice one often avoids computing and storing the new policy, but uses lazy evaluation to defer the computation of the maximizing actions to when they are actually needed.

A few problems with this procedure are as follows:

- The procedure may waste too much time on evaluating a suboptimal policy;
- It uses samples inefficiently in that a long trajectory is used to improve the estimate only of the *single* state-action pair that started the trajectory;
- When the returns along the trajectories have *high variance*, convergence will be slow;
- It works in *episodic problems only*;
- It works in *small, finite MDPs only*.

Temporal difference methods

The first issue is easily corrected by allowing the procedure to change the policy (at all, or at some states) before the values settle. However good this sounds, this may be dangerous as this might prevent convergence. Still, most current algorithms implement this idea, giving rise to the class of *generalized policy iteration* algorithm. We note in passing that actor critic methods belong to this category.

The second issue can be corrected within the algorithm by allowing trajectories to contribute to any state-action pair in them. This may also help to some extent with the third problem, although a better solution when returns have high variance is to use Sutton's temporal difference (TD) methods which are based on the recursive Bellman

equation. Note that the computation in TD methods can be incremental (when after each transition the memory is changed and the transition is thrown away), or batch (when the transitions are collected and then the estimates are computed once based on a large number of transitions). Batch methods, a prime example of which is the least-squares temporal difference method due to Bradtke and Barto (1996), may use the information in the samples better, whereas incremental methods are the only choice when batch methods become infeasible due to their high computational or memory complexity. In addition, there exist methods that try to unify the advantages of the two approaches. Methods based on temporal differences also overcome the second but last issue.

In order to address the last issue mentioned in the previous section, *function approximation methods* are used. In *linear function approximation* one starts with a mapping φ that assigns a finite dimensional vector to each state-action pair. Then, the action values of a state-action pair (*s*,*a*) are obtained by linearly combining the components of φ(*s*,*a*) with some *weights* θ:

$$Q(s,a) = \sum_{i=1}^{d} \theta_i \phi_i(s,a)$$

.

The algorithms then adjust the weights, instead of adjusting the values associated with the individual state-action pairs. However, linear function approximation is not the only choice. More recently, methods based on ideas from nonparametric statistics (which can be seen to construct their own features) have been explored.

So far, the discussion was restricted to how policy iteration can be used as a basis of the designing reinforcement learning algorithms. Equally importantly, value iteration can also be used as a starting point, giving rise to the Q-Learning algorithm (Watkins 1989) and its many variants.

The problem with methods that use action-values is that they may need highly precise estimates of the competing action values, which can be hard to obtain when the returns are noisy. Though this problem is mitigated to some extent by temporal difference methods and if one uses the so-called compatible function approximation method, more work remains to be done to increase generality and efficiency. Another problem specific to temporal difference methods comes from their reliance on the recursive Bellman equation. Most temporal difference methods have a so-called λ parameter $(0 \leq \lambda \leq 1)$ that allows one to continuously interpolate between Monte-Carlo methods (which do not rely on the Bellman equations) and the basic temporal difference methods (which rely entirely on the Bellman equations), which can thus be effective in palliating this issue.

Direct policy search

An alternative method to find a good policy is to search directly in (some subset) of the policy space, in which case the problem becomes an instance of stochastic optimization. The two approaches available are gradient-based and gradient-free methods.

Gradient-based methods (giving rise to the so-called *policy gradient methods*) start with a mapping from a finite dimensional (parameter) space to the space of policies: given the parameter vector θ, let π_θ denote the policy associated to θ. Define the performance function by

$$\rho(\theta) = \rho^{\pi_\theta}.$$

Under mild conditions this function will be differentiable as a function of the parameter vector θ. If the gradient of ρ was known, one could use gradient ascent. Since an analytic expression for the gradient is not available, one must rely on a noisy estimate. Such an estimate can be constructed in many ways, giving rise to algorithms like Williams' REINFORCE method (which is also known as the likelihood ratio method in the simulation-based optimization literature). Policy gradient methods have received a lot of attention in the last couple of years (e.g., Peters et al. (2003)), but they remain an active field. The issue with many of these methods is that they may get stuck in local optima (as they are based on local search).

A large class of methods avoids relying on gradient information. These include simulated annealing, cross-entropy search or methods of evolutionary computation. Many gradient-free methods can achieve (in theory and in the limit) a global optimum. In a number of cases they have indeed demonstrated remarkable performance.

The issue with policy search methods is that they may converge slowly if the information based on which they act is noisy. For example, this happens when in episodic problems the trajectories are long and the variance of the returns is large. As argued beforehand, value-function based methods that rely on temporal differences might help in this case. In recent years, several actor-critic algorithms have been proposed following this idea and were demonstrated to perform well on various benchmarks.

Theory

The theory for small, finite MDPs is quite mature. Both the asymptotic and finite-sample behavior of most algorithms is well-understood. As mentioned beforehand, algorithms with provably good online performance (addressing the exploration issue) are known. The theory of large MDPs needs more work. Efficient exploration is largely untouched (except for the case of bandit problems). Although finite-time performance bounds appeared for many algorithms in the recent years, these bounds are expected to be rather loose and thus more work is needed to better understand the relative advantages, as well as the limitations of these algorithms. For incremental algorithm asymptotic convergence issues have been settled. Recently, new incremental, temporal-difference-based algorithms have appeared which converge under a much wider set of conditions than was

previously possible (for example, when used with arbitrary, smooth function approximation).

Current research

Current research topics include: adaptive methods which work with fewer (or no) parameters under a large number of conditions, addressing the exploration problem in large MDPs, large scale empirical evaluations, learning and acting under partial information (e.g., using Predictive State Representation), modular and hierarchical reinforcement learning, improving existing value-function and policy search methods, algorithms that work well with large (or continuous) action spaces, transfer learning, lifelong learning, efficient sample-based planning (e.g., based on Monte-Carlo tree search). Multiagent or Distributed Reinforcement Learning is also a topic of interest in current research. There is also a growing interest in real life applications of reinforcement learning. Successes of reinforcement learning are collected on here and here.

Reinforcement learning algorithms such as TD learning are also being investigated as a model for Dopamine-based learning in the brain. In this model, the dopaminergic projections from the substantia nigra to the basal ganglia function as the prediction error. Reinforcement learning has also been used as a part of the model for human skill learning, especially in relation to the interaction between implicit and explicit learning in skill acquisition (the first publication on this application was in 1995-1996, and there have been many follow-up studies).

Learning algorithms

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and statistical estimation. Recent developments in this field use particle swarm optimization and other swarm intelligence techniques.

Most of the algorithms used in training artificial neural networks employ some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction.

Evolutionary methods, simulated annealing, expectation-maximization and non-parametric methods are some commonly used methods for training neural networks.

Temporal perceptual learning relies on finding temporal relationships in sensory signal streams. In an environment, statistically salient temporal correlations can be found by monitoring the arrival times of sensory signals. This is done by the perceptual network.

Employing artificial neural networks

Perhaps the greatest advantage of ANNs is their ability to be used as an arbitrary function approximation mechanism which 'learns' from observed data. However, using them is not so straightforward and a relatively good understanding of the underlying theory is essential.

- Choice of model: This will depend on the data representation and the application. Overly complex models tend to lead to problems with learning.
- Learning algorithm: There are numerous trade-offs between learning algorithms. Almost any algorithm will work well with the *correct hyperparameters* for training on a particular fixed data set. However selecting and tuning an algorithm for training on unseen data requires a significant amount of experimentation.
- Robustness: If the model, cost function and learning algorithm are selected appropriately the resulting ANN can be extremely robust.

With the correct implementation, ANNs can be used naturally in online learning and large data set applications. Their simple implementation and the existence of mostly local dependencies exhibited in the structure allows for fast, parallel implementations in hardware.

Applications

The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical.

Real-life applications

The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.
- Robotics, including directing manipulators, Computer numerical control.

Application areas include system identification and control (vehicle control, process control), quantum chemistry, game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical

diagnosis, financial applications (automated trading systems), data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

Neural networks and neuroscience

Theoretical and computational neuroscience is the field concerned with the theoretical analysis and computational modeling of biological neural systems. Since neural systems are intimately related to cognitive processes and behavior, the field is closely related to cognitive and behavioral modeling.

The aim of the field is to create models of biological neural systems in order to understand how biological systems work. To gain this understanding, neuroscientists strive to make a link between observed biological processes (data), biologically plausible mechanisms for neural processing and learning (biological neural network models) and theory (statistical learning theory and information theory).

Types of models

Many models are used in the field defined at different levels of abstraction and modeling different aspects of neural systems. They range from models of the short-term behavior of individual neurons, models of how the dynamics of neural circuitry arise from interactions between individual neurons and finally to models of how behavior can arise from abstract neural modules that represent complete subsystems. These include models of the long-term, and short-term plasticity, of neural systems and their relations to learning and memory from the individual neuron to the system level.

Current research

While initial research had been concerned mostly with the electrical characteristics of neurons, a particularly important part of the investigation in recent years has been the exploration of the role of neuromodulators such as dopamine, acetylcholine, and serotonin on behavior and learning.

Biophysical models, such as BCM theory, have been important in understanding mechanisms for synaptic plasticity, and have had applications in both computer science and neuroscience. Research is ongoing in understanding the computational algorithms used in the brain, with some recent biological evidence for radial basis networks and neural backpropagation as mechanisms for processing data.

Computational devices have been created in CMOS for both biophysical simulation and neuromorphic computing. More recent efforts show promise for creating nanodevices for very large scale principal components analyses and convolution. If successful, these effort could usher in a new era of neural computing that is a step beyond digital computing, because it depends on learning rather than programming and because it is fundamentally analog rather than digital even though the first instantiations may in fact be with CMOS digital devices.

Neural network software

Neural network software is used to simulate, research, develop and apply artificial neural networks, biological neural networks and in some cases a wider array of adaptive systems.

Types of artificial neural networks

Artificial neural network types vary from those which have only one or two layers of single direction logic to complicated multi–input many directional feedback loop and layers. On the whole, these systems use algorithms in their programming to determine control and organization of their functions. Some may be as simple, one neuron layer with an input and an output, and others can mimic complex systems such as dANN which can mimic chromosomal DNA through sizes at cellular level, into artificial organisms and simulate reproduction, mutation and population sizes.

Most systems use "weights" to change the parameters of the throughput and the varying connections to the neurons. Artificial neural networks can be autonomous and learn by input from outside "teachers" or even self-teaching from written in rules.

Theoretical properties

Computational power

The multi-layer perceptron (MLP) is a universal function approximator, as proven by the Cybenko theorem. However, the proof is not constructive regarding the number of neurons required or the settings of the weights.

Work by Hava Siegelmann and Eduardo D. Sontag has provided a proof that a specific recurrent architecture with rational valued weights (as opposed to full precision real number-valued weights) has the full power of a Universal Turing Machine using a finite number of neurons and standard linear connections. They have further shown that the use of irrational values for weights results in a machine with super-Turing power.

Capacity

Artificial neural network models have a property called 'capacity', which roughly corresponds to their ability to model any given function. It is related to the amount of information that can be stored in the network and to the notion of complexity.

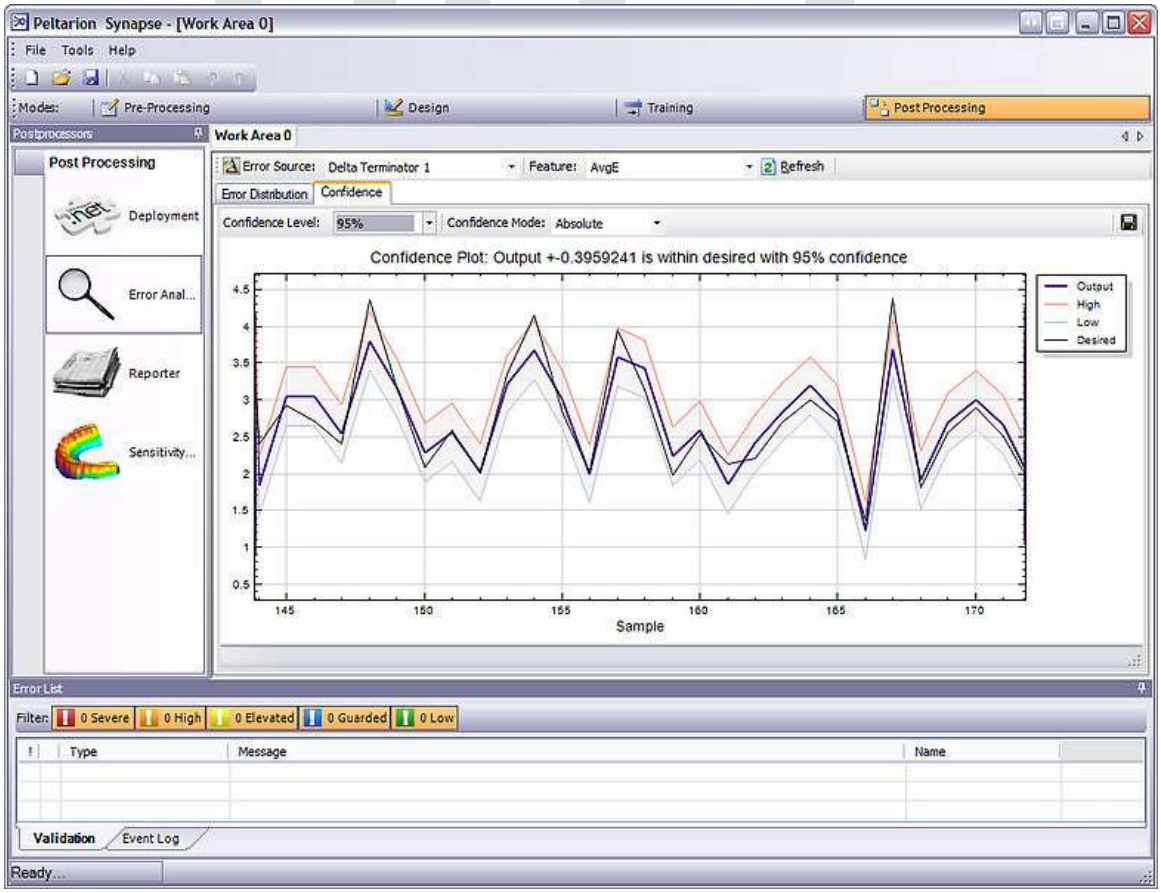
Convergence

Nothing can be said in general about convergence since it depends on a number of factors. Firstly, there may exist many local minima. This depends on the cost function

and the model. Secondly, the optimization method used might not be guaranteed to converge when far away from a local minimum. Thirdly, for a very large amount of data or parameters, some methods become impractical. In general, it has been found that theoretical guarantees regarding convergence are an unreliable guide to practical application.

Generalization and statistics

In applications where the goal is to create a system that generalizes well in unseen examples, the problem of over-training has emerged. This arises in convoluted or over-specified systems when the capacity of the network significantly exceeds the needed free parameters. There are two schools of thought for avoiding this problem: The first is to use cross-validation and similar techniques to check for the presence of overtraining and optimally select hyperparameters such as to minimize the generalization error. The second is to use some form of *regularization*. This is a concept that emerges naturally in a probabilistic (Bayesian) framework, where the regularization can be performed by selecting a larger prior probability over simpler models; but also in statistical learning theory, where the goal is to minimize over two quantities: the 'empirical risk' and the 'structural risk', which roughly corresponds to the error over the training set and the predicted error in unseen data due to overfitting.



Confidence analysis of a neural network

Supervised neural networks that use an MSE cost function can use formal statistical methods to determine the confidence of the trained model. The MSE on a validation set can be used as an estimate for variance. This value can then be used to calculate the confidence interval of the output of the network, assuming a normal distribution. A confidence analysis made this way is statistically valid as long as the output probability distribution stays the same and the network is not modified.

By assigning a softmax activation function on the output layer of the neural network (or a softmax component in a component-based neural network) for categorical target variables, the outputs can be interpreted as posterior probabilities. This is very useful in classification as it gives a certainty measure on classifications.

The softmax activation function is:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{c} e^{x_j}}$$

Dynamic properties

Various techniques originally developed for studying disordered magnetic systems (i.e., the spin glass) have been successfully applied to simple neural network architectures, such as the Hopfield network. Influential work by E. Gardner and B. Derrida has revealed many interesting properties about perceptrons with real-valued synaptic weights, while later work by W. Krauth and M. Mezard has extended these principles to binary-valued synapses.

Criticism

A common criticism of artificial neural networks, particularly in robotics, is that they require a large diversity of training for real-world operation. Dean Pomerleau, in his research presented in the paper "Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving," uses a neural network to train a robotic vehicle to drive on multiple types of roads (single lane, multi-lane, dirt, etc.). A large amount of his research is devoted to (1) extrapolating multiple training scenarios from a single training experience, and (2) preserving past training diversity so that the system does not become overtrained (if, for example, it is presented with a series of right turns – it should not learn to always turn right). These issues are common in neural networks that must decide from amongst a wide variety of responses.

A. K. Dewdney, a former *Scientific American* columnist, wrote in 1997, "Although neural nets do solve a few toy problems, their powers of computation are so limited that I am surprised anyone takes them seriously as a general problem-solving tool." (Dewdney, p. 82)

Arguments for Dewdney's position are that to implement large and effective software neural networks, much processing and storage resources need to be committed. While the brain has hardware tailored to the task of processing signals through a graph of neurons, simulating even a most simplified form on Von Neumann technology may compel a NN designer to fill many millions of database rows for its connections - which can lead to abusive RAM and HD necessities. Furthermore, the designer of NN systems will often need to simulate the transmission of signals through many of these connections and their associated neurons - which must often be matched with incredible amounts of CPU processing power and time. While neural networks often yield *effective* programs, they too often do so at the cost of time and money *efficiency*.

Arguments against Dewdney's position are that neural nets have been successfully used to solve many complex and diverse tasks, ranging from autonomously flying aircraft to detecting credit card fraud. Technology writer Roger Bridgman commented on Dewdney's statements about neural nets:

Neural networks, for instance, are in the dock not only because they have been hyped to high heaven, (what hasn't?) but also because you could create a successful net without understanding how it worked: the bunch of numbers that captures its behaviour would in all probability be "an opaque, unreadable table...valueless as a scientific resource". In spite of his emphatic declaration that science is not technology, Dewdney seems here to pillory neural nets as bad science when most of those devising them are just trying to be good engineers. An unreadable table that a useful machine could read would still be well worth having.

Some other criticisms came from believers of hybrid models (combining neural networks and symbolic approaches). They advocate the intermix of these two approaches and believe that hybrid models can better capture the mechanisms of the human mind (Sun and Bookman 1994).

Chapter- 3

Distributed Control System

A **distributed control system** (DCS) refers to a control system usually of a manufacturing system, process or any kind of dynamic system, in which the controller elements are not central in location (like the brain) but are distributed throughout the system with each component sub-system controlled by one or more controllers. The entire system of controllers is connected by networks for communication and monitoring.

DCS is a very broad term used in a variety of industries, to monitor and control distributed equipment.

- Electrical power grids and electrical generation plants
- Environmental control systems
- Traffic signals
- radio signals
- Water management systems
- Oil refining plants
- Chemical plants
- Pharmaceutical manufacturing
- Sensor networks
- Dry cargo and bulk oil carrier ships

Elements

A DCS typically uses custom designed processors as controllers and uses both proprietary interconnections and communications protocol for communication. Input and output modules form component parts of the DCS. The processor receives information from input modules and sends information to output modules. The input modules receive information from input instruments in the process (a.k.a. field) and transmit instructions to the output instruments in the field. Computer buses or electrical buses connect the processor and modules through multiplexer or demultiplexers. Buses also connect the

distributed controllers with the central controller and finally to the Human-Machine Interface (HMI) or control consoles.

Elements of a distributed control system may directly connect to physical equipment such as switches, pumps and valves or may work through an intermediate system such as a SCADA system.

Applications

Distributed Control Systems (DCSs) are dedicated systems used to control manufacturing processes that are continuous or batch-oriented, such as oil refining, petrochemicals, central station power generation, pharmaceuticals, food & beverage manufacturing, cement production, steelmaking, and papermaking. DCSs are connected to sensors and actuators and use setpoint control to control the flow of material through the plant. The most common example is a setpoint control loop consisting of a pressure sensor, controller, and control valve. Pressure or flow measurements are transmitted to the controller, usually through the aid of a signal conditioning Input/Output (I/O) device. When the measured variable reaches a certain point, the controller instructs a valve or actuation device to open or close until the fluidic flow process reaches the desired setpoint. Large oil refineries have many thousands of I/O points and employ very large DCSs. Processes are not limited to fluidic flow through pipes, however, and can also include things like paper machines and their associated quality controls.

A typical DCS consists of functionally and/or geographically distributed digital controllers capable of executing from 1 to 256 or more regulatory control loops in one control box. The input/output devices (I/O) can be integral with the controller or located remotely via a field network. Today's controllers have extensive computational capabilities and, in addition to proportional, integral, and derivative (PID) control, can generally perform logic and sequential control.

DCSs may employ one or several workstations and can be configured at the workstation or by an off-line personal computer. Local communication is handled by a control network with transmission over twisted pair, coaxial, or fiber optic cable. A server and/or applications processor may be included in the system for extra computational, data collection, and reporting capability.

History

Early minicomputers were used in the control of industrial processes since the beginning of the 1960s. The IBM 1800, for example, was an early computer that had input/output hardware to gather process signals in a plant for conversion from field contact levels (for digital points) and analog signals to the digital domain.

The first industrial control computer system was built 1959 at the Texaco Port Arthur, Texas, refinery with an RW-300 of the Ramo-Wooldridge Company .

The DCS was introduced in 1975. Both Honeywell and Japanese electrical engineering firm Yokogawa introduced their own independently produced DCSs at roughly the same time, with the TDC 2000 and CENTUM systems, respectively. US-based Bristol also introduced their UCS 3000 universal controller in 1975. In 1980, Bailey (now part of ABB) introduced the NETWORK 90 system. Also in 1980, Fischer & Porter Company (now also part of ABB) introduced DCI-4000 (DCI stands for Distributed Control Instrumentation).

The DCS largely came about due to the increased availability of microcomputers and the proliferation of microprocessors in the world of process control. Computers had already been applied to process automation for some time in the form of both Direct Digital Control (DDC) and Set Point Control. In the early 1970s Taylor Instrument Company, (now part of ABB) developed the 1010 system, Foxboro the FOX1 system and Bailey Controls the 1055 systems. All of these were DDC applications implemented within minicomputers (DEC PDP-11, Varian Data Machines, MODCOMP etc.) and connected to proprietary Input/Output hardware. Sophisticated (for the time) continuous as well as batch control was implemented in this way. A more conservative approach was Set Point Control, where process computers supervised clusters of analog process controllers. A CRT-based workstation provided visibility into the process using text and crude character graphics. Availability of a fully functional graphical user interface was a way away.

Central to the DCS model was the inclusion of control function blocks. Function blocks evolved from early, more primitive DDC concepts of "Table Driven" software. One of the first embodiments of object-oriented software, function blocks were self contained "blocks" of code that emulated analog hardware control components and performed tasks that were essential to process control, such as execution of PID algorithms. Function blocks continue to endure as the predominant method of control for DCS suppliers, and are supported by key technologies such as **Foundation Fieldbus** today.

Midac Systems of Sydney Australia developed an objected-oriented distributed direct digital control system in 1982. The central system ran 11 microprocessors sharing tasks and common memory and connected to a serial communication network of distributed controllers each running two Z80s. The system was installed at the University of Melbourne.

Digital communication between distributed controllers, workstations and other computing elements (peer to peer access) was one of the primary advantages of the DCS. Attention was duly focused on the networks, which provided the all-important lines of communication that, for process applications, had to incorporate specific functions such as determinism and redundancy. As a result, many suppliers embraced the IEEE 802.4 networking standard. This decision set the stage for the wave of migrations necessary when information technology moved into process automation and IEEE 802.3 rather than IEEE 802.4 prevailed as the control LAN.

The Network Centric Era of the 1980s

The DCS brought distributed intelligence to the plant and established the presence of computers and microprocessors in process control, but it still did not provide the reach and openness necessary to unify plant resource requirements. In many cases, the DCS was merely a digital replacement of the same functionality provided by analog controllers and a panelboard display. This was embodied in The Purdue Reference Model (PRM) that was developed to define Manufacturing Operations Management relationships. PRM later formed the basis for ISA95 standards activities today.

In the 1980s, users began to look at DCSs as more than just basic process control. A very early example of a Direct Digital Control DCS was completed by the Australian business Midac in 1981-1982 using R-Tec Australian designed hardware. The system installed at the University of Melbourne used a serial communications network, connecting campus buildings back to a control room "front end". Each remote unit ran 2 Z80 microprocessors whilst the front end ran 11 in a Parallel Processing configuration with paged common memory to share tasks and could run up to 20,000 concurrent controls objects.

It was believed that if openness could be achieved and greater amounts of data could be shared throughout the enterprise that even greater things could be achieved. The first attempts to increase the openness of DCSs resulted in the adoption of the predominant operating system of the day: *UNIX*. UNIX and its companion networking technology TCP-IP were developed by the Department of Defense for openness, which was precisely the issue the process industries were looking to resolve.

As a result suppliers also began to adopt Ethernet-based networks with their own proprietary protocol layers. The full TCP/IP standard was not implemented, but the use of Ethernet made it possible to implement the first instances of object management and global data access technology. The 1980s also witnessed the first PLCs integrated into the DCS infrastructure. Plant-wide historians also emerged to capitalize on the extended reach of automation systems. The first DCS supplier to adopt UNIX and Ethernet networking technologies was Foxboro, who introduced the I/A Series system in 1987.

The Application Centric Era of the 1990s

The drive toward openness in the 1980s gained momentum through the 1990s with the increased adoption of Commercial off-the-shelf (COTS) components and IT standards. Probably the biggest transition undertaken during this time was the move from the UNIX operating system to the Windows environment. While the realm of the real time operating system (RTOS) for control applications remains dominated by real time commercial variants of UNIX or proprietary operating systems, everything above real-time control has made the transition to Windows.

The introduction of Microsoft at the desktop and server layers resulted in the development of technologies such as OLE for Process Control (OPC), which is now a de

facto industry connectivity standard. Internet technology also began to make its mark in automation and the DCS world, with most DCS HMI supporting Internet connectivity. The '90s were also known for the "Fieldbus Wars", where rival organizations competed to define what would become the IEC fieldbus standard for digital communication with field instrumentation instead of 4-20 milliamp analog communications. The first fieldbus installations occurred in the 1990s. Towards the end of the decade, the technology began to develop significant momentum, with the market consolidated around Foundation Fieldbus and Profibus PA for process automation applications. Some suppliers built new systems from the ground up to maximize functionality with fieldbus, such as Honeywell with Experion & Plantscape SCADA systems, ABB with System 800xA , Emerson Process Management with the DeltaV control system, Siemens with the Simatic PCS7 and **azbil** from Yamatake with the Harmonas-DEO system.

The impact of COTS, however, was most pronounced at the hardware layer. For years, the primary business of DCS suppliers had been the supply of large amounts of hardware, particularly I/O and controllers. The initial proliferation of DCSs required the installation of prodigious amounts of this hardware, most of it manufactured from the bottom up by DCS suppliers. Standard computer components from manufacturers such as Intel and Motorola, however, made it cost prohibitive for DCS suppliers to continue making their own components, workstations, and networking hardware.

As the suppliers made the transition to COTS components, they also discovered that the hardware market was shrinking fast. COTS not only resulted in lower manufacturing costs for the supplier, but also steadily decreasing prices for the end users, who were also becoming increasingly vocal over what they perceived to be unduly high hardware costs. Some suppliers that were previously stronger in the PLC business, such as Rockwell Automation and Siemens, were able to leverage their expertise in manufacturing control hardware to enter the DCS marketplace with cost effective offerings, while the stability/scalability/reliability and functionality of these emerging systems are still improving. The traditional DCS suppliers introduced new generation DCS System based on the latest Communication and IEC Standards, which resulting in a trend of combining the traditional concepts/functionalities for PLC and DCS into a one for all solution—named "Process Automation System". The gaps among the various systems remain at the areas such as: the database integrity, pre-engineering functionality, system maturity, communication transparency and reliability. While it is expected the cost ratio is relatively the same (the more powerful the systems are, the more expensive they will be), the reality of the automation business is often operating strategically case by case. The current next evolution step is called Collaborative Process Automation Systems.

To compound the issue, suppliers were also realizing that the hardware market was becoming saturated. The lifecycle of hardware components such as I/O and wiring is also typically in the range of 15 to over 20 years, making for a challenging replacement market. Many of the older systems that were installed in the 1970s and 1980s are still in use today, and there is a considerable installed base of systems in the market that are approaching the end of their useful life. Developed industrial economies in North America, Europe, and Japan already had many thousands of DCSs installed, and with few

if any new plants being built, the market for new hardware was shifting rapidly to smaller, albeit faster growing regions such as China, Latin America, and Eastern Europe.

Because of the shrinking hardware business, suppliers began to make the challenging transition from a hardware-based business model to one based on software and value-added services. It is a transition that is still being made today. The applications portfolio offered by suppliers expanded considerably in the '90s to include areas such as production management, model-based control, real-time optimization, Plant Asset Management (PAM), Real Time Performance Management (RPM) tools, alarm management, and many others. To obtain the true value from these applications, however, often requires a considerable service content, which the suppliers also provide.

Chapter- 4

User Interface

In the industrial design field of human-machine interaction, the user interface is (a place) where interaction between humans and machines occurs. The goal of interaction between a human and a machine at the user interface is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls. and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

A **user interface** is the system by which people (users) interact with a machine. The user interface includes hardware (physical) and software (logical) components. User interfaces exist for various systems, and provide a means of:

- Input, allowing the users to manipulate a system, and/or
- Output, allowing the system to indicate the effects of the users' manipulation.

Generally, the goal of human-machine interaction engineering is to produce a user interface which makes it easy, efficient, and enjoyable to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human.

Ever since the increased use of personal computers and the relative decline in societal awareness of heavy machinery, the term user interface has taken on overtones of the (graphical) user interface, while industrial control panel and machinery control design discussions more commonly refer to human-machine interfaces.

Other terms for **user interface** include ***human-computer interface*** (HCI) and ***man-machine interface*** (MMI).

Introduction

To work with a system, users have to be able to control and assess the state of the system. For example, when driving an automobile, the driver uses the steering wheel to control the direction of the vehicle, and the accelerator pedal, brake pedal and gearstick to control the speed of the vehicle. The driver perceives the position of the vehicle by looking through the windshield and exact speed of the vehicle by reading the speedometer. The *user interface of the automobile* is on the whole composed of the instruments the driver can use to accomplish the tasks of driving and maintaining the automobile.

Terminology

There is a distinct difference between **User Interface** versus **Operator Interface** or **Human Machine Interface (HMI)**.

- The term ***user interface*** is often used in the context of (personal) **computer systems** and electronic devices
  - Where a network of equipment or computers are interlinked through an MES (Manufacturing Execution System)-or Host.
  - An HMI is typically local to one machine or piece of equipment, and is the interface method between the human and the equipment/machine. An Operator interface is the interface method by which multiple equipment that are linked by a host control system is accessed or controlled.
  - The system may expose several user interfaces to serve different kinds of users. For example, a computerized library database might provide two user interfaces, one for library patrons (limited set of functions, optimized for ease of use) and the other for library personnel (wide set of functions, optimized for efficiency).
- The user interface of a **mechanical system, a vehicle or an industrial installation** is sometimes referred to as the **human-machine interface** (**HMI**). HMI is a modification of the original term MMI (man-machine interface). In practice, the abbreviation MMI is still frequently used although some may claim that MMI stands for something different now. Another abbreviation is HCI, but is more commonly used for human-computer *interaction*. Other terms used are operator interface console (OIC) and operator interface terminal (OIT). However it is abbreviated, the terms refer to the 'layer' that separates a human that is operating a machine from the machine itself.

In science fiction, HMI is sometimes used to refer to what is better described as **direct neural interface**. However, this latter usage is seeing increasing application in the real-life use of (medical) prostheses—the artificial extension that replaces a missing body part (e.g., cochlear implants).

In some circumstance computers might observe the user, and react according to their actions without specific commands. A means of tracking parts of the body is required,

and sensors noting the position of the head, direction of gaze and so on have been used experimentally. This is particularly relevant to **immersive interfaces**.

Usability

User interfaces are considered by some authors to be a prime ingredient of Computer user satisfaction.

The design of a user interface affects the amount of effort the user must expend to provide input for the system and to interpret the output of the system, and how much effort it takes to learn how to do this. Usability is the degree to which the design of a particular user interface takes into account the human psychology and physiology of the users, and makes the process of using the system effective, efficient and satisfying.

Usability is mainly a characteristic of the user interface, but is also associated with the functionalities of the product and the process to design it. It describes how well a product can be used for its intended purpose by its target users with efficiency, effectiveness, and satisfaction, also taking into account the requirements from its context of use.

User interfaces in computing

In computer science and human-computer interaction, the *user interface (of a computer program)* refers to the graphical, textual and auditory information the program presents to the user, and the control sequences (such as keystrokes with the computer keyboard, movements of the computer mouse, and selections with the touchscreen) the user employs to control the program.

Types

Currently (as of 2009) the following types of user interface are the most common:

- **Graphical user interfaces** (GUI) accept input via devices such as computer keyboard and mouse and provide articulated graphical output on the computer monitor. There are at least two different principles widely used in GUI design: Object-oriented user interfaces (OOUIs) and application oriented interfaces.
- **Web-based user interfaces** or **web user interfaces** (WUI) are a subclass of GUIs that accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Newer implementations utilize Java, AJAX, Adobe Flex, Microsoft .NET, or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser. Administrative web interfaces for web-servers, servers and networked computers are often called Control panels.

- **Touchscreens** are displays that accept input by touch of fingers or a stylus. Used in a growing amount of mobile devices and many types of point of sale, industrial processes and machines, self-service machines etc.

User interfaces that are common in various fields outside desktop computing:

- **Command line interfaces**, where the user provides the input by typing a command string with the computer keyboard and the system provides output by printing text on the computer monitor. Used by programmers and system administrators, in engineering and scientific environments, and by technically advanced personal computer users.

- **Touch user interface** are graphical user interfaces using a touchpad or touchscreen display as a combined input and output device. They supplement or replace other forms of output with haptic feedback methods. Used in computerized simulators etc.

Other types of user interfaces:

- **Attentive user interfaces** manage the user attention deciding when to interrupt the user, the kind of warnings, and the level of detail of the messages presented to the user.
- **Batch interfaces** are non-interactive user interfaces, where the user specifies all the details of the *batch job* in advance to batch processing, and receives the output when all the processing is done. The computer does not prompt for further input after the processing has started.
- **Conversational Interface Agents** attempt to personify the computer interface in the form of an animated person, robot, or other character (such as Microsoft's Clippy the paperclip), and present interactions in a conversational form.
- **Crossing-based interfaces** are graphical user interfaces in which the primary task consists in crossing boundaries instead of pointing.
- **Gesture interface** are graphical user interfaces which accept input in a form of hand gestures, or mouse gestures sketched with a computer mouse or a stylus.
- **Intelligent user interfaces** are human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture).
- **Motion tracking interfaces** monitor the user's body motions and translate them into commands, currently being developed by Apple
- **Multi-screen interfaces**, employ multiple displays to provide a more flexible interaction. This is often employed in computer game interaction in both the commercial arcades and more recently the handheld markets.
- **Noncommand user interfaces**, which observe the user to infer his / her needs and intentions, without requiring that he / she formulate explicit commands.
- **Object-oriented user interface (OOUI)**

- **Reflexive user interfaces** where the users control and redefine the entire system via the user interface alone, for instance to change its command verbs. Typically this is only possible with very rich graphic user interfaces.
- **Tangible user interfaces**, which place a greater emphasis on touch and physical environment or its element.
- **Task-Focused Interfaces** are user interfaces which address the information overload problem of the desktop metaphor by making tasks, not files, the primary unit of interaction
- **Text user interfaces** are user interfaces which output text, but accept other form of input in addition to or in place of typed command strings.
- **Voice user interfaces**, which accept input and provide output by generating voice prompts. The user input is made by pressing keys or buttons, or responding verbally to the interface.
- **Natural-Language interfaces -** Used for search engines and on webpages. User types in a question and waits for a response.
- **Zero-Input interfaces** get inputs from a set of sensors instead of querying the user with input dialogs.
- **Zooming user interfaces** are graphical user interfaces in which information objects are represented at different levels of scale and detail, and where the user can change the scale of the viewed area in order to show more detail.

- Archy, a keyboard-driven user interface by Jef Raskin, arguably more efficient than mouse-driven user interfaces for document editing and programming.

History

The history of user interfaces can be divided into the following phases according to the dominant type of user interface:

- Batch interface, 1945–1968
- Command-line user interface, 1969 to present
- Graphical user interface, 1981 to present

Consistency

A key property of a good user interface is consistency. Good user interface design is about getting a user to have a consistent set of expectations, and then meeting those expectations.

There are three important aspects to consistency.

First, the controls for different features should be presented in a consistent manner so that users can find the controls easily. For example, users find it difficult to use software when some commands are available through menus, some through icons, some through right-clicks, some under a separate button at one corner of a screen, some grouped by function, some grouped by “common,” some grouped by “advanced.” A user looking for

a command should have a consistent search strategy for finding it. The more search strategies a user has to use, the more frustrating the search will be. The more consistent the grouping, the easier the search.

Second, the "principle of least astonishment" is crucial. Various features should work in similar ways. For example, some features in Adobe Acrobat are "select tool, then select text to which apply." Others are "select text, then apply action to selection." . Commands should work the same way in all contexts.

Third, consistency counsels against user interface changes version-to-version. Change should be minimized, and forward-compatibility should be maintained. For example, the change from the menu bars of Microsoft Office 2003 to the ribbon toolbar of Microsoft Office 2007 caused mixed reactions to the redesign, intended to enhance access to the most used functions. It was said to cause "anger and frustration," and "major efforts in time, training and costs." Power users said it "takes too much time and patience to learn" the new interface. An online survey by an Excel user group reports that about 80% of respondents had a negative opinion of the change, and within that 80%, the self-estimated reduction in productivity was "about 35%".

Consistency is one quality to trade off in user interface design—not the only thing, but one of the most important. In some cases, a violation of consistency principles can provide sufficiently clear advantages that a wise and careful user interface designer may choose to violate consistency to achieve some other important goal. Generally, less mature software has fewer users who are entrenched in the status quo. Older, more broadly used software must more carefully hew to the status quo to avoid disruptive costs. The most experienced users, and the ones who derive most value from a program, are the users who tend to bear the greatest costs from change. However, of those trade-offs, consistency is one of the most important core principles, and it should be violated least often. Bad user interface design, and poorly implemented changes to an existing user interface, can impose staggering costs on users.

Modalities and modes

A **modality** is a path of communication employed by the user interface to carry input and output. Examples of modalities:

- Input — computer keyboard allows the user to enter typed text, digitizing tablet allows the user to create free-form drawing
- Output — computer monitor allows the system to display text and graphics (*vision modality*), loudspeaker allows the system to produce sound (*auditory modality*)

The user interface may employ several redundant input modalities and output modalities, allowing the user to choose which ones to use for interaction.

A **mode** is a distinct method of operation within a computer program, in which the same input can produce different perceived results depending of the state of the computer program. For example, caps lock sets an input mode in which typed letters are uppercase by default; the same typing produces lowercase letters when not in caps lock mode. Heavy use of modes often reduces the usability of a user interface, as the user must expend effort to remember current mode states, and switch between mode states as necessary.

Chapter- 5

Programmable Logic Controller

At rack, left-to-right: power supply unit PS407 4A,CPU 416-3, interface module IM 460-0 and communication processor CP 443-1.

A **programmable logic controller** (**PLC**) or **programmable controller** is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures. PLCs are used in many industries and machines. Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed or non-volatile memory. A PLC is an example of a real time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result.

History

The PLC was invented in response to the needs of the American automotive manufacturing industry. Programmable logic controllers were initially adopted by the automotive industry where software revision replaced the re-wiring of hard-wired control panels when production models changed.

Before the PLC, control, sequencing, and safety interlock logic for manufacturing automobiles was accomplished using hundreds or thousands of relays, cam timers, and drum sequencers and dedicated closed-loop controllers. The process for updating such facilities for the yearly model change-over was very time consuming and expensive, as electricians needed to individually rewire each and every relay.

In 1968 GM Hydramatic (the automatic transmission division of General Motors) issued a request for proposal for an electronic replacement for hard-wired relay systems. The winning proposal came from Bedford Associates of Bedford, Massachusetts. The first PLC, designated the 084 because it was Bedford Associates' eighty-fourth project, was the result. Bedford Associates started a new company dedicated to developing, manufacturing, selling, and servicing this new product: Modicon, which stood for MOdular DIgital CONtroller. One of the people who worked on that project was Dick Morley, who is considered to be the "father" of the PLC. The Modicon brand was sold in 1977 to Gould Electronics, and later acquired by German Company AEG and then by French Schneider Electric, the current owner.

One of the very first 084 models built is now on display at Modicon's headquarters in North Andover, Massachusetts. It was presented to Modicon by GM, when the unit was retired after nearly twenty years of uninterrupted service. Modicon used the 84 moniker at the end of its product range until the 984 made its appearance.

The automotive industry is still one of the largest users of PLCs.

Development

Early PLCs were designed to replace relay logic systems. These PLCs were programmed in "ladder logic", which strongly resembles a schematic diagram of relay logic. This

program notation was chosen to reduce training demands for the existing technicians. Other early PLCs used a form of instruction list programming, based on a stack-based logic solver.

Modern PLCs can be programmed in a variety of ways, from ladder logic to more traditional programming languages such as BASIC and C. Another method is State Logic, a very high-level programming language designed to program PLCs based on state transition diagrams.

Many early PLCs did not have accompanying programming terminals that were capable of graphical representation of the logic, and so the logic was instead represented as a series of logic expressions in some version of Boolean format, similar to Boolean algebra. As programming terminals evolved, it became more common for ladder logic to be used, for the aforementioned reasons. Newer formats such as State Logic and Function Block (which is similar to the way logic is depicted when using digital integrated logic circuits) exist, but they are still not as popular as ladder logic. A primary reason for this is that PLCs solve the logic in a predictable and repeating sequence, and ladder logic allows the programmer (the person writing the logic) to see any issues with the timing of the logic sequence more easily than would be possible in other formats.

Programming

Early PLCs, up to the mid-1980s, were programmed using proprietary programming panels or special-purpose programming terminals, which often had dedicated function keys representing the various logical elements of PLC programs. Programs were stored on cassette tape cartridges. Facilities for printing and documentation were very minimal due to lack of memory capacity. The very oldest PLCs used non-volatile magnetic core memory.

More recently, PLCs are programmed using application software on personal computers. The computer is connected to the PLC through Ethernet, RS-232, RS-485 or RS-422 cabling. The programming software allows entry and editing of the ladder-style logic. Generally the software provides functions for debugging and troubleshooting the PLC software, for example, by highlighting portions of the logic to show current status during operation or via simulation. The software will upload and download the PLC program, for backup and restoration purposes. In some models of programmable controller, the program is transferred from a personal computer to the PLC though a programming board which writes the program into a removable chip such as an EEPROM or EPROM.

Functionality

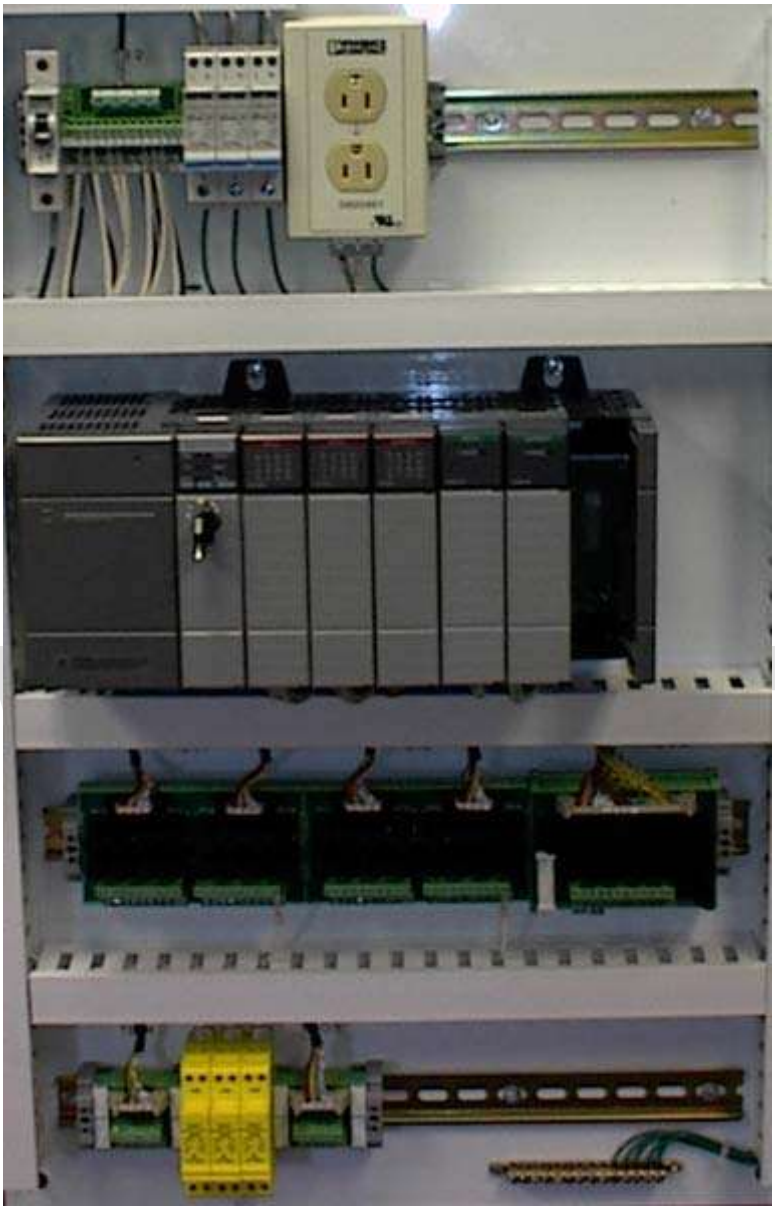
The functionality of the PLC has evolved over the years to include sequential relay control, motion control, process control, distributed control systems and networking. The data handling, storage, processing power and communication capabilities of some modern PLCs are approximately equivalent to desktop computers. PLC-like programming combined with remote I/O hardware, allow a general-purpose desktop

computer to overlap some PLCs in certain applications. Regarding the practicality of these desktop computer based logic controllers, it is important to note that they have not been generally accepted in heavy industry because the desktop computers run on less stable operating systems than do PLCs, and because the desktop computer hardware is typically not designed to the same levels of tolerance to temperature, humidity, vibration, and longevity as the processors used in PLCs. In addition to the hardware limitations of desktop based logic, operating systems such as Windows do not lend themselves to deterministic logic execution, with the result that the logic may not always respond to changes in logic state or input status with the extreme consistency in timing as is expected from PLCs. Still, such desktop logic applications find use in less critical situations, such as laboratory automation and use in small facilities where the application is less demanding and critical, because they are generally much less expensive than PLCs.

In more recent years, small products called PLRs (programmable logic relays), and also by similar names, have become more common and accepted. These are very much like PLCs, and are used in light industry where only a few points of I/O (i.e. a few signals coming in from the real world and a few going out) are involved, and low cost is desired. These small devices are typically made in a common physical size and shape by several manufacturers, and branded by the makers of larger PLCs to fill out their low end product range. Popular names include PICO Controller, NANO PLC, and other names implying very small controllers. Most of these have between 8 and 12 digital inputs, 4 and 8 digital outputs, and up to 2 analog inputs. Size is usually about 4" wide, 3" high, and 3" deep. Most such devices include a tiny postage stamp sized LCD screen for viewing simplified ladder logic (only a very small portion of the program being visible at a given time) and status of I/O points, and typically these screens are accompanied by a 4-way rocker push-button plus four more separate push-buttons, similar to the key buttons on a VCR remote control, and used to navigate and edit the logic. Most have a small plug for connecting via RS-232 or RS-485 to a personal computer so that programmers can use simple Windows applications for programming instead of being forced to use the tiny LCD and push-button set for this purpose. Unlike regular PLCs that are usually modular and greatly expandable, the PLRs are usually not modular or expandable, but their price can be two orders of magnitude less than a PLC and they still offer robust design and deterministic execution of the logic.

PLC Topics

Features



Control panel with PLC (grey elements in the center). The unit consists of separate elements, from left to right; power supply, controller, relay units for in- and output

The main difference from other computers is that PLCs are armored for severe conditions (such as dust, moisture, heat, cold) and have the facility for extensive input/output (I/O) arrangements. These connect the PLC to sensors and actuators. PLCs read limit switches, analog process variables (such as temperature and pressure), and the positions of complex positioning systems. Some use machine vision. On the actuator side, PLCs operate electric motors, pneumatic or hydraulic cylinders, magnetic relays, solenoids, or analog outputs. The input/output arrangements may be built into a simple PLC, or the PLC may have external I/O modules attached to a computer network that plugs into the PLC.

System scale

A small PLC will have a fixed number of connections built in for inputs and outputs. Typically, expansions are available if the base model has insufficient I/O.

Modular PLCs have a chassis (also called a rack) into which are placed modules with different functions. The processor and selection of I/O modules is customised for the particular application. Several racks can be administered by a single processor, and may have thousands of inputs and outputs. A special high speed serial I/O link is used so that racks can be distributed away from the processor, reducing the wiring costs for large plants.

User interface

PLCs may need to interact with people for the purpose of configuration, alarm reporting or everyday control.

A Human-Machine Interface (HMI) is employed for this purpose. HMIs are also referred to as MMIs (Man Machine Interface) and GUIs (Graphical User Interface).

A simple system may use buttons and lights to interact with the user. Text displays are available as well as graphical touch screens. More complex systems use a programming and monitoring software installed on a computer, with the PLC connected via a communication interface.

Communications

PLCs have built in communications ports, usually 9-pin RS-232, but optionally EIA-485 or Ethernet. Modbus, BACnet or DF1 is usually included as one of the communications protocols. Other options include various fieldbuses such as DeviceNet or Profibus. Other communications protocols that may be used are listed in the List of automation protocols.

Most modern PLCs can communicate over a network to some other system, such as a computer running a SCADA (Supervisory Control And Data Acquisition) system or web browser.

PLCs used in larger I/O systems may have peer-to-peer (P2P) communication between processors. This allows separate parts of a complex process to have individual control while allowing the subsystems to co-ordinate over the communication link. These communication links are also often used for HMI devices such as keypads or PC-type workstations.

Programming

PLC programs are typically written in a special application on a personal computer, then downloaded by a direct-connection cable or over a network to the PLC. The program is stored in the PLC either in battery-backed-up RAM or some other non-volatile flash memory. Often, a single PLC can be programmed to replace thousands of relays.

Under the IEC 61131-3 standard, PLCs can be programmed using standards-based programming languages. A graphical programming notation called Sequential Function Charts is available on certain programmable controllers. Initially most PLCs utilized Ladder Logic Diagram Programming, a model which emulated electromechanical control panel devices (such as the contact and coils of relays) which PLCs replaced. This model remains common today.

IEC 61131-3 currently defines five programming languages for programmable control systems: FBD (Function block diagram), LD (Ladder diagram), ST (Structured text, similar to the Pascal programming language), IL (Instruction list, similar to assembly language) and SFC (Sequential function chart). These techniques emphasize logical organization of operations.

While the fundamental concepts of PLC programming are common to all manufacturers, differences in I/O addressing, memory organization and instruction sets mean that PLC programs are never perfectly interchangeable between different makers. Even within the same product line of a single manufacturer, different models may not be directly compatible.

PLC compared with other control systems

Allen-Bradley PLC installed in a control panel

PLCs are well-adapted to a range of automation tasks. These are typically industrial processes in manufacturing where the cost of developing and maintaining the automation system is high relative to the total cost of the automation, and where changes to the system would be expected during its operational life. PLCs contain input and output devices compatible with industrial pilot devices and controls; little electrical design is required, and the design problem centers on expressing the desired sequence of operations. PLC applications are typically highly customized systems so the cost of a packaged PLC is low compared to the cost of a specific custom-built controller design. On the other hand, in the case of mass-produced goods, customized control systems are economic due to the lower cost of the components, which can be optimally chosen instead of a "generic" solution, and where the non-recurring engineering charges are spread over thousands or millions of units.

For high volume or very simple fixed automation tasks, different techniques are used. For example, a consumer dishwasher would be controlled by an electromechanical cam timer costing only a few dollars in production quantities.

A microcontroller-based design would be appropriate where hundreds or thousands of units will be produced and so the development cost (design of power supplies, input/output hardware and necessary testing and certification) can be spread over many sales, and where the end-user would not need to alter the control. Automotive applications are an example; millions of units are built each year, and very few end-users alter the programming of these controllers. However, some specialty vehicles such as transit busses economically use PLCs instead of custom-designed controls, because the volumes are low and the development cost would be uneconomic.

Very complex process control, such as used in the chemical industry, may require algorithms and performance beyond the capability of even high-performance PLCs. Very high-speed or precision controls may also require customized solutions; for example, aircraft flight controls.

Programmable controllers are widely used in motion control, positioning control and torque control. Some manufacturers produce motion control units to be integrated with PLC so that G-code (involving a CNC machine) can be used to instruct machine movements.

PLCs may include logic for single-variable feedback analog control loop, a "proportional, integral, derivative" or "PID controller". A PID loop could be used to control the temperature of a manufacturing process, for example. Historically PLCs were usually configured with only a few analog control loops; where processes required hundreds or thousands of loops, a distributed control system (DCS) would instead be used. As PLCs have become more powerful, the boundary between DCS and PLC applications has become less distinct.

PLCs have similar functionality as Remote Terminal Units. An RTU, however, usually does not support control algorithms or control loops. As hardware rapidly becomes more

powerful and cheaper, RTUs, PLCs and DCSs are increasingly beginning to overlap in responsibilities, and many vendors sell RTUs with PLC-like features and vice versa. The industry has standardized on the IEC 61131-3 functional block language for creating programs to run on RTUs and PLCs, although nearly all vendors also offer proprietary alternatives and associated development environments.

Digital and analog signals

Digital or discrete signals behave as binary switches, yielding simply an On or Off signal (1 or 0, True or False, respectively). Push buttons, limit switches, and photoelectric sensors are examples of devices providing a discrete signal. Discrete signals are sent using either voltage or current, where a specific range is designated as *On* and another as *Off*. For example, a PLC might use 24 V DC I/O, with values above 22 V DC representing *On*, values below 2VDC representing *Off*, and intermediate values undefined. Initially, PLCs had only discrete I/O.

Analog signals are like volume controls, with a range of values between zero and full-scale. These are typically interpreted as integer values (counts) by the PLC, with various ranges of accuracy depending on the device and the number of bits available to store the data. As PLCs typically use 16-bit signed binary processors, the integer values are limited between -32,768 and +32,767. Pressure, temperature, flow, and weight are often represented by analog signals. Analog signals can use voltage or current with a magnitude proportional to the value of the process signal. For example, an analog 0 - 10 V input or 4-20 mA would be converted into an integer value of 0 - 32767.

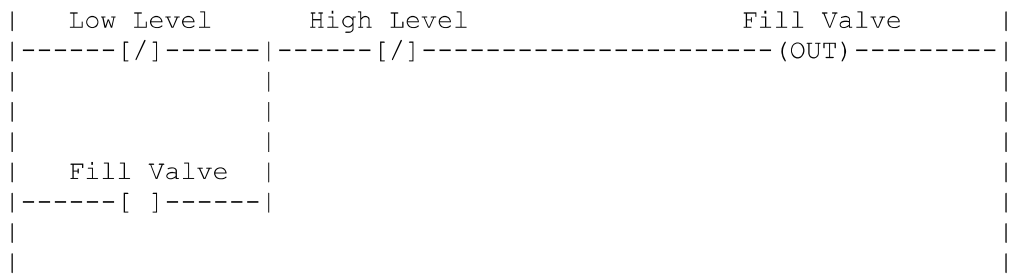
Current inputs are less sensitive to electrical noise (i.e. from welders or electric motor starts) than voltage inputs.

Example

As an example, say a facility needs to store water in a tank. The water is drawn from the tank by another system, as needed, and our example system must manage the water level in the tank.

Using only digital signals, the PLC has two digital inputs from float switches (Low Level and High Level). When the water level is above the switch it closes a contact and passes a signal to an input. The PLC uses a digital output to open and close the inlet valve into the tank.

When the water level drops enough so that the Low Level float switch is off (down), the PLC will open the valve to let more water in. Once the water level rises enough so that the High Level switch is on (up), the PLC will shut the inlet to stop the water from overflowing. This rung is an example of seal-in (latching) logic. The output is sealed in until some condition breaks the circuit.



An analog system might use a water pressure sensor or a load cell, and an adjustable (throttling) dripping out of the tank, the valve adjusts to slowly drip water back into the tank.

In this system, to avoid 'flutter' adjustments that can wear out the valve, many PLCs incorporate "hysteresis" which essentially creates a "deadband" of activity. A technician adjusts this deadband so the valve moves only for a significant change in rate. This will in turn minimize the motion of the valve, and reduce its wear.

A real system might combine both approaches, using float switches and simple valves to prevent spills, and a rate sensor and rate valve to optimize refill rates and prevent water hammer. Backup and maintenance methods can make a real system very complicated.

Chapter- 6

SCADA (Automation Tool)

SCADA stands for *supervisory control and data acquisition*. It generally refers to industrial control systems: computer systems that monitor and control industrial, infrastructure, or facility-based processes, as described below:

- Industrial processes include those of manufacturing, production, power generation, fabrication, and refining, and may run in continuous, batch, repetitive, or discrete modes.
- Infrastructure processes may be public or private, and include water treatment and distribution, wastewater collection and treatment, oil and gas pipelines, electrical power transmission and distribution, Wind farms, civil defense siren systems, and large communication systems.
- Facility processes occur both in public facilities and private ones, including buildings, airports, ships, and space stations. They monitor and control HVAC, access, and energy consumption.

Common system components

A SCADA System usually consists of the following subsystems:

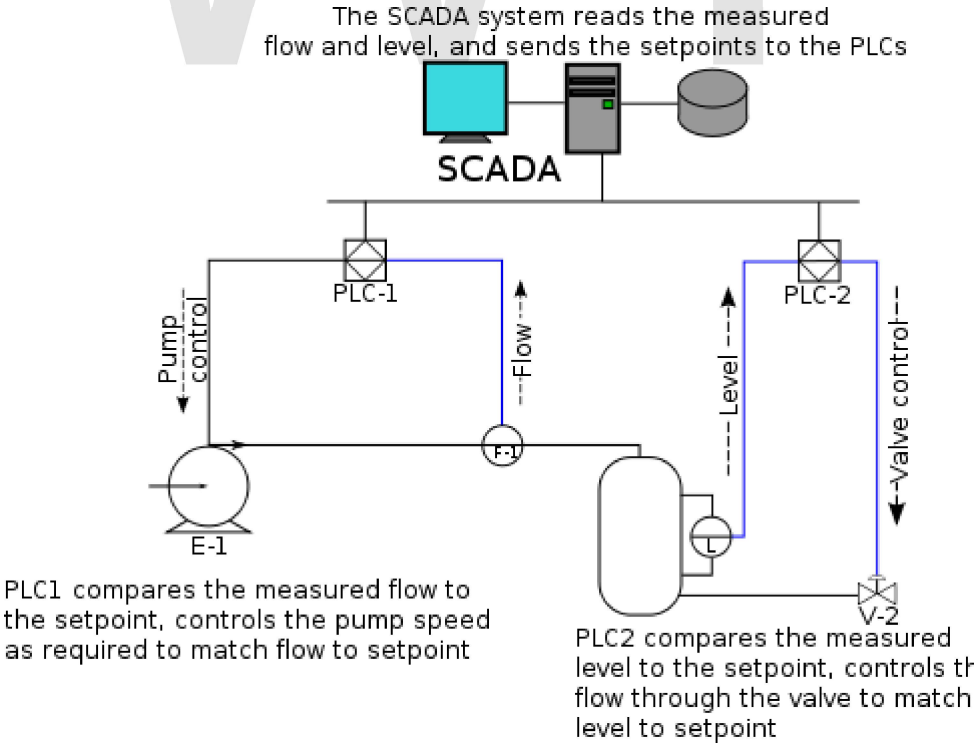
- A Human-Machine Interface or HMI is the apparatus which presents process data to a human operator, and through this, the human operator monitors and controls the process.
- A supervisory (computer) system, gathering (acquiring) data on the process and sending commands (control) to the process.
- Remote Terminal Units (RTUs) connecting to sensors in the process, converting sensor signals to digital data and sending digital data to the supervisory system.
- Programmable Logic Controller (PLCs) used as field devices because they are more economical, versatile, flexible, and configurable than special-purpose RTUs.
- Communication infrastructure connecting the supervisory system to the Remote Terminal Units.

Supervision vs. control

There is, in several industries, considerable confusion over the differences between SCADA systems and distributed control systems (DCS). Generally speaking, a SCADA system always refers to a system that *coordinates*, but does not *control* processes in real time. The discussion on real-time control is muddied somewhat by newer telecommunications technology, enabling reliable, low latency, high speed communications over wide areas. Most differences between SCADA and DCS are culturally determined and can usually be ignored. As communication infrastructures with higher capacity become available, the difference between SCADA and DCS will fade.

Systems concepts

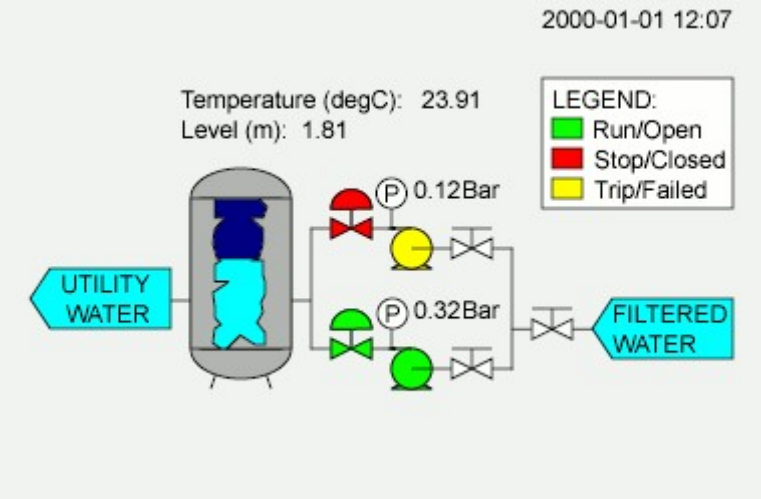
The term SCADA usually refers to centralized systems which monitor and control entire sites, or complexes of systems spread out over large areas (anything between an industrial plant and a country). Most control actions are performed automatically by Remote Terminal Units ("RTUs") or by programmable logic controllers ("PLCs"). Host control functions are usually restricted to basic overriding or *supervisory* level intervention. For example, a PLC may control the flow of cooling water through part of an industrial process, but the SCADA system may allow operators to change the set points for the flow, and enable alarm conditions, such as loss of flow and high temperature, to be displayed and recorded. The feedback control loop passes through the RTU or PLC, while the SCADA system monitors the overall performance of the loop.



Data acquisition begins at the RTU or PLC level and includes meter readings and equipment status reports that are communicated to SCADA as required. Data is then compiled and formatted in such a way that a control room operator using the HMI can make supervisory decisions to adjust or override normal RTU (PLC) controls. Data may also be fed to a Historian, often built on a commodity Database Management System, to allow trending and other analytical auditing.

SCADA systems typically implement a distributed database, commonly referred to as a *tag database*, which contains data elements called *tags* or *points*. A point represents a single input or output value monitored or controlled by the system. Points can be either "hard" or "soft". A hard point represents an actual input or output within the system, while a soft point results from logic and math operations applied to other points. (Most implementations conceptually remove the distinction by making every property a "soft" point expression, which may, in the simplest case, equal a single hard point.) Points are normally stored as value-timestamp pairs: a value, and the timestamp when it was recorded or calculated. A series of value-timestamp pairs gives the history of that point. It's also common to store additional metadata with tags, such as the path to a field device or PLC register, design time comments, and alarm information.

Human Machine Interface



Typical Basic SCADA Animations

A Human-Machine Interface or HMI is the apparatus which presents process data to a human operator, and through which the human operator controls the process.

An HMI is usually linked to the SCADA system's databases and software programs, to provide trending, diagnostic data, and management information such as scheduled maintenance procedures, logistic information, detailed schematics for a particular sensor or machine, and expert-system troubleshooting guides.

The HMI system usually presents the information to the operating personnel graphically, in the form of a mimic diagram. This means that the operator can see a schematic representation of the plant being controlled. For example, a picture of a pump connected to a pipe can show the operator that the pump is running and how much fluid it is pumping through the pipe at the moment. The operator can then switch the pump off. The HMI software will show the flow rate of the fluid in the pipe decrease in real time. Mimic diagrams may consist of line graphics and schematic symbols to represent process elements, or may consist of digital photographs of the process equipment overlain with animated symbols.

The HMI package for the SCADA system typically includes a drawing program that the operators or system maintenance personnel use to change the way these points are represented in the interface. These representations can be as simple as an on-screen traffic light, which represents the state of an actual traffic light in the field, or as complex as a multi-projector display representing the position of all of the elevators in a skyscraper or all of the trains on a railway.

An important part of most SCADA implementations is alarm handling. The system monitors whether certain alarm conditions are satisfied, to determine when an alarm event has occurred. Once an alarm event has been detected, one or more actions are taken (such as the activation of one or more alarm indicators, and perhaps the generation of email or text messages so that management or remote SCADA operators are informed). In many cases, a SCADA operator may have to acknowledge the alarm event; this may deactivate some alarm indicators, whereas other indicators remain active until the alarm conditions are cleared. Alarm conditions can be explicit - for example, an alarm point is a digital status point that has either the value NORMAL or ALARM that is calculated by a formula based on the values in other analogue and digital points - or implicit: the SCADA system might automatically monitor whether the value in an analogue point lies outside high and low limit values associated with that point. Examples of alarm indicators include a siren, a pop-up box on a screen, or a coloured or flashing area on a screen (that might act in a similar way to the "fuel tank empty" light in a car); in each case, the role of the alarm indicator is to draw the operator's attention to the part of the system 'in alarm' so that appropriate action can be taken. In designing SCADA systems, care is needed in coping with a cascade of alarm events occurring in a short time, otherwise the underlying cause (which might not be the earliest event detected) may get lost in the noise. Unfortunately, when used as a noun, the word 'alarm' is used rather loosely in the industry; thus, depending on context it might mean an alarm point, an alarm indicator, or an alarm event.

Hardware solutions

SCADA solutions often have Distributed Control System (DCS) components. Use of "smart" RTUs or PLCs, which are capable of autonomously executing simple logic processes without involving the master computer, is increasing. A functional block programming language, IEC 61131-3 (Ladder Logic), is frequently used to create programs which run on these RTUs and PLCs. Unlike a procedural language such as the

C programming language or FORTRAN, IEC 61131-3 has minimal training requirements by virtue of resembling historic physical control arrays. This allows SCADA system engineers to perform both the design and implementation of a program to be executed on an RTU or PLC. A Programmable automation controller (PAC) is a compact controller that combines the features and capabilities of a PC-based control system with that of a typical PLC. PACs are deployed in SCADA systems to provide RTU and PLC functions. In many electrical substation SCADA applications, "distributed RTUs" use information processors or station computers to communicate with digital protective relays, PACs, and other devices for I/O, and communicate with the SCADA master in lieu of a traditional RTU.

Since about 1998, virtually all major PLC manufacturers have offered integrated HMI/SCADA systems, many of them using open and non-proprietary communications protocols. Numerous specialized third-party HMI/SCADA packages, offering built-in compatibility with most major PLCs, have also entered the market, allowing mechanical engineers, electrical engineers and technicians to configure HMIs themselves, without the need for a custom-made program written by a software developer.

Remote Terminal Unit (RTU)

The RTU connects to physical equipment. Typically, an RTU converts the electrical signals from the equipment to digital values such as the open/closed status from a switch or a valve, or measurements such as pressure, flow, voltage or current. By converting and sending these electrical signals out to equipment the RTU can control equipment, such as opening or closing a switch or a valve, or setting the speed of a pump.

Supervisory Station

The term "Supervisory Station" refers to the servers and software responsible for communicating with the field equipment (RTUs, PLCs, etc), and then to the HMI software running on workstations in the control room, or elsewhere. In smaller SCADA systems, the master station may be composed of a single PC. In larger SCADA systems, the master station may include multiple servers, distributed software applications, and disaster recovery sites. To increase the integrity of the system the multiple servers will often be configured in a dual-redundant or hot-standby formation providing continuous control and monitoring in the event of a server failure.

Operational philosophy

For some installations, the costs that would result from the control system failing are extremely high. Possibly even lives could be lost. Hardware for some SCADA systems is ruggedized to withstand temperature, vibration, and voltage extremes, but in most critical installations reliability is enhanced by having redundant hardware and communications channels, up to the point of having multiple fully equipped control centres. A failing part can be quickly identified and its functionality automatically taken over by backup hardware. A failed part can often be replaced without interrupting the process. The

reliability of such systems can be calculated statistically and is stated as the mean time to failure, which is a variant of mean time between failures. The calculated mean time to failure of such high reliability systems can be on the order of centuries.

Communication infrastructure and methods

SCADA systems have traditionally used combinations of radio and direct serial or modem connections to meet communication requirements, although Ethernet and IP over SONET / SDH is also frequently used at large sites such as railways and power stations. The remote management or monitoring function of a SCADA system is often referred to as telemetry.

This has also come under threat with some customers wanting SCADA data to travel over their pre-established corporate networks or to share the network with other applications. The legacy of the early low-bandwidth protocols remains, though. SCADA protocols are designed to be very compact and many are designed to send information to the master station only when the master station polls the RTU. Typical legacy SCADA protocols include Modbus RTU, RP-570, Profibus and Conitel. These communication protocols are all SCADA-vendor specific but are widely adopted and used. Standard protocols are IEC 60870-5-101 or 104, IEC 61850 and DNP3. These communication protocols are standardized and recognized by all major SCADA vendors. Many of these protocols now contain extensions to operate over TCP/IP. It is good security engineering practice to avoid connecting SCADA systems to the Internet so the attack surface is reduced.

RTUs and other automatic controller devices were being developed before the advent of industry wide standards for interoperability. The result is that developers and their management created a multitude of control protocols. Among the larger vendors, there was also the incentive to create their own protocol to "lock in" their customer base. A list of automation protocols is being compiled here.

Recently, OLE for Process Control (OPC) has become a widely accepted solution for intercommunicating different hardware and software, allowing communication even between devices originally not intended to be part of an industrial network.

SCADA architectures

TM 5-601

TECHNICAL MANUAL

SUPERVISORY CONTROL AND DATA ACQUISITION (SCADA) SYSTEMS FOR COMMAND, CONTROL, COMMUNICATIONS, COMPUTER, INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE (C4ISR) FACILITIES

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

HEADQUARTERS, DEPARTMENT OF THE ARMY
21 JANUARY 2006

The United States Army's Training Manual 5-601 covers "SCADA Systems for C4ISR Facilities".

SCADA systems have evolved through 3 generations as follows:

First generation: "Monolithic"

In the first generation, computing was done by mainframe computers. Networks did not exist at the time SCADA was developed. Thus SCADA systems were independent systems with no connectivity to other systems. Wide Area Networks were later designed by RTU vendors to communicate with the RTU. The communication protocols used were often proprietary at that time. The first-generation SCADA system was redundant since a

back-up mainframe system was connected at the bus level and was used in the event of failure of the primary mainframe system.

Second generation: "Distributed"

The processing was distributed across multiple stations which were connected through a LAN and they shared information in real time. Each station was responsible for a particular task thus making the size and cost of each station less than the one used in First Generation. The network protocols used were still mostly proprietary, which led to significant security problems for any SCADA system that received attention from a hacker. Since the protocols were proprietary, very few people beyond the developers and hackers knew enough to determine how secure a SCADA installation was. Since both parties had invested interests in keeping security issues quiet, the security of a SCADA installation was often badly overestimated, if it was considered at all.

Third generation: "Networked"

These are the current generation SCADA systems which use open system architecture rather than a vendor-controlled proprietary environment. The SCADA system utilizes open standards and protocols, thus distributing functionality across a WAN rather than a LAN. It is easier to connect third party peripheral devices like printers, disk drives, and tape drives due to the use of open architecture. WAN protocols such as Internet Protocol (IP) are used for communication between the master station and communications equipment. Due to the usage of standard protocols and the fact that many networked SCADA systems are accessible from the Internet, the systems are potentially vulnerable to remote cyber-attacks. On the other hand, the usage of standard protocols and security techniques means that standard security improvements are applicable to the SCADA systems, assuming they receive timely maintenance and updates.

==Trends in SCADA=Reliability Corporation (NERC) has specified that electrical system data should be time-tagged to the nearest millisecond. Electrical system SCADA systems provide this Sequence of events recorder function, using Radio clocks to synchronize the RTU or distributed RTU clocks.

SCADA systems are coming in line with standard networking technologies. Ethernet and TCP/IP based protocols are replacing the older proprietary standards. Although certain characteristics of frame-based network communication technology (determinism, synchronization, protocol selection, environment suitability) have restricted the adoption of Ethernet in a few specialized applications, the vast majority of markets have accepted Ethernet networks for HMI/SCADA.

A few vendors have begun offering application specific SCADA systems hosted on remote platforms over the Internet. This removes the need to install and commission systems at the end-user's facility and takes advantage of security features already available in Internet technology, VPNs and SSL. Some concerns include security, Internet connection reliability, and latency.

SCADA systems are becoming increasingly ubiquitous. Thin clients, web portals, and web based products are gaining popularity with most major vendors. The increased convenience of end users viewing their processes remotely introduces security considerations. While these considerations are already considered solved in other sectors of Internet services, not all entities responsible for deploying SCADA systems have understood the changes in accessibility and threat scope implicit in connecting a system to the Internet.

Security issues

The move from proprietary technologies to more standardized and open solutions together with the increased number of connections between SCADA systems and office networks and the Internet has made them more vulnerable to attacks. Consequently, the security of SCADA-based systems has come into question as they are increasingly seen as extremely vulnerable to cyberwarfare/cyberterrorism attacks.

In particular, security researchers are concerned about:

- the lack of concern about security and authentication in the design, deployment and operation of existing SCADA networks
- the belief that SCADA systems have the benefit of security through obscurity through the use of specialized protocols and proprietary interfaces
- the belief that SCADA networks are secure because they are physically secured
- the belief that SCADA networks are secure because they are disconnected from the Internet

SCADA systems are used to control and monitor physical processes, examples of which are transmission of electricity, transportation of gas and oil in pipelines, water distribution, traffic lights, and other systems used as the basis of modern society. The security of these SCADA systems is important because compromise or destruction of these systems would impact multiple areas of society far removed from the original compromise. For example, a blackout caused by a compromised electrical SCADA system would cause financial losses to all the customers that received electricity from that source. How security will affect legacy SCADA and new deployments remains to be seen.

There are two distinct threats to a modern SCADA system. First is the threat of unauthorized access to the control software, whether it be human access or changes induced intentionally or accidentally by virus infections and other software threats residing on the control host machine. Second is the threat of packet access to the network segments hosting SCADA devices. In many cases, there is rudimentary or no security on the actual packet control protocol, so anyone who can send packets to the SCADA device can control it. In many cases SCADA users assume that a VPN is sufficient protection and are unaware that physical access to SCADA-related network jacks and switches provides the ability to totally bypass all security on the control software and fully control those SCADA networks. These kinds of physical access attacks bypass firewall and VPN

security and are best addressed by endpoint-to-endpoint authentication and authorization such as are commonly provided in the non-SCADA world by in-device SSL or other cryptographic techniques.

Many SCADA systems are deployed in critical infrastructures. Attacks to these systems can impact public health and safety. A famous example of a computer-based attack to a SCADA system is the attack carried out on Maroochy Shire Council's sewage control system in Queensland, Australia. Shortly after a contractor had installed the system in January 2000, it began to experience unexplainable problems. Pumps did not run when needed, alarms were not reported, and extensive flooding with sewage occurred in a nearby park and contamination of a river. After careful observation, it was determined that sewage valves were opening of their own accord. This was believed to be a system bug, until careful monitoring of the system logs revealed that these malfunctions were the result of a cyber attack. After 46 reported attacks, the culprit was identified as an disgruntled ex-employee of the contractor company that installed the system, and who was trying to be hired back to solve the problem.

Many vendors of SCADA and control products have begun to address these risks in a basic sense by developing lines of specialized industrial firewall and VPN solutions for TCP/IP-based SCADA networks as well as external SCADA monitoring and recording equipment. . Additionally, application whitelisting solutions are being implemented because of their ability to prevent malware and unauthorized application changes without the performance impacts of traditional antivirus scans . Also, the ISA Security Compliance Institute (ISCI) is emerging to formalize SCADA security testing starting as soon as 2009. ISCI is conceptually similar to private testing and certification that has been performed by vendors since 2007. Eventually, standards being defined by ISA99 WG4 will supersede the initial industry consortia efforts, but probably not before 2011.

The increased interest in SCADA vulnerabilities has resulted in vulnerability researchers discovering vulnerabilities in commercial SCADA software and more general offensive SCADA techniques presented to the general security community. In electric and gas utility SCADA systems, the vulnerability of the large installed base of wired and wireless serial communications links is addressed in some cases by applying bump-in-the-wire devices that employ authentication and Advanced Encryption Standard encryption rather than replacing all existing nodes.

In June 2010, VirusBlokAda reported the first detection of malware that attacks SCADA systems (Siemens' WinCC/PCS7 systems) running on Windows operating systems. The malware is called Stuxnet and uses four zero-day attacks to install a rootkit which in turn logs in to the SCADA's database and steals design and control files The malware is also capable of changing the control system and hiding those changes. The malware was found by an anti-virus security company on 14 systems with the majority in Iran.

Chapter- 7

Other Tools of Automation

Programmable automation controller

A **programmable automation controller (PAC)** is a compact controller that combines the features and capabilities of a PC-based control system with that of a typical programmable logic controller (PLC). PACs are most often used in industrial settings for process control, data acquisition, remote equipment monitoring, machine vision, and motion control. Additionally, because they function and communicate over popular network interface protocols like TCP/IP, OLE for process control (OPC) and SMTP, PACs are able to transfer data from the machines they control to other machines and components in a networked control system or to application software and databases.

Programmable Automation Controller with Multiple Ethernet and Serial Ports

PAC Origins

The ARC Advisory Group, an analyst group focused on the manufacturing industry, is generally credited for originating the term "PAC". It was first coined in 2001 as a way to help users of control hardware better define their needs, and to give the leading control hardware vendors a term to more clearly communicate the capabilities of their products.

PAC - PLC Comparison

Generally, PACs and PLCs serve the same purpose. Both are primarily used to perform automation, process control, and data acquisition functions such as digital and analog control, serial string handling, PID, motion control, and machine vision. The parameters within which PACs operate to achieve this, however, sometimes run counter to how a PLC functions.

Unlike PLCs, PACS offer open, modular architectures, the rationale being that because most industrial applications are customized, the control hardware used for them needs to allow engineers to pick and choose the other components in the control system architecture without having to worry whether or not they will be compatible with the controller.

PACs and PLCs are also programmed differently. PLCs are often programmed in ladder logic, a graphical programming language resembling the rails and rungs of ladders that is designed to emulate old electrical relay wiring diagrams. PAC control programs are usually developed with more generic software tools that permit the designed program to be shared across several different machines, processors, HMI terminals or other components in the control system architecture.

PAC processing and I/O scanning is also very different. Unlike PLCs, which constantly scan all the I/O inputs in the control system at very high rates of speed, PACs utilize a single tagname database and a logical address system to identify and map I/O points as needed.

Well known control hardware vendors offering PACs include General Electric, National Instruments, Opto 22, and Allen Bradley.

Laboratory information management system

A **Laboratory Information Management System** or **Laboratory Integration Management Solution** (LIMS) is a software system used in laboratories for the integraton of all laboratory softwares, instruments, and the management of samples, laboratory users, standards and other laboratory functions such as QA/QC Quality Assurance and Quality Control, sample planning, invoicing, plate management, and workflow automation.

A Laboratory Information Management System and a Process Development Execution System (PDES) perform similar functions. A LIMS will generally target environmental, research or commercial analysis, such as pharmaceutical or petrochemical work, whereas a LIS tends to cover the clinical market (hospitals and other clinical labs). A PDES normally addresses a wider scope: including, for example, virtual manufacturing techniques, while not necessarily integrating with laboratory equipment.

LIMS implementations may also support information gathering, decision making, calculation, review and release into the workplace and away from the office. More recently, LIMS products are starting to expand into Electronic Laboratory Notebooks (ELN), assay data management, data mining and data analysis. This broader set of capabilities enables the realization of translational medicine completely within a single software solution.

Technology

Laboratory information management

Sample Management

The core function of LIMS is the management of samples. This typically is initiated when a sample is received in the laboratory at which point the sample will be registed in the LIMS. This registration process may involve accessioning the sample and producing barcodes to affix to the sample container. Various other parameters will usually be recorded such as clinical or phenotypic information corresponding with the sample. The LIMS will then track chain of custody as well as sample location. Location tracking usually involves assigning the sample to a particular freezer location such as a shelf/rack/box/row/column. Other event tracking may be required such as freeze\thaw cycles that a sample undergoes in the laboratory.

Modern LIMS have implemented extensive configurability as each laboratories needs for tracking additional data points can vary widely. LIMS vendors cannot typically make assumptions about what these data tracking needs are and therefore need to be adaptable to each environment. LIMS users may also have regulatory concerns to comply with such as CLIA, HIPAA, GLP and FDA specifications and this can affect certain aspects of sample management in a LIMS solution. One key to compliance with many of these standards is audit logging of all changes to LIMS data, and in some cases a full electronic signature system is required for rigorous tracking of field level changes to LIMS data.

LIMS Roles

One may configure a LIMS whereby users are assigned roles or groups. Typically the role of a user will dictate their access to specific data records in the LIMS. Each user account is protected by security mechanisms such as a user id and a password. Users may have customized interfaces based on their role in the organization. For example, a laboratory manager might have full access to all of a LIMS functions and data, whereas

technicians might have access only to data and functionality needed for their individual work-tasks.

Instrument Data Management

Most LIMS offer some capability for integration with instruments. A LIMS may create control files that are "fed" into the instrument and direct its operation on some physical item such as a sample tube or sample plate. The LIMS may then import instrument results files to extract QC or results data for assessment of the operation on the sample(s). Data owners may access the resulting stored information at any time.

A relatively new development in LIMS products is the ability to import and manage raw assay data results. Modern targeted assays such as qPCR and deep sequencing can produce tens of thousands of data points per sample. Furthermore, in the case of drug and diagnostic development as many as 12 or more assays may be run for each sample. In order to track this data, a LIMS solution needs to be adaptable to many different assay formats at both the data layer and import creation layer, while maintaining a high level of overall performance. Some LIMS products address this by simply attaching assay data as BLOB's to samples, but this limits the utility of that data in data mining and downstream analysis.

Web-based technology

A LIMS can use many delivery technologies. Web-based LIMS implementations, as well as Java based solutions, often require no special client-side installation resulting in less IT involvement in their deployment. An excpetion is web based solutions that are based on .NET technologies and require a special plug-in on the client and may be limited to Microsoft only browsers. This can lead to issues where there is a high penetration of Apple and Linux usage by lab technicians or researchers. Another concern regarding web based deployment is possible exploitation by hackers where highly sensitive laboratory and research data may be compromised. Even with modern security methods in place, deploying a LIMS solution "outside the firewall" of an organization opens the LIMS to potential instrusion.

Some vendors are beginning to offer hosted LIMS solutions (SaaS) on a rental basis. These solutions tend to be less configurable than on premise solutions and are therefore considered for less demanding implementation such as laboratories with few users and limited sample processing volumes.

LIMS Configurability

LIMS implementations are notorious for often being lengthy and costly. This is due in part to the diversity of requirements within each lab, but also to the inflexible nature of LIMS products for adapting to these widely varying requirements. Newer LIMS solutions are beginning to emerge that take advantage of modern techniques in software design that are inherently more configurable and adaptable, particularly at the data layer, than prior

solutions. This means not only that implementations are much faster, but also that the costs are lower and the risk of obsolescence are minimized.

Standards covered by LIMS

A LIMS covers standards such as:

Title 21 CFR Part 11

Title 21 CFR Part 11 of the Code of Federal Regulations deals with the Food and Drug Administration (FDA) guidelines on electronic records and electronic signatures in the United States. **Part 11**, as it is commonly called, defines the criteria under which electronic records and electronic signatures are considered to be trustworthy, reliable and equivalent to paper records.

Practically speaking, Part 11 requires drug makers, medical device manufacturers, biotech companies, biologics developers, and other FDA-regulated industries, with some specific exceptions, to implement controls, including audits, system validations, audit trails, electronic signatures, and documentation for software and systems involved in processing electronic data that are (a) required to be maintained by the FDA predicate rules or (b) used to demonstrate compliance to a predicate rule.

The rule also applies to submissions made to the FDA in electronic format (e.g., a New Drug Application) but not to paper submissions by electronic methods (i.e., faxes). It specifically does not require the 21CFR1 requirement for record retention for tracebacks by food manufacturers. Most food manufacturers are not otherwise explicitly required to keep detailed records, but electronic documentation kept for HACCP and similar requirements must meet these requirements.

As of 2007, broad sections of the regulation have been challenged as excessive, and the FDA has stated in guidance that it will exercise enforcement discretion on many parts of the rule. This has led to confusion on exactly what is required, and the rule is being revised. (An update was posted on April 1, 2010 on the FDA Website). In practice, the requirements on access controls are the only part routinely enforced. The "predicate rules" which required the records to be kept in the first place are still in effect. If electronic records are illegible, inaccessible, or corrupted the manufacturers are still subject to those requirements.

If a regulated firm keeps "hard copies" of all required records, the paper documents can be considered to be the authoritative document for regulatory purposes and the computer system need not meet these requirements. Firms should be careful to make a claim that "hard copies" of required records are authoritative document. In order for the "hard copy" produced from its electronic source be considered as the authoritative document, the "hard copy" must (a) be a complete and accurate copy of its electronic source and (b) be used exclusively for regulated activities. The current technical architecture of computer

systems increasingly makes the burden of proof for the complete and accurate copy requirement extremely high.

Electronic signatures and controls

- Controls for identification codes/passwords

History

- FDA Title 21 CFR Part 11:Electronic Records; Electronic Signatures; Final Rule (1997)

Various keynote speeches by FDA insiders early in the 21st century (in addition to high-profile audit findings focusing on computer system compliance) resulted in many companies scrambling to mount a defense against rule enforcement that they were procedurally and technologically unprepared for. Many vendors of software and instrumentation released Part 11 "compliant" updates, which proved to be either incomplete or insufficient to fully comply with the rule. Complaints about the wasting of critical resources, non-value added aspects, in addition to confusion within the drug, medical device, biotech/biologic and other industries about the true scope and enforcement aspects of Part 11 resulted in the FDA release of:

- FDA Guidance for Industry Part 11, Electronic Records: Electronic Signatures – Scope and Application (2003)

This document was intended to clarify how Part 11 should be implemented and would be enforced. But, as with all FDA guidances, it was not intended to convey the full force of law—rather, it expressed the FDA's "currently thinking" on Part 11 compliance. Many within the industry, while pleased with the more limited scope defined in the guidance, complained that, in some areas, the 2003 guidance contradicted requirements in the 1997 Final Rule.

- Guidance for Industry Computerized Systems Used in Clinical Investigations

In May 2007, the FDA issued the final version of their guidance on computerized systems in clinical investigations. This guidance supersedes the guidance of the same name dated April 1999; and supplements the guidance for industry on Part 11, Electronic Records; Electronic Signatures — Scope and Application and the Agency's international harmonization efforts when applying these guidances to source data generated at clinical study sites.

FDA had previously announced that a new Part 11 would be released late 2006. The Agency has since pushed that release date back. The FDA has not announced a revised time of release. John Murray, member of the Part 11 Working Group (the team at FDA

developing the new Part 11), has publicly stated that the timetable for release is "flexible."

ISO/IEC 17025

ISO/IEC 17025 is the main standard used by testing and calibration laboratories. Originally known as ISO/IEC Guide 25, ISO/IEC 17025 was initially issued by the International Organization for Standardization in 1999. There are many commonalities with the ISO 9000 standard, but ISO/IEC 17025 adds in the concept of competence to the equation. And it applies directly to those organizations that produce testing and calibration results. Since its initial release, a second release was made in 2005 after it was agreed that it needed to have its quality system words more closely aligned with the 2000 version of ISO 9001.

The standard was first published in 2001 and on 12 May 2005 the alignment work of the ISO committee responsible for it was completed with the issuance of the revised standard. The most significant changes introduced greater emphasis on the responsibilities of senior management, and explicit requirements for continual improvement of the management system itself, and particularly, communication with the customer .

The contents of ISO/IEC 17025 - The ISO/IEC 17025 standard itself comprises five elements that are Scope, Normative References, Terms and Definitions, Management Requirements and Technical Requirements. The two main sections in ISO/IEC 17025 are Management Requirements and Technical Requirements. Management requirements are primarily related to the operation and effectiveness of the quality management system within the laboratory. Technical requirements includes factors which determines the correctness and reliability of the tests and calibrations performed in laboratory.

Laboratories use ISO/IEC 17025 to implement a quality system aimed at improving their ability to consistently produce valid results . It is also the basis for accreditation from an Accreditation Body. Since the standard is about competence, accreditation is simply formal recognition of a demonstration of that competence.

A prerequisite for a laboratory to become accredited is to have a documented quality management system. The usual contents of the quality manual follow the outline of the ISO/IEC 17025 standard.

Predecessors

Some national systems (e.g. UKAS M10 in the UK) were the forerunners of ISO/IEC 17025:1999 but could sometimes be exceedingly prescriptive. ISO/IEC 17025 allows laboratories to carry out procedures in their own ways, but an auditor may require the laboratory to justify using a particular method.

In common with other ISO quality standards, ISO/IEC 17025 requires continual improvement. Regular internal audits are expected to indicate opportunities to make the test or calibration better than it was. Additionally, the laboratory will be expected to keep abreast of scientific and technological advances in relevant areas.

Unlike most ISO standards for systems, third party auditing and appraisal of the laboratory is not usually carried out by a certification body, but by the national organisation responsible for accreditation. Laboratories are therefore "accredited" under ISO/IEC 17025, rather than "certified" or "registered" (c.f. ISO 9000 series).

In short, "accreditation" differs from "certification" by adding the concept of a third party (Accreditation Body (AB)) **attesting to technical competence within a laboratory** in addition to its adherence and operation under a documented quality system, specific to a Scope of Accreditation.

Accreditation Bodies

In order for accreditation bodies to recognise each others' accreditations, the International Laboratory Accreditation Cooperation (ILAC) worked to establish methods of evaluating accreditation bodies against another ISO standard (ISO/IEC Guide 58 - which became ISO/IEC 17011). Around the world, geo-political regions such as the European Community, and Asia-Pacific, the Americas and others, established regional cooperations to manage the work needed for such mutual recognition. These regional bodies (all working within the ILAC umbrella) include European Accreditation Cooperation (EA), the Asia Pacific Laboratory Accreditation Cooperation (APLAC), Southern Africa Accreditation Cooperation (SADCA) and the Inter-American Accreditation Cooperation (IAAC).

In the US there are multiple Accreditation Bodies (ABs). The non-profit, non-government multidisciplinary ABs are the ANSI-ASQ National Accreditation Board/**ACLASS** and **A2LA**. Additionally there is **International Accreditation Service (IAS)**, **Laboratory Accreditation Bureau (L-A-B)**, **Perry Johnson Laboratories (PJLA)**, **NVLAP** (smaller but focused technical areas and some government involvement)and ASCLD (crime labs).

In Canada, there are two Accreditation bodies; The Standards Council of Canada and **The Canadian Association for Laboratory Accreditation**. The accreditation of calibration laboratories is the shared responsibility of the Standards Council of Canada (SCC) Program for the Accreditation of Laboratories-Canada (PALCAN), and the National Research Council of Canada (NRC) Calibration Laboratory Assessment Service (CLAS). The CLAS program provides quality system and technical assessment services and certification of specific measurement capabilities of calibration laboratories in support of the Canadian National Measurement System.

In other countries there is often only one Accreditation Body. Typically these bodies

encompass accreditation programs for Management Systems, Product Certification, Laboratory, Inspection, Personnel and others. The first laboratory accreditation bodies to be established were NATA in Australia (1947) and TELARC in New Zealand (1973) most other bodies are based on the NATA/TELARC model and include UKAS in the UK, FINAS in Finland and DANAK in Denmark to name a few. In India National Accreditation Board For Testing & Calibration Laboratories (NABL) under the Ministry of Department of Science & Technology, Government of India provides accreditation.

Within the United States of America, some industries and government recognize and acknowledge ILAC and its regional cooperations as acceptable; other national regulators and specifiers also acknowledge the National Cooperation for Laboratory Accreditation (NACLA).

ISO 15189

ISO 15189*:2003 Medical laboratories - Particular requirements for quality and competence* specifies the quality management system requirements particular to medical laboratories. The standard was developed by the International Organisation for Standardisations's Technical Committee 212 (ISO/TC 212). ISO/TC 212 assigned ISO 15189 to a working group to prepare the standard based on the details of ISO/IEC 17025:1999 General requirements for the competence of testing and calibration laboratories. This working group included provision of advice to users of the laboratory service, the collection of patient samples, the interpretation of test results, acceptable turnaround times, how testing is to be provided in a medical emergency and the lab's role in the education and training of health care staff.

While the standard is based on ISO/IEC 17025 and ISO 9001, it is a unique document that takes into consideration the specific requirements of the medical environment and the importance of the medical laboratory to patient care.

Good Laboratory Practice

In the clinical and research arena, the phrase ***good laboratory practice*** or **GLP** generally refers to a system of management controls for laboratories and research organizations to ensure the consistency and reliability of results - as outlined in the Organisation for Economic Co-operation and Development (OECD) Principles of GLP and national regulations.

GLP applies to non-clinical studies conducted for the assessment of the safety of chemicals to man, animals and the environment. The internationally accepted definition reads:

Good Laboratory Practice (GLP) embodies a set of principles that provides a framework within which laboratory studies are planned, performed, monitored, recorded, reported and archived. These studies are undertaken to generate data by which the hazards and risks to users, consumers and third parties, including the environment, can be assessed for pharmaceuticals (only preclinical studies), agrochemicals, cosmetics, food additives, feed additives and contaminants, novel foods, biocides, detergents etc.... GLP helps assure regulatory authorities that the data submitted are a true reflection of the results obtained during the study and can therefore be relied upon when making risk/safety assessments.

GLP can become confused with the standards of laboratory safety - wearing appropriate gloves, glasses and clothing to handle materials safely.

GLP and the OECD

Following Decision C(97),186/Final of the OECD Council, data generated in the testing of chemicals in one OECD Member Country, in accordance with OECD Test Guidelines and the Principles of GLP are accepted in all other OECD Member Countries.

Definition

GLP is a quality system concerned with the organisational processing process and conditions under which non-clinical health and environmental safety studies are planned, performed, monitored, recorded, archived and reported.

GLP ensures the quality, integrity, and reliability of safety data.

GLP principles include

1. Organization and Personnel
    - Management-Responsibilities
    - Sponsor-Responsibilities
    - Study Director-Responsibilities
    - Principle Investigator-Responsibilities
    - Study Personnel-Responsibilities
2. Quality assurance program
    - Quality Assurance Personnel
3. Facilities
    - Test System Facilities
    - Facilities for Test and Reference Items
4. Equipments, reagents and Materials
5. Test systems
    - Physical/Chemical
    - Biological
6. Test & Reference items
7. Standard operating procedures

8. Performance of Study
    - Study Plan
    - Conduct of Study
9. Reporting of results
10. Storage of Records and Reports

GLP and the FDA

The United States FDA has rules for GLP in 21CFR58. Preclinical trials on animals in the United States of America use these rules prior to clinical research in humans.

Research in the US not conducted under these restrictions or research done outside US not conducted according to the OECD Guidelines (or FDA rules) might be inadmissible in support of a New Drug Application in the US.

GLP and the European Union

Since 1987 the European Council had adopted two basic Directives and a Decision relating to the application of the GLP principles. Directive 2004/10/EC has replaced Directive 87/017/EEC as of 11 March 2004; Directive 2004/9/EC has replaced Directive 88/320/EEC as of 11 March 2004.

- " Directive 2004/10/EC of the European Parliament and of the Council of 11 February 2004 on the harmonisation of laws, regulations and administrative provisions relating to the application of the principles of good laboratory practice and the verification of their applications for tests on chemical substances."

This directive lays down the obligation of the Member States to designate the authorities responsible for GLP inspections in their territory. It also comprises requirements for reporting and for the internal market (i.e., mutual acceptance of data).

- " Directive 2004/9/EC of the European Parliament and of the Council of 11 February 2004 on the inspection and verification of good laboratory practice (GLP)".

The Directive requires that the OECD Revised Guides for Compliance Monitoring Procedures for GLP and the OECD Guidance for the Conduct of Test Facility Inspections and Study Audits must be followed during laboratory inspections and study audits.

- 89/569/EEC Council Decision of 28 July 1989 on the acceptance by the European Economic Community of an OECD decision / recommendation on compliance with principles of good laboratory practice.

There are also 'Product Oriented Directives' referring to GLP obligations:

- Chemical substances; Regulation (EC) N° 1907/2006 (also known as the "Evaluation, Authorisation and Restriction of Chemicals" Regulation, or "REACH" regulation) of 18 December 2006 and Directive 2006/121/EC of 18 December 2006
- Medicinal products; Directive 2001/83/EC on the Community code relating to medicinal products for human use of 6 November 2001 as amended by Commission Directive 2003/63/EC
- Veterinary Medicinal Products; Directive 2001/82/EC of the European Parliament and of the Council of 6 November 2001 on the Community code relating to veterinary medicinal products
- Cosmetics; Council Directive 93/35/EEC amending for the 6th time directive 76/768/EEC
- Feedingstuffs; Regulation (EC) No 1831/2003 of the European Parliament and of the Council of 22 September 2003 on additives for use in animal nutrition
- Foodstuffs; Directive 89/107/EEC
- Novel Foods and novel food ingredients; Regulation (EC) No 258/97 of the European Parliament and of the Council of 27 January 1997 concerning novel foods and novel food ingredients
- Pesticides; Council Directive 91/414/EEC of 15 July 1991 concerning the placing of plant protection products on the market
- Biocides; Directive 98/8/EC of the European Parliament and of the Council of 16 February 1998 concerning the placing of biocidal products on the market
- Detergents; Directive 98/8/EC Regulation (EC) No 648/2004 of the European Parliament and of the Council of 31 March 2004 on detergents
- EC Ecolabel; Commission Decision 2005/344/EC of 23 March 2005; establishing ecological criteria for the award of the Community eco-label to all-purpose cleaners and cleaners for sanitary facilities

In the meantime the EU has concluded Mutual Acceptance Agreements in the area of GLP with Israel, Japan and Switzerland. By means of the Treaty of the European Economic Area of 13 September 1993, the European Regulations and Directives also apply to Iceland, Liechtenstein and Norway.

GLP and non-OECD member-countries

An inspection in non-member economies by OECD inspectors will not guarantee that data generated in compliance with GLP will be accepted in other member countries than the one to which they are submitting data and which has thus sent inspectors to verify the accuracy of their compliance statement.

GLP and automated systems

Implementing GLP on an automated system, as an intellectual and labour-intensive task, requires a GxP company to make a great amount of effort. To ease the burden of this

management, Webster *et al.* have provided a tutorial for users to quickly embark on and do the job properly.