3.3

THE INFLUENCE OF THE ACCELERATED PROGRESS IN THE COMPUTING WORLD

3.3.1 "WHEN A CRITICAL MASS OF PROCESSES AND METHODS IS FORMED, A NEW PROFESSION IS BORN"

An Interview with Henry Broodney

One of the dilemmas systems engineers have to face is the question of the reciprocal relations between the systems engineering profession and basic engineering disciplines. After all, at its core, systems engineering is a methodology that ties the different engineering fields together. In order for it to do this successfully, its importance and necessity must be recognized by the engineers in those classical fields.

Technion Prof. Aviv Rosen (see Section 3.4.4) is of the opinion that the fact that systems engineers concern themselves mostly with the links between the components of a system, rather than with its professional engineering level, might be the cause of a rift between them and the engineers in the field. It follows that if systems engineering does not make a real connection between itself and the other engineering fields, it will find it difficult to evolve further. Rosen believes engineers expect to see "the link between systems engineering and physics and mathematics." One of the main ways to achieve this objective is the development of computerized systems engineering tools. In recent years, computer companies like IBM have begun developing exactly such tools, in collaboration with systems engineers representing the various industries, who help the company by pointing out their industries' specific needs. This process is a step toward the development of software tools that would provide the industries with the solutions they require.

This chapter discusses the happenings in this field through the story of an electronics engineer who had been employed by the defense and aviation industries, then, later, became a systems engineer, and currently manages the "Systems Engineering Technologies Unit" of the IBM R&D Center in Haifa.

From Electronics Engineer to Systems Engineer

After graduating from his studies at The Technion's School of Electrical Engineering within the framework of the IDF's academic reserve program, Henry Broodney joined the Israeli Air Force's EW (Electronic Warfare) array as a project officer: "As I handled the maintenance and upgrades of EW systems, my first encounter with systems engineering happened early in my career. I had to understand what client requirements were, learn how to translate the pilots' dreams into something practical. I worked with the operating companies and with software specialists from another unit that provided us with its services. Still, at the time, (the early 2000s), I did not think of myself as a systems engineer, only as a specialist officer in a technical field."

Broodney learned the trade on the job. The only formal training he received during the four years of his military service was two courses: an EW course and a Project Officers course.

In 2003, Broodney was discharged from the IDF (after receiving his MBA degree from the Technion). His experience in the military was his ticket to the electronics field in Rafael's EW department, where he held his position for a very short time. At the time, Rafael had won the right to take on a series of projects in the fast-evolving field of weapon stations (remote weapon control performed from a command and control station). Broodney was recruited into a new project in this field as a "Unit Leader in the Electronics Department," which in Rafael-speak meant he was in charge of all the electronics within that project.

There, he had to acquire new skills, both technical and system-integration related: "In the Air Force, I never had to make a device work; I only had to verify that it did. In Rafael, I was required to make sure the devices worked, formulate specifications and standards and then compare them to the requirements."

During his three and a half years in Rafael, Henry Broodney's systemic mindset gradually matured, until he finally became a full-fledged systems engineer: "I gradually began to venture into fields other than electronics, like mechanics. For example, if a box was being built to contain the electronic components, I, as the leader of the electronics units, had to instruct the one who built it. Still, I did not see myself as a systems engineer at first. I worked with the project's systems engineer, whose place was in the project's directorate, and he was the one who coordinated the project's various technological fields.

Later, when I found myself working on other projects (Henry Broodney led the electronics units in various projects during his years in Rafael), I began to realize I was no longer occupying myself exclusively with electronics, but with the entire system. In my last project in Rafael, I was still an electronics engineer, but I was a systems engineer too."

There were other factors, besides Henry Broodney's coordination efforts with other technological disciplines, which brought him to the realization that he was acting as a systems engineer: it was a time when systems engineering was slowly affirming its presence in the awareness of the industry. Rafael, too, had done much to advance the systems engineer profession within its own walls: it held internal systems engineering courses and sent its people to take part in external seminars and professional conferences. Then, there was the gleaming aura of management.

Henry Broodney: "I found the area appealing for me. Systems engineers in Rafael sit in directorates, dynamic business units that work directly with the clients. There were quite a few engineers of my age in the company (only in the fourth decade of his life, Henry Broodney is our youngest interviewee – the authors) who aimed for that position. There was a feeling in the air that the best engineers were going to become systems engineers."

In 2006, Henry Broodney's last year in Rafael, he took part in a systems engineering course, as part of his advancement within the company. At the end of the course, however, his career progression was cut short, as he decided to leave Rafael and, together with a partner, founded a start-up company in the software field. In this new initiative, he found himself with a range of new responsibilities: he managed various aspects of the emerging company's business, including fundraising, finances, recruitment and human resource management, product management, and more. The start-up proved to be a promising venture, and employed seven people at its peak, but the global economic crisis of 2008 struck a fatal blow to the dream of Henry Broodney and his partner. Financing sources ran dry, and the two were forced to end the venture. Henry Broodney, who by then had accumulated some valuable experience in both management and technology, felt his way back into the corporate world and managed to get a job offer from Soltam, a company that specialized in metalwork and ammunition. This offer was the final seal of approval for his status as a systems engineer – the head of the company's Department of Artillery offered him the position of a systems engineer in the company administration.

Henry Broodney: "Soltam was undergoing major changes at the time, as part of its recovery from a recent crisis. The position I was entering was a new one. The Head of the Administration, who had also transferred from Rafael several months earlier, offered me the job. We tried to instill systems engineering work patterns in Soltam. I also learned a new field along the way: heavy mechanical production."

During the period of his employment by Soltam, the ties between Soltam and Elbit were becoming increasingly close. The two companies collaborated on several projects, and Henry Broodney found himself working intensively with Elbit personnel. During the second year of his employment by Soltam (2010), the company was acquired by Elbit, and its people took over all the high ranking positions. Finding no career prospects in the newly reconfigured company, Henry Broodney chose to leave.

Not long after, he received a surprising job offer from the IBM R&D Lab in Haifa, to work, once again, as a systems engineer. Why did the computing giant's research laboratory need a systems engineer, whose background was in electronics and whose experience was in defense systems?

Indeed, ordinarily, IBM hardly ever needed people of Henry Broodney's professional background. However, at the time, a special opportunity arose at the company, and Henry Broodney fit into it like a glove.

Henry Broodney: "The mandate of IBM's development units is to develop innovative technologies for use in IBM's products. Around that time, IBM's subsidiary, Rational, whose product was software development tools, acquired a company called Telelogic, which specialized, among other things, in developing (software) modeling tools for use in software engineering. The Haifa Research Lab saw an opportunity, and the executive who recruited me began to form a team to take up this field. He made contact with the US Defense Advanced Research Projects Agency, the organization in charge of developing the American defense system, and they started working on a certain project. At this point, he needed a content expert who was familiar with the inner workings of the defense industry and knew how to use computerized modeling to work on a project."

Soon, other software development projects were referred to the Haifa Lab, and roughly one year later, IBM's management formally approved the establishment of a group that would develop systems engineering technologies. At the head of the group, which currently numbers 12 members, is systems engineer Henry Broodney.

The group began by developing software tools for systems engineers in the aviation industry. Later, demands began to arrive from other areas as well, such as the automotive and transportation systems industries.

Systems Engineering at IBM

IBM employs thousands of people in "systems engineer" positions worldwide. The company makes efforts to advance the systems engineering profession and organize a body of knowledge of the discipline and holds professional gatherings and seminars. Yet, Henry Broodney, who had spent much of his career in the defense industry – the cradle of systems engineering, holds that most, if not all of them, are not really systems engineers, but IT engineers who use systems engineering methodologies in their work. In other words, they are systems engineers, whose work is confined to the IT field. Additionally, most of IBM's systems engineers are not placed in the company's R&D units, but in its sales, marketing, and service units. The purpose of this strategy is to allow these units to speak the same language as the systems engineers of IBM's potential clients, those who will eventually be offered to buy IBM's computerized systems, which will help them in their work. Henry Broodney says: "Perhaps, with clients like Lockheed Martin, one does need a full-time systems engineer."

It follows that IBM will not need many systems engineers of Henry Broodney's type in the foreseeable future. It might need one systems engineer for each new field its development center decides to develop products for, because "IBM is not going to develop products, but product development tools."

The research activity of IBM's R&D labs is not pure, empirical research, but rather, applied research – the kind that aims toward the development of products, for which the industry will present a demand. However, IBM does not sit idly and wait for clients to place orders for the development of products that suit their needs (although sometimes, as foresaid, development takes place in collaboration with the client). Rather, it tries to understand where the industry is headed beforehand and develop software products to suit its future needs. This is the mission of the unit under Henry Broodney's management, centered on the specific case of developing tools to help systems engineers in various industries. This is also why he is perfect for the job. As a systems engineer who evolved in industries that are considered to be the spearhead of the systems engineer profession, Henry Broodney is an external content expert, able to help the IT giant become familiar with the needs of its potential clients in the systems engineering world.

Henry Broodney: "We mingle with the various industries, hear their problems, and then identify the ones IBM is able to offer solutions for. The next step is to find a partner, inside or outside IBM, and to start developing the solution.

Physically, the systems are similar; the difference lies in their domains. Some systems are mechanical intensive, while others are hydraulic intensive, like the equipment of an oil company. For the research teams developing the systems engineering software, the specific engineering discipline is of little importance. We do not need to know the engineering minutiae; we only need the top level. Then, we use the services of experts who usually come from our research partners, like Boeing or Daimler AG. IBM recruited me as a systems engineer who knows which products are relevant to the industry he came from."

Henry Broodney explains further: "There are two types of systems engineering, the first of which began to emerge as far back as the 60s and 70s. It includes processes and work methods for the management of projects that include three key elements: scope, schedule, and money. The second type of systems engineering deals with engineering planning and the process of designing the system itself. The first type focuses on system management, meaning what the project manager does, whereas the second type focuses on the engineering, meaning what the project's chief systems engineer does. This is the focus of my unit – technical, not management-oriented systems engineering.

The planning stage is crucial: 70 per cent of the product's costs, reliability, and performance are determined during the project's preliminary stages, when preliminary design is done right. This perception is slowly seeping into the awareness of the industry. There is an increasing demand for (computerized) tools that help design good system architecture; tools that help the developers understand the interaction between the software and the physical components it controls."

Insights on Systems Engineering

On the evolution of systems engineering

- In Roman times, it was enough for one smart man to understand how things worked. Systems were simple, back then. One talented man could see the whole picture, weigh the right considerations, and take many different aspects into account. Things worked more or less the same way until World War II, when big names like Wernher von Braun and Willy Messerschmitt were still prominent, and one man's talent could give birth to major innovations.

Starting from the mid-twentieth century, the leaps and bounds of technology have created a situation where being smart is not enough, and in order to innovate, one has to structure the effort. One also needs processes to help him think, because systems have become far too complicated for the capabilities of any one person. Teamwork has become a necessity. All of today's greatest inventions are the results of group efforts. Hence, we need processes that support group work. So it happens that, when the processes and methods reach a critical mass, a new profession is born, and, like anything new, it needs a name, a buzz word to help position it.

On the identity of systems engineering

– Systems engineering has evolved differently in each organization and has been called by many different names. Moreover, in each organization, systems engineers perceive their position differently (if that organization even has systems engineers). In Elbit, for example, there are no systems engineers; instead, there are "technical managers," a name that offers a rather accurate description of the job. There is an administration called the Systems Engineering Administration, but the people who work there are called "technical managers." In Rafael, on the other hand, there are systems engineers who are essentially technical managers.
– Most systems engineers in Rafael have backgrounds in either mechanics or electronics. If a project is not assigned a systems engineer, the electronics engineer automatically received the title, being likely perceived as the one most suitable for the job and best able to see the whole picture. In my estimate, the reason for this is the fact that systems have begun to be increasingly software-heavy, and software engineers have a good grasp of several areas, because all electronics engineering curriculums include software studies. These two factors created a situation where electronics engineers are assigned the positions of project systems engineers.
– In the automotive industry, there is no "systems engineer" position. There are, however, "architects," who essentially fill the same role and will, in time, become INCOSE members. Currently, most INCOSE members are employed in the defense and aviation industries, but more and more new members are flocking to it from other industries, as they come to realize that they are all members of the same trade. The industries may work in different ways, but the body of knowledge is the same: there is a wheel that turns and a software controller that oversees its movement.

The qualities of a good systems engineer

– Today, one person can no longer manage a large project. For example, one cannot demand that a project's chief systems engineer understand the workings of the project's financial management. He needs to recognize that there are financial considerations, or know how the design is reflected in the cost, but the costs themselves must be planned out by someone else.

– A systems engineer needs to know "enough," not "more." He needs to be able to handle many things at once, to multitask. I find it difficult to focus on something narrow and systematic. I am more holistic than analytical by nature; I am also highly intuitive. But that did not make me a systems engineer; the chain of events that introduced me to systems did. To this day, I miss delving into the inner workings of a technical system.

3.3.2 "LOOKING AT A PROBLEM FROM DIFFERENT ANGLES"

An Interview with Mimi Timnat

Systems engineering has evolved differently in every organization, in accordance with its unique needs and organizational culture. In a considerable number of cases, a system engineer's job title is not even necessarily "system engineer." The job descriptions tend to vary as well – the range of tasks a system engineer is given can change greatly from one organization to the next.

In this chapter, we will expand on the use of systems engineering in Elbit Systems, through an interview with Mimi Timnat, a high-ranking systems engineer who has filled a wide range of positions throughout her career, from her beginning as a software engineer to her current position. Among other subjects, our conversation also revolved around the evolution of software engineers into systems engineers.

From Software Engineer to Systems Engineer

Much like many other systems engineers, Mimi Timnat, currently the Head of System Engineering and Technical Management Process Improvement at Elbit Systems, had set foot on the path toward systems engineering unintentionally, as part of her natural development, guided by her personality and interests.

The transition to systems engineering drove Mimi Timnat to discover fields of engineering that have nothing to do with software, and lead the development of multidisciplinary systems, in which software plays only a minor role.

Having completed her studies in the department of Computer Science at the Technion (as part of the IDF's Academic Reserve program), she joined Elbit Systems as a software engineer and was tasked with software development. Two years later, she was promoted to Software Project Manager. In this position, she started to become acquainted with systemic issues, in addition to software-related ones.

Mimi Timnat: "Even as a Software Project Manager, I had discovered, as had many other software managers, that my duties often required me to go beyond mere software implemented solutions. We often discovered, in the midst of the software development process, that some things were difficult to implement in accordance with the predefined requirements or that the definitions themselves were not clear enough. I usually understood what was needed. On some occasions, I would not wait for the systems engineer to become available and provide me with a definition; rather, I would suggest solutions myself and then coordinate them with the systems engineer.

Back then (in the 80s), Elbit Systems had already had systems engineers who were referred to as such, but their job descriptions were rather amorphous. At the time, the practice of systems engineering was almost an art form: there were people who knew what to do, but no methodological foundation and no organized training (the field of software had very similar beginnings, by the way)."

Mimi Timnat's transition to systems engineering took place sometime later, in the 90s, and was completely unplanned and unintentional.

Mimi Timnat: "My transition to systems engineering began spontaneously, following my participation in a design review that discussed the solution to a complex problem. My part in the review was supposed to be insignificant – I was only to test whether the part that was meant to be implemented using software could indeed be accomplished. The solution presented in the review was based on the unification of partial solutions, which, together, formed a solution for the larger problem. When I saw the solution, it seemed to me that in certain situations, the all-inclusive solution would prove impossible. It was as if they required one car to be in two places at the same time. I asked whether I had understood the solution of the suspect situations correctly, and was met with silence. As it turned out, the solution really was faulty."

Shortly after, she was offered to temporarily assist Systems Engineering with resolving the problem and then return to Software. She brought up the proposal with her superior.

Mimi Timnat: "I approached my superior and told him about the offer. He said: 'If you move to Systems Engineering, I do not believe you will want to come back to Software. If you want to go back, you can. But, I think you will find the field attractive, and you will choose to stay.' It later turned out he was right."

Mimi Timnat's transition to systems engineering was not limited to solving the problem that initiated it; it opened up whole new worlds for her.

Mimi Timnat: "I joined meetings with clients, and was required to understand their expectations and look for solutions. I was exposed to engineering fields I had never dealt with, and terms I had never learned. At first, it was strange and unclear, but along the way, I asked questions, learned and began to understand many engineering subjects – far beyond mere software. I was naturally attracted to the need to understand the whole picture, and project-oriented systems engineering has given me that option."

How does a software engineer, whose knowledge base and experience are rooted in the software field, acquire the extra knowledge a systems engineer needs?

Mimi Timnat: "In any engineering discipline, the knowledge acquired in studying does not last very long. Therefore, one must always remain up to date. One of the things that can be said to the Technion's credit is that, among other things, they developed their students' independent thought and learning abilities. During the course of my work, I taught myself much about previously unfamiliar areas and used the help of others as well. In my estimation, people who find knowledge and understanding important find ways to acquire the knowledge they are missing. Even when I go to the doctor, I ask questions and try to gain a better understanding of things."

Today's sophisticated systems combine a variety of engineering areas and specializations. It is difficult to expect one person to be an expert in everything. One of the challenges of developing a complex system is handling the interactions between all the different disciplines and their specialists. Thus, Mimi Timnat holds that one of the systems engineer's most important tasks is to understand engineers from various disciplines and coordinate between them: "My growth as a systems engineer was an unplanned process, where one thing led to another. I clearly remember a discussion with many participants from various disciplines, about ways of implementing a solution for one of the issues in the project. At the beginning of the meeting, I raised several questions on subjects I was not clear on and wrote down the agreements we had reached. Later, other engineers raised other issues, seemingly irrelevant to me. There were many professional terms I did not know, I could barely understand what was being said, and I found it hard to keep summarizing the discussion. After the meeting was over, I asked one of the other engineers for help clarifying the unfamiliar subjects and checking whether I had summed them up correctly.

That meeting was just the beginning. As time passed, the subject was raised again and developed further. More details were added and improvements to the agreements we had arrived at in the preliminary discussion were suggested.

At the second discussion on the same subject, I was asked to update my earlier summary. The discussions were not easy. People from different engineering disciplines were having trouble understanding the problems and solutions of other groups. Sometimes, people used the same terms to refer to different things. Without planning to, I had become a 'translator' for the different groups (note: a similar problem is described in the interview with John Thomas).

During the course of the project, that same "protocol" had evolved into a specification, dozens of pages long, wherein were listed the details of a solution that involved several engineering disciplines. And so, to my surprise, as I was updating the "protocol," using my ability to understand the different groups and coordinate between them, I had become a knowledgeable leader on the subject, within that project."

Software Engineers as Systems Engineers

Previous chapters in this book have already addressed the subject of software engineering and its reciprocal relationship with systems engineering. Particularly, they raise the argument that software specialists tend to concern themselves with software and are less willing to deal with the other engineering disciplines. On her part, Mimi Timnat claims that such generalizations cannot be made and that these attitudes depend on the people: "Today, there are many disciplines and subdisciplines, even within the umbrella of software engineering. For instance, the expertise required for designing video games is not the same as the one needed for developing organizational information systems or communication applications. Some prefer to specialize in software, others, who have a tendency toward working with systems, enjoy working in large projects, in systems of systems, and being able to understand the whole picture. These people choose to evolve into systems engineers, because it allows them to grow laterally. This way, they familiarize themselves with many different fields, like control, communications, mechanics, optics and more – all as the project requires. Everyone finds what is right for him.

I believe it is important for a systems engineer to have a good grasp of the dominant area in the project he is developing. In software-heavy systems, software engineers have a natural advantage, but there are systems where the dominant area is not software. An example of that is robotics. In these systems, mechanical engineers and electronics engineers have an advantage, as systems engineers who also studied software, but not as thoroughly (today, this is true for all engineering disciplines). In computer science on the other hand, there is no requirement to study other disciplines, which is why, in some projects, software engineers find leading the engineering of the system difficult."

Another phenomenon presented herein, in this context, is the increasing number of software engineers found among systems engineers. Mimi Timnat agrees with the existence of this trend and explains it by saying that software is slowly taking up a central position in complex systems: "In the past, hardware components had much more dominance in systems than software components, and so a considerable part of the systems engineers rose from disciplines of relatively high technological complexity, such as electronics. Today, software takes up much more weight, causing more systems engineers to evolve from that area. The career change from software to systems engineering entails a shift of focus. The broad perspective becomes more important; the focus shifts to understanding the comprehensive solution that pertains to all disciplines, and how the system is compatible with the user; and away from the details of software implementations."

This is the place to note that Technion Prof. Aviv Rosen argues that the case of Mimi Timnat, who has achieved high-ranking positions in systems engineering, is a rarity. He stands among those who hold to the previously presented opinion, according to which the lion's share of software engineers still prefer to focus on software, and only a minority are willing to tackle other fields. He believes the reason for the growing number of systems engineers who rise from among the ranks of this group of engineers is that software engineers have greatly increased in numbers in recent years (in many projects, they constitute the largest group of engineers). He claims that the relative share of systems engineers who started out as software engineers is still fairly small.

On the Identities of the Systems Engineer and Chief Systems Engineer

Systems engineers are usually engineers by trade, but Mimi Timnat believes it is possible for systems engineers to be trained in other areas as well. "It depends on who the developers are. If most of them are engineers, and they expect a systems engineer to be well versed in the project's dominant discipline, then, naturally, he has to be an engineer. But it is possible for a developer of, for instance, a biomedical system, to be a doctor with a broad perspective and some comprehension of engineering, even if he is not an engineer per se.

Sometimes, an engineer can come from the operational side. Then, he would have an excellent grasp of client needs. I know some excellent systems engineers at Elbit Systems who rose from operational disciplines and have no engineering degree. For example, it is said that some of IAI's best systems engineers are former IAF pilots. In any event, it is important for a systems engineer to have several years of hands-on experience working on development projects."

The subject of professional background relates to the question of a chief systems engineer's suitability to the project he is meant to lead. Mimi Timnat believes the job of a project's chief systems engineer is influenced by the association between the project's technological core and the systems engineer's original discipline: "Having an understanding of a range of different fields is a fundamental condition for a systems engineer. He must be able to recognize what he knows, what he does not know, and when to consult other experts. But a chief systems engineer needs to have a very good grasp of the project's lead discipline. For example, I, as a systems engineer who started out in the software field, would find it difficult to be the chief software engineer in a project laden with mechanics or robotics. Like other disciplines, systems engineering has specializations."

This leads us to the question: if two of the most important qualities a systems engineer must possess are learning ability and a broad perspective, why would a potential chief systems engineer not learn the new area and lead the project? Why should he continue to focus on his generic area of expertise?

Mimi Timnat: "It can be done, but learning takes time. Projects usually have very tight schedules, so prior knowledge is preferable. If a chief systems engineer has gained relevant experience working on a similar project, it is best to use it. Engineers are indeed quick learners, but if there is the possibility of finding someone with relevant experience, we try to do so."

Systems Engineering at Elbit Systems

The evolution of systems engineering varies from one organization to the next. It depends on each company's needs and organizational culture. At Elbit Systems, a company that develops and produces advanced electronic and electro-optic defense systems, systems engineers focus mostly on the technical areas, but need to account for a variety of considerations, including budget, scheduling, risks, and more. Elbit's organizational culture makes all engineers consider these constraints, but the core of their work is, as aforesaid, engineering and technology oriented.

In addition to "systems engineers," Elbit also has a position called a "technical manager." Technical managers bear a general responsibility for the execution of a project's development, which they coordinate with the project manager (who, in Elbit, is known as the "Program Manager"). The program manager is more concerned with the business and contractual aspects of the project. He outranks the technical manager, but the latter is in charge of the project's technical areas. In most cases, the technical managers' occupational history includes the position of a systems engineer at Elbit. A program manager can grow out of the company's engineering areas, but may also come from a business or operational background and with no engineering experience.

Having filled the positions of Software Engineer, Software Project Engineer, Technical Manager, and Program Manager, Mimi Timnat advanced to a new, unique position within the company – Head of Systems Engineering Methodology and Development Process Characterization. This position's uniqueness stems from the fact that only a select few of the hundreds of systems engineers employed at Elbit Systems work outside the framework of a project. They are organized in a centralized body that provides services to projects and business divisions that deal with system development.

Further Insights on Systems Engineering

On the Essence of Systems Engineering

– Systems engineering is a discipline that deals with problems and solutions that combine many different fields of engineering. It includes a variety of tasks, all of them involved with the process of developing a project. These include analyzing requirements, formalizing the concept of the solution, integrating, testing, and more. Systems engineers deal with different engineering fields at different scopes, and therefore, there are different levels to their positions. Systems engineering focuses on application, rather than research. This is why academia finds it difficult to "raise" systems engineers, unfamiliar with development projects. It is difficult to explain how to handle the issue of "understanding the needs of the client" until one meets a real client who has trouble defining his need and has to do it for him.

What Makes a Good Systems Engineer?

– (Mimi Timnat relies on the results of a study on the subject performed by Prof. Moti Frank):

- Broad perspective – "This ability is vital, and not necessarily inborn; it can be learned. One is taught to ask questions, to examine things from different angles. One is given examples of typical questions, and considerations, useful when comparing alternate solutions to different problems. This is how this ability is acquired."
- Good self-learning ability.
- "The ability to distinguish between what one knows and what one does not."
- Resourcefulness and flexibility – "A systems engineer constantly encounters unexpected problems and mismatches between different engineering disciplines. During the course of the project's development, he has to deal with the reactions of developers from different fields, such as 'what you defined cannot be done' or 'these constraints cannot be met.' It is important

to constantly check whether the main requirements are met and come up with creative solutions for any surprises encountered along the way."

- Good interpersonal skills.
- Leadership – "It is important to not only find the correct solution, but also convince the involved parties to accept it."
- The ability to "move" things.
- The ability to communicate with the client and come to understand his expectations – "This means seeing things from different perspectives, being able to use other people's terms, understand their real needs, even if the client himself is having trouble defining them."
- The ability to delve into the details, without losing the broader perspective. Because "at times, only when you get to the details of the implementation, do the problems begin to show themselves."
- Common sense.
- The ability to formulate solutions in conditions of uncertainty – "Because the systemic solution concept needs to be designed in the project's early stages, when many details are still unclear. The systems engineer needs to be able to understand what can and cannot be implemented, without knowing all the details (and in larger systems, a single engineer can never know all the details)."
- A systems engineer needs to know when he understands enough, when to ask for more details, and how to get them. For instance, when a client asks for a change to be made in mid-project: at first, the systems engineer makes a preliminary assessment of the consequences of the change, in order to see whether the change is even reasonably possible. Only if he finds the change plausible, does he consult the leaders of the various disciplines for a deeper examination of the consequences of the change and the budgeting of its implementation.

Mimi Timnat summarizes: "I find that, to a great extent, systems engineering is more than just a job. It is an approach to handling and solving problems, and not only work-related ones. It encourages one to look at a problem from different angles, to ask questions and try to gain a better understanding of the problem, before making decisions and formulating solutions."

3.3.3 "VENTURING BEYOND THE CORE-SUBJECTS TO STUDY NEW AREAS"

An Interview with Harold (Bud) Lawson

One of the major factors for the constant increase in the complexity of technological systems and integrated systems is the dramatic increase in computing and communication, which has allowed for the development of impressive technological capabilities. One of the pioneers of the global computing industry, Bud Lawson, believes that Systems' Engineering evolved as a discipline intended to assist us in dealing with such complexity.

This chapter presents Bud Lawson's developmental path; a man who initially was not trained in engineering, had developed into a leading computer expert and later on into a systems' engineer. His transformation into a systems' engineer had come about with a clear awareness that is not always characteristic of computer experts, as one cannot function as a computer specialist without understanding the overall system that a computer-based solution is supposed to serve.

It's Not the Computer, It's the Application

Bud (Harold) Lawson, a computer engineer-cum-systems engineer, who is considered one of the pioneers of the global computer industry, never studied computers in an orderly fashion, since no organized studies in the field existed, naturally, in the late 1950s, the period in which he entered into this emerging industry. Born in the United States, Lawson's formal educational foundation was in mathematics and statistics, which he studied along with economics.

In 1959, Lawson began working in the new computing industry that had intrigued and excited him. He learned from experience. He worked at Remington Rand Univac and then at IBM between 1961 and 1967. Being an expert in programming, he was involved, during his first years in the company, in a series of projects dealing with the development of computer languages and compilers. At a later stage, he decided to change track and get involved in computer design, focusing upon the field of hardware.

During those years, the industry was dominated by the big computers, first and foremost of which was the IBM System/360, whose development and penetration into global markets the Company had focused its attention upon. This approach was detrimental to Lawson as an entrepreneur and developer. In 1967, he headed a small research team that examined one of the fundamental problems of the computer world – the relationship between hardware and software. But at the same time, according to him, "IBM was not open to ideas that could prepare it for present, as well as future, operations. The Company traveled along the path of the '360.' As such, even though the research had discovered ways to improve the hardware–software interface, it did not receive the attention it deserved."

Lawson left IBM, joined the academic world, and had also become a senior consultant in "Computer Engineering" – which was practically a new area of activity at the time and which did not exist in academia.

Lawson became a professor and a senior consultant in "Computer Engineering," even though he had not explicitly been trained nor worked as an engineer, in any classical sense. According to him, "the field was called 'Computer Engineering' because it dealt with the design of hardware and software systems and with defining the relations between them." He began to study and join projects in the United States, but mostly in Europe. He relocated to Sweden first as a consultant to Datasaab and joined the Linköping University faculty, where the field of Computer Engineering was becoming highly active.

We shall present two prominent projects in which he took part during those years (the 1970s), as a senior computing consultant. The systemic perspective, which lays at the foundation of systems engineering, is prominent in both. This was so, even before he saw himself as a systems engineer. Eventually, this approach, by his own definition, turned him into "a systems engineer, specializing in computer-based systems."

His first project was related to the development of a remote monitoring system for high-voltage power transmission.

Bud Lawson: "One of my students was from Barcelona. He had worked at the Enher power company that developed a remote monitoring system (tel-control system) for power distribution. To that end, they needed to construct a computer-based system. He asked me to assist them. We discovered several interesting solutions that were successfully implemented. This is an example, not only of the work done on the computer itself, but also of its applications. For me, that marked the beginning of a shift, because as I look back and examine my work, it seems that I have always worked at systems engineering. But that conceptual shift began, for me, in the mid 1970s, when I stopped focusing upon the computers themselves and started looking deeply into applications for computers."

But were you aware then, in the mid 1970s, that you were formulating an approach that was different from that of most contemporary computer specialists?

Bud Lawson: "I felt that I was doing something different, but I did not use the word 'systems engineer'. I had worked at computer engineering previously, as well. But after I had started working on the applications, I told myself that I needed to learn much more about other areas, such as power distribution. If I wanted to design a suitable solution, then I could not focus just upon understanding the computerization part, I needed to understand the area that the computer was supposed to serve."

The second project involved the development of an automatic control system for trains, the first of its kind in the world.

Bud Lawson: "In the mid 1970s, operations, which could not have been done in the past, were made possible by the microprocessors that started to appear on the global market. These microprocessors changed the economic world of hardware. Standard Radio & Telefon contracted to Swedish Rail asked me to take the lead in a development project, the first of its kind at the time (no such system was in operation anywhere in the world) – computerized monitoring of the train network, which until then had been electro-mechanic. For instance, the system would allow us to monitor the drivers' behavior. If, for example, the train driver suffered a heart attack (and such things do occur), the monitoring system will automatically stop the train; or, you could define speed limits between the signal segments. Such a system also optimizes the engine's operational performance.

The company that was put in charge of the project was as mentioned 'Standard Radio', a local subsidiary of the ITT communication corporation. When I had first arrived on the scene, they already had an active team that had constructed a simulator that included what the system needed, to their understanding. I looked at what they built and said: 'there are going to be a lot of problems, because there's a lot of 'spaghetti code' (long and complex lines of code – The authors) that are difficult to understand'. I suggested that we go back and examine the system's requirements.

A key requirement was the ability to maintain stability every 250 milliseconds, or one quarter of a second. You can do much with a computer in that sort of timeframe. I told them: ‘Let’s look at this differently, as if it’s a clock cycle. It might be hardware, but let’s look at it as if the software’ was hardware. I suggested a different approach, to look at the system as a continuous system rather than as a discrete system; since the most critical variable here is time. In this way we transformed it into and far simpler solution. We built a system that had only 4,000 instructions and only 10 KBs of code. We installed it in around 1,000 train engines and it worked very well.”

(Note: Looking at the system as a continuous system prevents us from focusing upon *specific* (discrete) instances, usually rare instances, which also require a solution from the system. As an analogy, we can imagine the system as focusing upon the flow of a wide river without being required to focus upon its dozens or hundreds of tributaries. The river flow is the essential point, and we focus upon it. Obviously, such a concept reduces the system’s complexity and cumbersomeness – The authors).

This approach has served Lawson in additional instances as well, while allowing his skills as a systems engineer to manifest themselves. He is one who is wise enough to use a solution that has been successful in one system to seek out solutions in a different system.

For example: He was approached by Haldex contracted to Volkswagen in the 1990s, to assist them in the development of a device that could increase the fuel efficiency and safety in their 4 × 4 vehicles.

Lawson: “They encountered problems caused by their historical mechanical view of product development. To them dealing with electronics and software was a new experience.

These were basic problems of systems engineering *such as the need and usefulness of well-defined processes* that were the result, among other things, of mechanical engineers doing all the work, even though they did not understand electronics and computers. Once Haldex realized they had a problem, they approached me as a real-time computer systems expert. I developed for them a monitoring system that was similar to the system originally developed for Swedish Rail. These days, this component is installed in most 4 × 4 vehicles in the world.

In order to address the development of this automotive component composed of electronic hardware, software, and mechanical parts, I tailored a version of the ISO/IEC 12207 (Software Life Cycle Process) standard in order to include not only software, but also the electronic and mechanical hardware. This was prior to the development of the Systems Engineering standard ISO/IEC 15288 of which I became the architect.”

Insights into Systems Engineering

On the essence of systems

– A system is a collection of items that creates something that possesses new functions. As a result, we sometimes witness the emergence of new patterns of behavior.

- There are hard systems, and there are soft systems. As time goes by, even my fellow engineers understand that they need to see the human side, along with the engineering\physical side. The word engineering has been used to describe soft systems: social engineering or human factor engineering. We must be aware of these aspects when we design a system. When we design an aircraft, we need to take into consideration the person intending to operate it. Even if it's a completely automated vehicle, there is still a human component.

On the complexity of systems

- Many systems suffer from over-complication, from an overabundance of unnecessary functions. This makes monitoring and controlling them difficult. The problem is especially prevalent in computer-based systems, because programs are virtual thing and can be developed endlessly. In the past, computer memory was relatively small and as such made the development complex and complicated programs impossible.
- A systems engineer is supposed to prevent, as much as possible, unnecessary complexity in the system and to deal with the problems caused by complex systems as well. On the one hand, a relatively simple system needs to be created, one that can be worked with. On the other hand, there are systemic problems that require modification and the development of features that assist in dealing with their complexity (meaning that we need to add elements to the complex systems that will help in dealing with their complexity issues – The authors).

On the essence of systems engineering

- One of the problems in systems engineering is that their design is based upon engineering systems, and not upon the broad perspective within the context of a system, which includes the people operating and using it. Systemic thinking is an area that has yet to be well defined, despite the fact that we use such systemic thinking, in practice, in our daily activities. The Academic world is currently researching the science of systems, systemic thinking, and their connection to systems engineering. But the purpose of science is to create understanding, not presenting solutions.
- Systems engineering began with people focusing upon narrow aspects of the system*s*. When we develop systems with different aspects such as mechanics, electronics, computers, and so on, it involves multiple disciplines. Achieving results – the creation of products and services – requires the creation of cooperation between them. Therein lays the crux of the problem – someone needs to bridge between them.
- In Sweden, systems engineering is not studied under this name – they use other names such as "industrial management." I myself teach in educational programs, which fit in better with programs that include systemic embedded infrastructure. These programs are not called "systems engineering" due to the division between those areas that involve engineering and those areas that do

not really involve engineering. So, even though we recognize the importance of this issue and might even pay it lip service, integration between disciplines is not so simple in reality.

On the characteristics of the job

- There are those that will say that systems engineering is both a profession and the discipline. For me, it has become a profession. I am a systems engineer that specializes in computer-based systems.
- A systems engineer does not have to be an engineer. I am not an engineer by education, but I am an engineer in practice. It is not a profession but rather a way of thinking. We are constantly finding solutions to problems.
- A systems engineer needs to be open to the study of new topics and not focus solely upon his area of expertise. Thus, I have learned much about engineering in practice.

Authors' Insights:

On the complexity of systems

- Complex systems are developed, not because the clients require all of their functions, but rather because the ability to develop them exists – Lawson concurs.

On the essence of systems engineering

- Systems engineering is a developing area that has yet to define itself – Lawson concurs.
- The professional community accepts Lawson as a systems engineer, even though he lacks an education as an engineer, because his professional background is computer sciences, which is seen, in many cases, as belong to the world of engineering.
- Despite the fact that in a variety of fields, there is a tendency to adopt the thought patterns of systems engineering, there are forces, which are very strong, which cling to the traditional engineering professions, which do not take this area of expertise for granted. To them, systems engineering looks like a discipline that goes beyond its true area of expertise.

3.3.4 "THE ABSTRACT LEVEL OF DISCUSSION IS OF GREAT VALUE"

An Interview with Sharon Shoshany Tavory

There is a unique, reciprocal relationship between the two worlds of software and systems engineering. Many claim that this relationship emerged from traditional, physical engineering. The need to plan and integrate complex, technological systems has raised the need for a design and integration toolset that has since become the infrastructure of systems engineering. Software, however, is not always perceived as part of the engineering world, mostly because of its virtual nature. Technion Prof. Aviv Rosen tells us of a conversation he had had with Amos Chorev, former president of the Technion, who had wondered: "Where is the engineering here?" In another chapter in this book, Kobi Reiner says that as a systems engineer, he had, on several occasions, encountered problems when working with software specialists, because they had been unwilling to see the limitations of the system as a whole, persistently focusing almost exclusively on their software-related tasks.

Nonetheless, software is often the factor that facilitates system integration. This means that the ability to design software bears a lot of weight in the development of systems engineering. Today, the discipline's two overarching missions, namely, preliminary system design, followed by integration of the system components, entail massive use of software-based tools. Moreover, some would claim that software engineering resembles systems engineering more than any other engineering discipline, because they both create virtual engineering environments, unlike traditional engineering disciplines, which, as we said, are anchored to the physical world.

This chapter will present the career progression of a chief systems engineer who started out as an electronics engineer, transitioned to software engineering early on, and then moved on to evolve into a chief systems engineer.

Her views and the progression of her career tell the story of the evolution of systems engineering in recent years, with an emphasis on its unique reciprocity with the world of software.

From Electronics and Software Engineer to Systems Engineer

In 1986, two years after she had begun working for Rafael, electronics engineer Sharon Shoshany Tavory was recruited to work on a large project. She was tasked with developing real-time software (a program that had to meet requirements for performance under certain, predetermined time-related constraints) for certain parts of the project.

What would an electronics engineer have to do with systems engineering?

Sharon Shoshany Tavory: "At the Department of Digital Systems, where I worked, the prevalent perception was that real-time software development should be done by electronics engineers, because software people could not see the whole picture. They only saw their own 'bits'."

In those days, Sharon Shoshany Tavory had already begun practicing systems engineering, but she only came to realize it about a decade later, when she discovered systems engineering as an emerging methodology: "At those days, the term 'systems engineering' was already in existence, but it did not bear the meaning it does today. It was usually used to refer to someone who got promoted to Assistant Project Manager. Back then, I had begun doing systems engineering work. For example, I managed interfaces and analyzed processes, but I was not called a 'systems engineer'."

Sharon Shoshany Tavory's entrance into the world of systems engineering was also influenced by her personality, as well as by her work as a software developer:

"I was not introduced to systems engineering during my academic studies at the Technion. At the time, it was not considered to be an academic discipline. I acquired my expertise in the field on the job, learning from experience. Every engineer starts his professional career as a 'screw'. He can choose to remain a screw and focus on the field of engineering he specialized in, or he can choose to look around him. I chose to look around me.

At that time, software engineering had only just begun creating tools for systemic work and systems thinking. Because software is abstract by nature, it had an especially great need for tools such as analysis of requirements or systemic abstract modeling. One can safely say that systemic models were born in the world of software."

In 1999, Sharon Shoshany Tavory, then head of Rafael's Department of Digital Systems, decided to go on sabbatical in Australia. It was there that she encountered the terminology of systems engineering in full force: "This was a much 'hyped up' subject at the time, and people began seriously considering formalizing and institutionalizing systems engineering as a discipline. One of the consequences of this was the change in the name of the institute, where I was a guest, that year. When I had arrived, it was called the 'Test and Evaluation Center,' but during my stay, it was renamed 'Systems Engineering and Evaluation Center'."

In the year 2000, Sharon Shoshany Tavory returned to Israel and was appointed Project Manager. In this position, she was faced with the varied challenges of project management for the first time, including such responsibilities as working with outside clients, like the Israeli Air Force and US aircraft manufacturer Lockheed Martin. In her previously held positions, she had served clients from within the organization, including project managers, who received professional services from her and the department she led.

Was this not too sharp a transition? Would it not have been better to undergo a gradual training course that included, for instance, an interim job as Deputy Project Manager and/or a project's Chief Systems Engineer?

Sharon Shoshany Tavory: "It is true that I had never attempted actual project management before then, but I had gained experience in similar frameworks. First, as a department manager, I, of course, had management responsibilities: I devised work schemes, negotiated with clients within Rafael, and managed employees. Second, I was no stranger to contact with external clients, either. As a department head, I met with clients who wanted to gain a deeper understanding of the specific field I was in charge of (as part of her work on other projects – the authors). Third, the project was not a huge one, and I am good at adapting to new situations, asking the right questions, and making decisions accordingly.

In large projects, responsibilities are usually shared between a project manager and a chief systems engineer. The project manager focuses on managing the client, planning and, control; and the systems engineer mostly deals with the technical issues. In a small project, however, the project manager can also deal with the technical system's engineering aspects, which was what I did."

This was the first time Sharon Shoshany Tavory got to manage a group of people who specialized in different professional areas (the project employed 12 people from

4 different fields): "When it comes to his area of expertise, the specialist knows better than me, but I can ask him which alternatives he had considered; ask him to explain why he had chosen a certain alternative.

This was also the first time I had a budget to manage, and I had to manage the profits in a fixed price undertaking (as a department manager, I had also dealt with finances, but from a planning perspective; the work done was measured by the hour). But I did not work alone; in this area, I used the help of my deputy for planning and control."

In 2003, Sharon Shoshany Tavory was appointed Chief Systems Engineer of the Armaments Systems Division.

Are there major differences between the systems engineering–related functions of a project manager and those of a division's chief systems engineer?

Sharon Shoshany Tavory: "A project is a field activity, where the manager has to choose which systems engineering practices to use in his work (for instance, formulating a systemic architecture, considering alternatives, devising the trial plan and so on). A division's chief systems engineer is different, because it is a headquarters position. Its responsibilities are more oriented towards the discipline than towards its specific applications. Headquarters examines the entire spectrum – all the tools and methods needed by all the projects. This is why a division's chief systems engineer needs to possess a more 'lateral' education and more experience than a project manager. However, it is not always the natural next step in a project manager's career. One can become a chief systems engineer of the engineering division without first serving as a project manager, and vice versa. They are two different positions, with two different areas of responsibility."

The relationship between engineering department and the management of a project is characterized by similar conflicts to those that often arise between the departments in the organization and headquarters units. One such conflict concerns the question of making professional decisions in the engineering unit's area of expertise. On the one hand, the engineering unit's client is the management of the project, which means the unit should serve the project management and meet its needs. On the other hand, the engineering unit is the highest professional authority when it comes to its own area of specialization.

Sharon Shoshany Tavory illustrates this dilemma: "My job as chief systems engineer is not only to provide service, but also to oversee the level of systems engineering implemented in the project. In one particular project, the engineering field specialists recommended a certain mechanical structure material, and the project management refused to follow their recommendation. As a rule, project management has the final word, because it is the client. Traditionally, in Rafael, when the engineering department staff insists on something, claiming to have a well-founded technical recommendation, the project management does its best to overrule it. This is exactly what happened in our case. We appealed, and a team of high-ranking division position holders deliberated the issue and decided to accept the engineering department staff's position."

In 2005, Sharon Shoshany Tavory was appointed the position of Head of Directorate in one of Rafael's product lines. The Head of Directorate supervises the project managers, systems engineers, and design personnel and controls the projects in his jurisdiction.

Sharon Shoshany Tavory: “This position is considered part of the career path of project managers, and not systems engineers, who have no career paths to speak of.”

(Note: In our view, systems engineers do have their own career paths, albeit not formal ones, seeing as systems engineering is, as yet, an emerging profession. Systems engineers can be promoted to chief systems engineers or to project management positions that require in-depth knowledge of systems and technology. It should also be noted that although there is a separation between the career progressions of project managers and systems engineers, it is not absolute. In technological organizations that deal with areas like defense, aviation, or space, many of the project managers are former chief systems engineers. This is also the case for Sharon Shoshany Tavory herself. In business organizations, however, project managers are raised from among marketing and business management specialist – The authors).

Sharon Shoshany Tavory: “As a rule, the Head of Directorate only deals with the overarching control aspects of systems engineering, in the projects under his responsibility. Here, too, systems thinking is a useful instrument; but, seeing as systems engineering has always been close to my heart (I suppose the combination of technical and philosophical abstract thinking was what drew me to it), I continued practicing it more deeply than is traditional: in administration projects, at the Israeli Association on Systems Engineering (INCOSE_IL) and in the capability maturity model integration (CMMI) process (a program for assessing an organization’s ‘maturity’ as it pertains to systems engineering and project management) in Rafael.

During my time as Head of Directorate, I learned about systems engineering in product lines, where it includes the various considerations entailed in finding the optimal technical solutions for products that share similar properties.

Explanation: managing the systems engineering of a family of products necessitates the creation of commonalities. This entails the planning of subsystems, able to serve different products that belong to the same family; for example, using the same control system in different vehicles. This is an expression of one of the changes the systems engineering field has undergone. From a discipline that supports one project, it has become a discipline that accounts for the fact that one project begets others. This is a systems engineering that designs subsystems that fit into different products.”

We shall present an event from that time that demonstrates Sharon Shoshany Tavory’s systemic thought patterns and conduct as a systems engineer who sees a technical problem, gathers the relevant knowledge, and suggests solutions while facing constraints: “One day, during the 2006 Lebanon War, I came to visit one of my colleagues in the missile division, who had been stationed in the control room of the early warning system that warned civilians of imminent attacks by rockets. I was excited, both by the work itself and by the fact that he had such a direct contribution to the war effort. I, too, wanted to contribute, and it turned out they were in need of a quick solution for a connectivity requirement from the Air Force (between the system that detects the launch point and the system that guides the aircraft towards it). He suggested that I sit in on a discussion on the subject the following morning, and I did. The Air Force representatives wanted us to link the rocket detection system to the aircraft, so that they can damage the launchers in real-time, rather than just alert the civilians of imminent impact. Not only that, but they wanted this task accomplished within four days – a very short time.

I volunteered for the task without hesitation, having learned not to 'blink' in cases like these. I received the approval of my superiors, and we got to work. My part consisted of designing the system and coordinating the factors. I was assisted by two software and integration specialists, who were in charge of adjusting the existing system and building the interface between it and the aircraft. Later, another project leader joined the effort. On the fourth day, the system was installed in the Air Force's control center and functioned as required. This task required me to rely heavily on my 'peer leadership' skills, because I had no formal authority over the people I worked with."

In 2008, Sharon Shoshany Tavory had left her position as Rafael's Head of Directorate and became a consultant and a lecturer.

The Relations Between Software Engineering and Systems Engineering

In our retelling of Sharon Shoshany Tavory's career, we mentioned that in the beginning, she and her fellow electronics engineers were asked to develop real-time software, because, at the time, The Department of Digital Systems, to which she belonged, believed they would do this better than software specialists (see preceding text).

Sharon Shoshany Tavory gives an example that explains how this perception came to be: "One of the members of the team I led was an academic software engineer who wanted to work in the field and was given a task: to design the software for a box that contained an electronic system. Every time I visited him I saw a different design. I did not understand this. The previous design seemed fine, so why start over and change everything? I decided to get to the root of the matter and found out that all those times, the integration with the hardware had failed, and he went back to square one and redesigned (and then rewrote) the software. It seemed to me, however, that the problem had nothing to do with the software. When the hardware specialist looked into the matter, he discovered that the system's grounding had been faulty, and so, roughly once a day, when the programmer would run the software he had designed, the system would receive a minor electric shock and the content would be distorted; the programmer interpreted this glitch as a software problem.

Grounding is one of the most common reasons for electronic equipment failures, but only an electronics expert would recognize that he is dealing with faulty grounding. Software specialists, who are strangers to electronics, assume there is nothing wrong with it, conclude that the problem must lie with their software, and decide to rearrange their designs."

But do electronics engineers have sufficient skill in the software field to lead the development of real-time software without seeking help from software experts?

Sharon Shoshany Tavory: "Electronics engineering and software engineering are closely interconnected. Electronics is the base infrastructure of all computers and of the software that runs on them. In the Technion, where I studied, the prevalent approach was that computer engineering stood in the middle between electronics engineering and computer science. 4 out of the 13 programs of study in the electronics engineering faculty were in computing fields.

Of course, one can study electronics and ignore the world of software, but today, most students choose not to skip computer science studies, and do become acquainted with the world of software and its systemic aspects. This is the prevalent approach in all engineering disciplines, today."

Indeed, software engineering has been becoming more and more important in the engineering world. After all, software is a core element of the connectivity between technological systems. As technological ability increases along with the capability of designing complex systems, so does the importance of software (and consequently, of software engineers).

Sharon Shoshany Tavory: "In the past, systems were developed without the help of software specialists; that approach had failed. Today, unlike then, one person cannot grasp and contain the range of engineering disciplines entailed in a complex project, and so varied teams are needed."

She illustrates the increase in the importance of software by giving an example from Rafael's area of activity: "The missiles developed in the past mostly consisted of physical parts, like the warhead and engine, and a little bit of 'brain'. Now, a missile is only one part of a much larger command and control system; the system that controls the missile. Missile development used to be a classic aeronautical project. Today, it is part of a project an aeronautics expert can easily get lost in, even if he does have some knowledge of software.

These transformations have had an impact on the risk management model, as well. In the past, we dealt with such risks as engines that did not allow the required aerodynamic performance; today's risks are systemic. The most problematic area has shifted towards software, because of the increase in the importance of connectivity (which software facilitates), which, in itself, is a source of complications. All these make the system harder to understand for those who specialized in other engineering disciplines.

In the reporting sessions of the past, you could often hear statements like '... let the software people make their report last, it's not really interesting, and we want to get this done with and leave'. This is not an option today, because software is embedded into every field."

This suggests that the bilateral connection between software engineering and systems engineering has really existed all along. Since systems engineering is a methodological tool, meant to support the planning of engineering systems and link them together, a systems engineer has to understand and show interest in the engineering disciplines that make up the system he is in charge of. Software engineers, on the other hand, have often been thought of as specialists, concerned only with software, unfamiliar with the traditional, physical world of engineering.

Systems engineering and software engineering bear a resemblance to each other, seeing as they are both, in essence, abstract systems. Both disciplines need virtual models that allow them to accomplish at least one of their shared overarching missions: to create integration between technological systems.

Sharon Shoshany Tavory: "In systems engineering, there is a lot of abstract discussion that needs to take place before moving on to the 'physical' stages. It is different from classical engineering fields. In many cases, engineers who focus on the physical aspect and attach little importance to the abstract aspect are criticized. Software engineering, however, is abstract in its very essence. Software specialists do not produce a physical product. The blueprint of the product – the code – is the product. And so, in time, the work methods of system engineers have become more and more similar to those of software specialists.

On the other hand, seeing as software engineers do not come into contact with any other technology except software, they are often unable to comprehend it. This is why, back at the department of digital systems, we wanted electronics engineers to be the ones to write real-time software."

Has the similarity between these two engineering discipline brought about mutual adoption of work methods?

Sharon Shoshany Tavory: "Indeed, it has. For example, the method called 'System Modeling Language' or SysML. SysML is a computerized visual modeling tool that allows for easier understanding and testing of ideas, before there is a system to speak of. Originally, the tool was developed to be used by software developers, who refer to it as 'Unified Modeling Language' or UML in short. When we first discovered UML, we saw that it could be used for systems engineering. We used it not only for modeling software, but also for modeling systems. Sometime later, INCOSE used the UML as a foundation for an expanded systems engineering tool, and SysML was born."

Insights on the Systems Engineer Position

- A systems engineer needs to communicate with other disciplines. One can only lead the development of a system when he leaves the confines of single-disciplinary engineering. I did not just perform integration with other engineers; I also worked in their areas, on subjects outside my area of expertise. I did this, not just because it was required in practice, but because I was willing to venture into those areas.
- On the correct mix between the three factors that affect the development of a systems engineer: personality, experience, and training:
  Many systems engineers have a knack for good systems engineering and practice it intuitively. For these people, personality and experience are crucial factors. One of the most talented systems engineers I have met had also been formally trained (in the Technion's ME program in systems engineering), despite the fact that he possessed considerable experience. He said to me: "The course gave me tools to express myself with; I needed that supplementation."
  The order of things is also important. Systems engineering training should only be had after gaining hands-on experience, otherwise it offers no benefit. A systems engineer first needs to practice integration processes, see systems fail, and understand why it happens. If he does not encounter these situations, he will not be able to absorb the lessons of the training program.

To conclude, a systems engineer's personality, his ability to see things from a broad perspective, is very important. Having that, he needs to gain experience in order to become more aware of potential problems. Finally, formal training is the factor that completes a systems engineer's expertise. Personally, I found the fact that I majored in technological subjects in high school helpful, because there, I was introduced to various fields and gained some hands-on experience as well.

- In some cases, people want to become systems engineers because they think of it as a promotion. But, compared to project management, which certainly is a promotion, the status of a systems engineer depends on the discipline's positioning within the organization. Companies do not always know how to position the job, and systems engineers often have no options for promotion beyond their current position. (See aforementioned author's note on this subject).
- There is a list of generally acceptable qualities a good systems engineer should possess. Not all of them are needed to succeed on the job, but effective communication is definitely among those that are.