Systems Modeling & Requirements Specification Using ECSAM: A Method for Embedded Computer-Based Systems Analysis

Abstract

ECSAM is a requirements engineering and modeling method for computer-based systems (CBS). It is practiced and enhanced since 1980 by large numbers of systems and software engineers. ECSAM was developed in part at Israel Aircraft Industries for the analysis and design of complex reactive embedded systems and software.

ECSAM guides the developers in systematic analysis and modeling of systems being developed and describing those using three consistent views; its conceptual architecture, operating modes, and system's capabilities. Each capability is further analyzed and described as a dynamic process. Using ECSAM it is possible to generate systematically the system's use cases and the resulting operational scenarios. The modeling technique is applicable in the analysis of systems and any level of subsystems. The method guides the developer in the derivation of the system requirements and their systematic allocation to and association with the model's elements at the respective levels.

ECSAM was the prime motive for the development of the Statecharts technology developed by Prof. David Harel as part of the method.

The paper provides an overview of the ECSAM method, which is described in detail in a book that will be published in 2004 [1].

1. Introduction

Computer-based systems have become ubiquitous in recent years. Raising your eyes, you are bound to notice numerous, common devices that are computer-embedded or computer-based: cellular phones, digital watches, cars, CD and mini-disc players, and many more. The pervasiveness of computer-based devices and the ability to interconnect them into integrated systems brought about the development of larger, more complicated systems. Some of these are relatively small and simple, such as digital cameras; some are more complicated, such as the networked computer-based devices controlling the operation of our cars; while others are large and complex, such as telephone exchanges or integrated air-traffic control systems.

This paper introduces ECSAM, an Embedded Computer Systems Analysis and Modeling method [1]. The ECSAM model-based method allows systematic analysis of the interaction of a system with its operational environment, and provides a procedure for the systematic identification of characteristics needed by the system to fulfill its missions. Among these characteristics are the outputs the system must produce, the inputs needed to produce these outputs, the dynamic processes that transform the inputs to outputs, and the operating modes that determine when specific processes should be performed. The ECSAM method supports a systematic object-based decomposition of the proposed system into conceptual subsystems or objects, allowing the analysis and the specification of the subsystems and how they should interact to satisfy the system's objectives.

Although the method was originally developed to help in the analysis of embedded systems—which typically consist of hardware and software—over the years it has shown its effectiveness in the analysis of a wide range of system types. It was successfully applied in the analysis of systems composed of software only, hardware-software systems, as well as in the analysis of operation of organizations and teams.

The ECSAM method addresses both the static and dynamic characteristics of such systems, as well as their hardware and software aspects. By providing a consistent model of the system, ECSAM supports the systematic treatment of stakeholders' requirements, as well as the methodical derivation of requirements that specify the mandatory characteristics of the system and of its subsystems. Such characteristics are properties that the resulting system must possess to be able to fulfill its mission.

The analysis process focuses initially on the E-level model, an extended scope of the system. That scope includes the system—presented as a "black box" whose inner workings are not of concern at this stage of analysis—and the environmental systems with which it interacts. The E-level model defines the system

boundary, the external ("environmental") systems with which the system interacts, the interfaces between the system and the external systems, and the known interfaces between the external systems. The dynamic view of the E-level model provides the basis for the systematic derivation of use cases and the resulting operational scenarios that define the required dynamic behavior of the system, as well as for the derivation of operational and test requirements.

The static and dynamic properties of the system discovered through analysis of the E-level model serve as a starting point for the second phase of analysis in which the inner workings of the system to be built are modeled and analyzed using a "white box" modeling approach. The transition between black-box and white-box models is carried out by limiting the scope of analysis to the system and its interfaces, creating a system model that is called the "S-level model." The S-level model is a hierarchical conceptual model of the system, which allows separate analysis of conceptual subsystems to a level of refinement deemed appropriate by the analyst. The analysis of the S-level model addresses its static and dynamic properties in a manner similar to the one employed in the analysis of the E-level model. During successive iterations of modeling and analysis, the system is broken down into lower-level conceptual (logical) subsystems.

The analysis process verifies the correctness of the decomposition and the consistency of the properties of the system and its subsystems (at any level). Correctness is verified when the analyst can demonstrate that the static and dynamic properties of the system (or any of its subsystems) can be expressed in terms of the properties of its conceptual components.

Requirements originally allocated to the system are iteratively allocated to progressively lower-level subsystems. This flow-down process drives the refinement of high-level requirements and determines the mandatory characteristics of the individual subsystems.

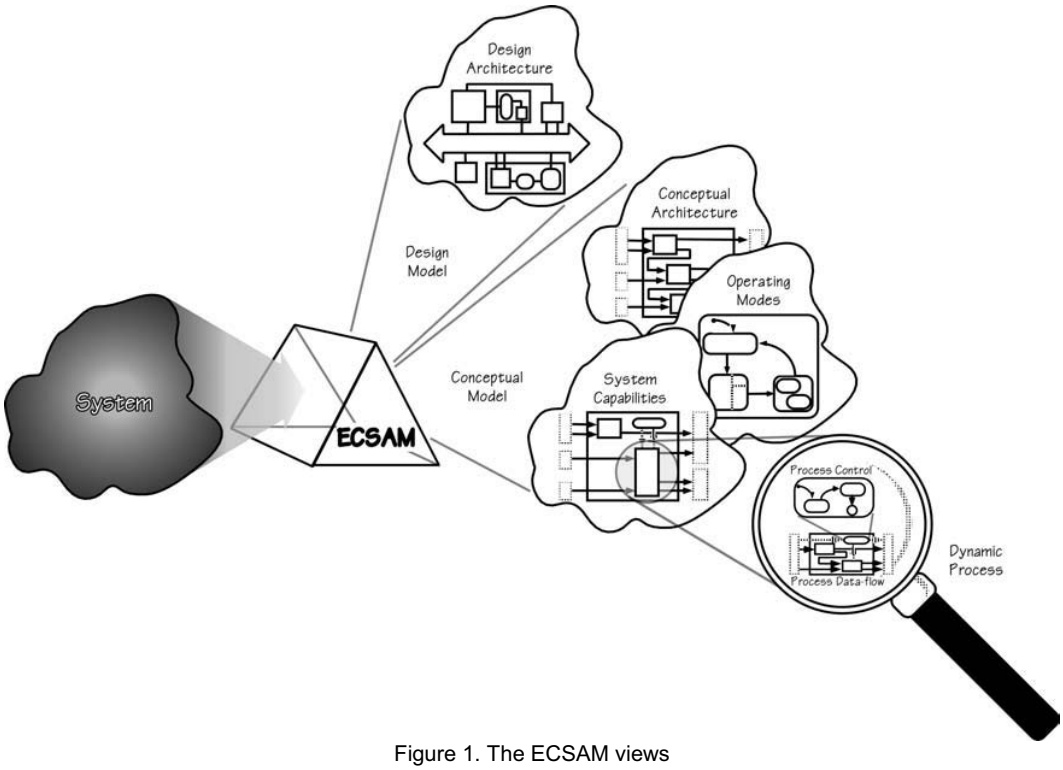

Figure 1. The ECSAM views

The ECSAM method uses several complementary and interrelated views,[1] all of which utilize graphical representations. ECSAM also uses formal (graphical and mathematical) semantics, wherever applicable, in the specification of the E-level and S-level models producing executable specifications that allow the analysts to test the system's conceptual, static specification and simulate its dynamic behavior.

2. The ECSAM modeling approach

ECSAM facilitates the creation of clear, consistent, and testable conceptual black-box and white-box models of reactive CBSs; it also enables the systematic derivation of the computer-based systems' requirements and their allocation and association with the model elements. Using the ECSAM method, analysts can investigate the hardware, software, and human-related aspects of single- and multi-computer systems concurrently during all development phases. The method also dictates concurrent and iterative analysis of static and dynamic aspects, to provide for internal consistency of the model. Following the recommendations of Swartout [2], ECSAM suggests concurrent analysis of requirements and top-level design, as appropriate, during development of the conceptual model.

2.1 Overview of the ECSAM model

The following sections describe the general characteristics of ECSAM models. Due to the inherent complexity of embedded computer systems, it is impractical to try to describe all of their essential properties using a single view or model. A practical solution for the modeler is to use several interrelated views, modeling the system's static and dynamic aspects, as shown in Fig. 1. Focusing on various model views, one at a time, reduces the level of complexity the analyst faces at each analysis step.

As shown in Fig. 1, we can view ECSAM as a kind of prism that splits the system description into a conceptual model and a design model. The conceptual model is further divided into three interrelated views.

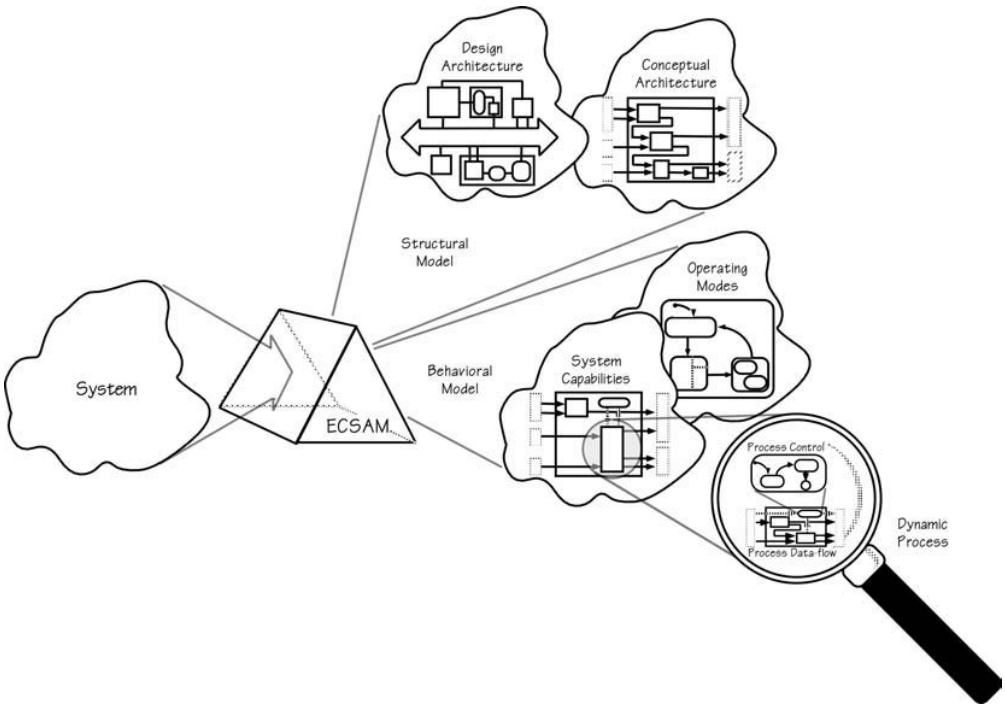

Figure 2. Structural vs. behavioral ECSAM views

Conceptual models specify families of systems (or products) that can be mapped onto different design architectures having similar functionality (although not necessarily similar performance or other non-functional characteristics), depending on implementation constraints. Existing model templates and generic specifications of known systems and subsystems can be reused, considerably shortening the development process.

2.1.1 The ECSAM conceptual model. The ECSAM conceptual model is described by three views:

- *The conceptual architecture* view describes the system's conceptual structure, namely, its internal conceptual subsystems, their functionality, and the external and internal interfaces of the system.
- *The functional capabilities* view describes the activities representing the functional capabilities of the system. Each of these capabilities is

described as a dynamic process expressed in terms of the transformations it performs, its data-flow, and the associated process control.

- *The operating modes* view describes the main operating modes of the system and the transitions between them.

Jointly, the functional capabilities and operating modes views describe the dynamic behavior of the system.

These three views are complementary and interrelated, as discussed later in the paper. During the system analysis phase, the various views are iteratively examined to assure consistency and completeness of the model and its resulting specifications.

2.1.2 The ECSAM design model. The ECSAM design model describes the design architecture of the system, the structure of its hardware and software subsystems, and their relationship with one another. It also specifies the dynamic interaction of the architectural elements in various system modes, addressing both the data flow and control flow aspects of the interaction.

The design model also expresses performance and implementation constraints, and describes the human-machine interface.

2.1.3 Structural versus behavioral (static versus dynamic) system models. As stated previously, ECSAM models address the system's static and dynamic aspects. Another way to group the model views is by means of their structural (static) and behavioral (dynamic) characteristics, as shown in Fig. 2.

The static model represents the system's structure, while the dynamic model represents its behavior. The static model is described by the system's conceptual and design architectures. The dynamic model is described by the dynamic views of the conceptual and design models. Since this paper does not address the dynamic aspects of design models, this issue was omitted from the figures used in this paper.

Experience shows that the architecture of systems changes little, if at all, over the years. There are several reasons for that, such as the difficulty of changing the "building blocks" from which a system is constructed or the way they are interconnected, and the high cost of retrofitting systems already in operation. To provide for that, the modeler must focus on creation of conceptual and design architectures that will maintain their stability throughout the system's life cycle.

Such stability can't be guaranteed with respect to the system's desired behavior, which tends to change during the system's life cycle as its environment changes. Change may be brought about either as the result of some type of influence of the system itself on its environment, or because of improvement in the user's understanding of the system's operation, or because of changes in the user's needs and expectations [3]. Hence, the dynamic model is the one that usually changes most dramatically as a system evolves. Provisions for the adaptation of the system's behavior over its life cycle must be incorporated into the models during the analysis process.

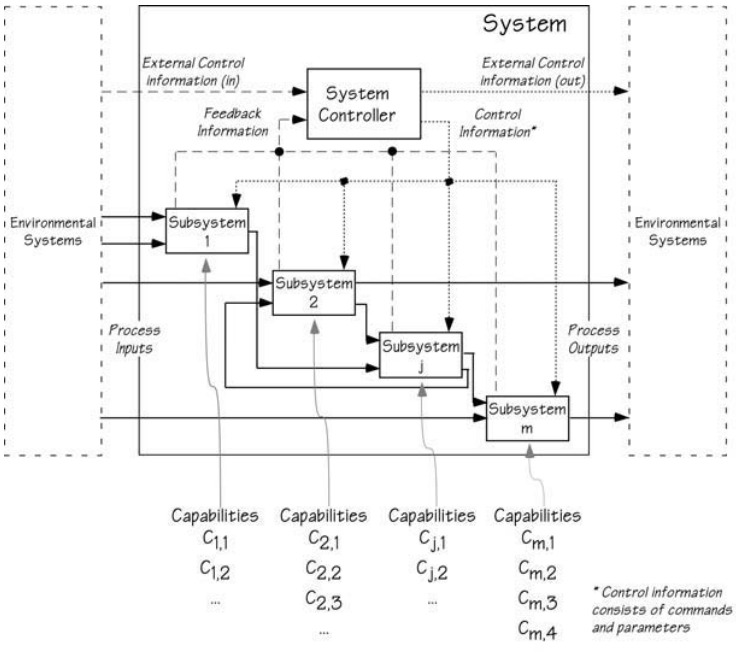

Figure 3. The generic model of a system

3. The relationship between conceptual views.

ECSAM assumes that every system and every subsystem can be modeled as a hierarchical control system [4, 5]. All systems, including computer-based ones, can be modeled using a generic model like the one presented in Fig. 3. Since this generic model is crucial in explaining the relationships between the ECSAM views, we find a brief discussion of its generic characteristics to be justified here.

The model describes the system's conceptual architecture, and is based on the assumption that every system or subsystem is composed of

- a number of conceptual subsystems
- a conceptual controller

The model's components act jointly to carry out the system's mission. Each of the subsystems can be viewed as an abstract machine or object, that is, as a kind of information-hiding module (as defined by PARNAS [6]). These subsystems provide services (that is, capabilities, such as $C_{1,1}$, $C_{1,2}$) that participate in the realization of the system's capabilities. In fact, each of the system's capabilities can be presented as a combination of functional contributions provided by the subsystems.

The conceptual controller governs the joint behavior of the subsystems that is required to accomplish the system's mission. It controls the system's operating modes and the dynamics of the processes performed by the system. The subsystems can be mapped to hardware, software, and mixed hardware-software systems, and—in systems implementing a man-in-the-loop approach—also to humans.

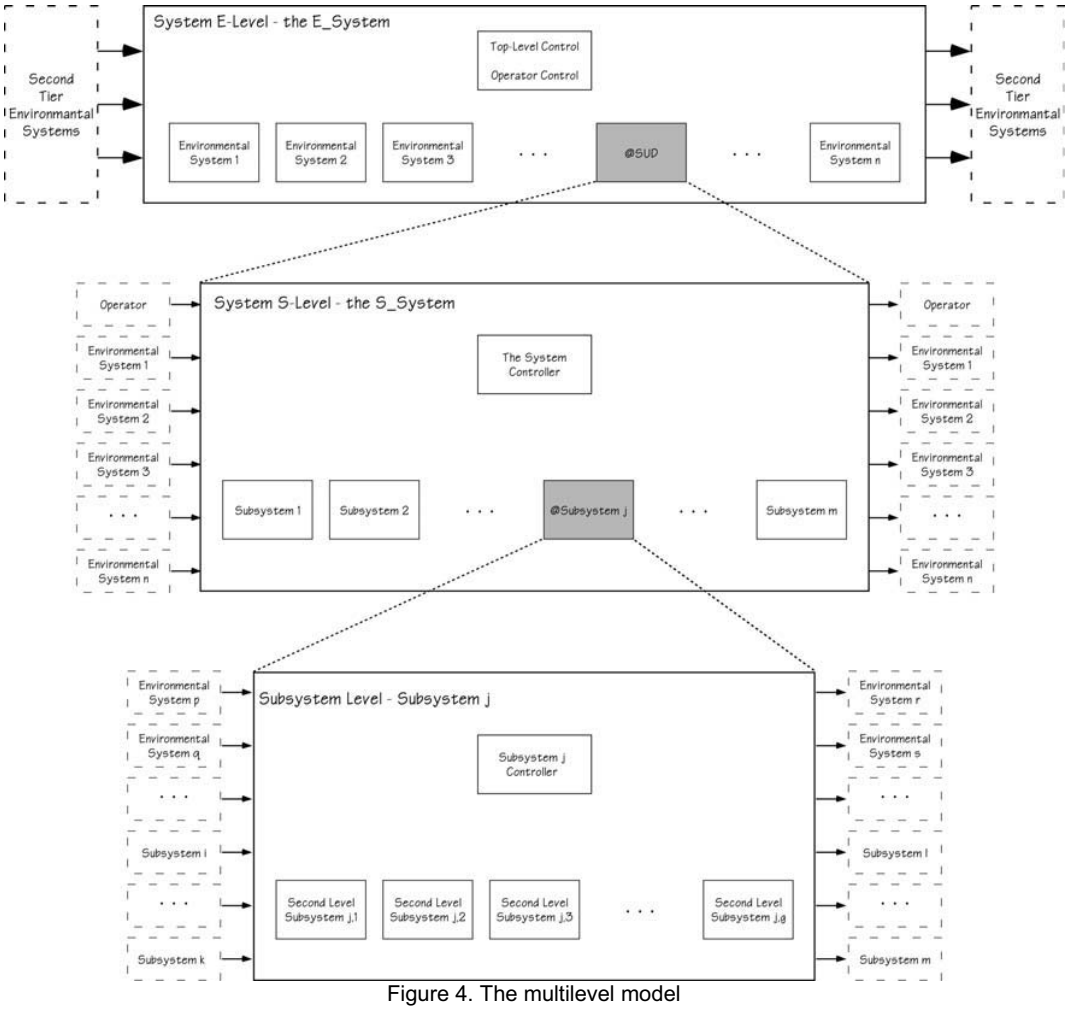

Figure 4. The multilevel model

The controller can be

- implemented in software and/or hardware or assigned to human operators
- real or conceptual
- centralized or distributed among the subsystems
- implemented on one or multiple levels

system's controller, governs the operation of the subsystems and their behavior. Furthermore, external control data is exchanged between the system's controller and the higher-level systems that manage the system's overall behavior. Feedback information provides the controller with details about the actual state of any given

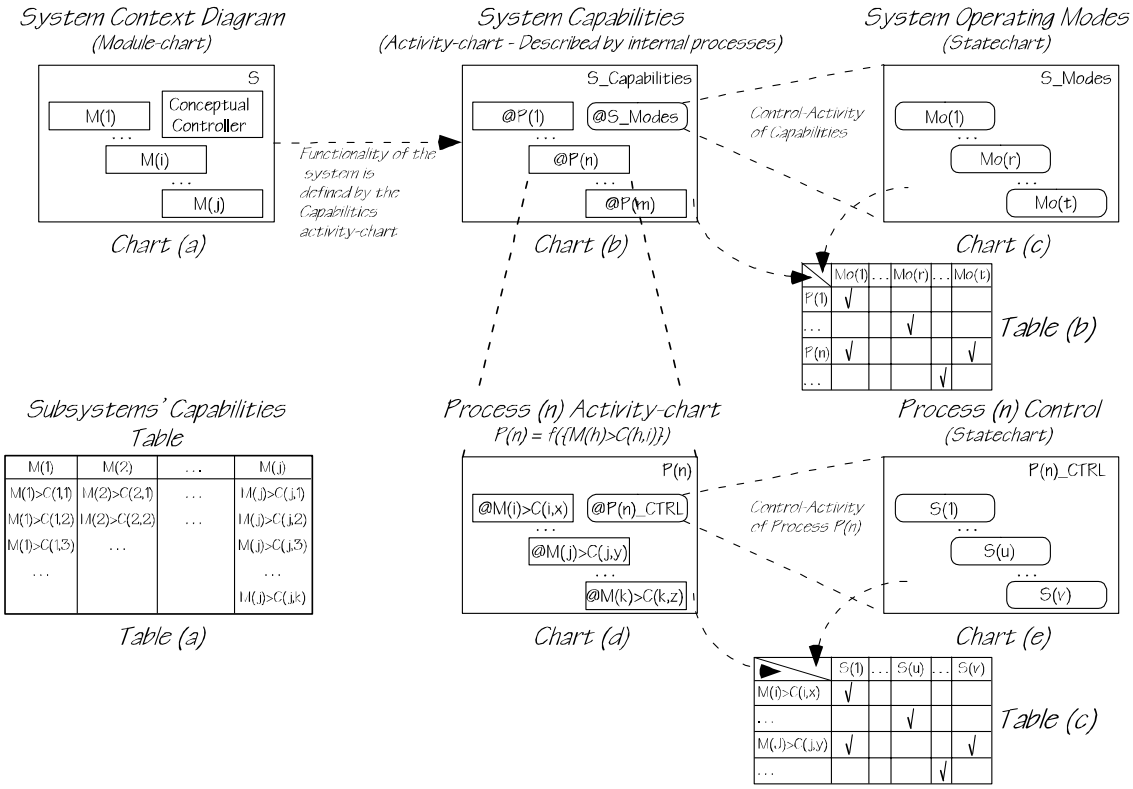

Figure 5. The relationships between components of the conceptual model

When users or operators are involved (which is quite common at the E-level), it is not unusual to find that at least some of the control functions are performed by humans.

The model presented in Fig. 3 identifies three types of information-flows[2]:

- process information-flows
- control information-flows
- feedback information-flows

Process information represents data, signals, and physical entities that are transformed by the system and its subsystems. Control information, produced by the subsystem at any time.

One of the goals of the analysis is to identify all of these information-flows, as well as their sources, destinations, types, and scope.

Each conceptual subsystem can be further decomposed into lower-level conceptual subsystems and a conceptual controller of its own, using white-box modeling methods shown in Fig 4.

Systems analysts in charge of developing a conceptual model of a system usually limit the depth of decomposition to a single level, the S-level. The S-level model typically identifies conceptual subsystems, describes their behavior, their interfaces, and the capabilities performed by them. Analysis and decomposition of subsystems at lower levels are performed by analysts in charge of these subsystems. However, when internal conceptual subsystems are very large or complex, the analyst in charge of the S-level

model may find it necessary to analyze and detail some (or all) of the subsystems. Although different analysts may be involved in the analysis of S-level system and its subsystems, the analyst responsible for the S-level must review the results of the analysis performed at both levels to ensure that the subsystems jointly fulfill the system's mission. This review may reveal, for example, the need to reallocate capabilities among the subsystems in order to improve their cohesiveness and reduce their mutual coupling.

The generic model of a system presented in Fig. 3 provides a starting point for the explanation of the relationships between the conceptual views used in ECSAM models.

Figure 5 presents the relationships between all types of elements constituting each level of the model.

Chart (a) in Fig. 5 presents the module-chart of the system or any of the conceptual subsystems, its conceptual next-level subsystems, its conceptual controller, and the associated external and internal information-flows.[3] The capabilities provided by each of the subsystems are listed in Table (a). The activity-chart describing the capabilities appears in Chart (b). The operating modes are shown in the Statechart presented in Chart (c); their correlation with the capabilities is presented in Table (b). Each of the capabilities, *P(n)*, is presented as a process described by the activity-chart shown in Chart (d) and by the Statechart shown in Chart (e). Table (c) identifies the correlation between the process states and the capabilities that are available in each of them. The conceptual controller, shown in chart (a), controls the operating modes of the system as well as the dynamics of the processes.

It is important to remember that the capabilities listed in Table (c) are members of the set listed in Table (a). This is shown explicitly in Chart (d), where each component of the process is related to the subsystem executing it, using the notation *M(k)>C(k,i)*—that is, capability *C(k,i)* is performed by conceptual subsystem (or module) *k*.

A modeler using table notation—such as the one used in Tables (b) and (c)—can tell only a partial story regarding the invocation of system's capabilities. This is because such tables do not provide a detailed specification of the events and conditions governing the invocation and termination of the capabilities. This information is provided in control tables located in the model's database.

4. Reuse of conceptual models and specifications

Development of the conceptual model and preparation of the system specification are intensely rewarding activities, despite the extensive work required. The reward is especially great when the developed system belongs to a product family. In product families, conceptual models can be reused in the development of new systems with similar characteristics, considerably reducing development time and development risks [7]. The authors have demonstrated such dramatic reduction by developing a conceptual model of a new member of a complex product family in one month. The development of the original model took one year. Conceptual models can also be mapped to different system designs to allow rapid development of products. The mapping of the conceptual model to the design model is described in [8].

5. Products of the ECSAM modeling and analysis process

The main products of the ECSAM modeling and analysis process are the specifications of the system and its components. These specifications support the analysis process, describe the models, and summarize the requirements that have been discovered using the method. ECSAM provides a framework for systematic generation of such specifications.

Although documentation is frequently the main tangible product of an analysis process, it is not a goal of the ECSAM process, but rather its side-effect or derivative. What motivates the process is the need for consistent and comprehensive models as well as manageable requirements that can be used to describe and understand the system and its underlying structural and operational concepts.

6. Overview of the ECSAM process

The ECSAM process consists of several well-defined activities, beginning with identification of stakeholders and elicitation of their requirements, proceeding to modeling and requirements refinement, and ending with a top-level system and software design.

The process steps described in this paper have proven their effectiveness in many projects. Nevertheless, the order in which individual steps may be applied should be adapted to suit a project's particular needs.

Understanding and knowledge gained during an analysis step often sheds light on earlier steps. To best exploit this knowledge, one must sometimes reiterate previous steps—a feature that is inherent to the ECSAM

method. The iterations cease, usually very rapidly, once the internal consistency of the model is established. Since ECSAM models represent real-world systems (which must be internally consistent, if they are expected to work), convergence of the aforementioned iterations is assured.

The basic activities used in each process step are described and explained (including an example of a real-world system) in [1]. Modeling of the system is carried out as a sequence of steps that form two major phases. In the first phase, the system's environmental model (the E-model) is developed. The environmental model describes the system's external structure and behavior, as seen by operators and users. In the second phase, the internal conceptual model of the system, known as the S-model, is developed. This model describes the system's internal conceptual structure and dynamics. Concurrently, the system's behavior is simulated, technical requirements are derived from the system's model and recorded, and the top-level design model is developed.

The main steps of the ECSAM iterative analysis process are

1. definition of the system's mission, scope, and establishment of external requirements and constraints
2. definition of the system's boundaries, identification of the environmental systems with which the system interacts and of the system's major inputs and outputs, development of the system's context diagram and the resulting environmental system's module-chart
3. identification of the system's externally observable capabilities, called E-level system capabilities
4. determination of basic, externally observable operating modes (E-level modes) and the transitions between them, and determination of the E-level capabilities available in each E-level mode
5. specification of the system's environmental processes (E-level processes), and derivation of use cases that describe dynamic interactions between the environmental systems (such as operators) and the system
6. systematic derivation of the system's operational scenarios
7. elicitation of the system's external requirements from stakeholders, their refinement using the E-model, extraction of requirements from the environmental model, and preparation of the system's external specifications
8. simulation of the system's external dynamics using CAS^{2}E tools[4]
9. generation of the system's external specifications, based on the environmental model stored in the CAS^{2}E tool
10. identification of conceptual subsystems (objects), their capabilities, and the major internal signals and information-flows exchanged by them
11. determination of basic, internal operating modes, called S-level modes
12. specification of the system's internal dynamic processes (the S-level processes) and their invocation
13. simulation of the system's internal behavior based on the dynamic model
14. specification of the system's top-level design architecture and mapping of the conceptual model elements to the design architecture
15. refinement of system's requirements through flowdown

7. Supporting tools

The ECSAM analysis process produces large quantities of information about the system being analyzed at a surprisingly high rate. The volume of information and its nontrivial relationship to various views and models practically mandates the use of computer-based tools.

Automated tools and skill sets that are needed by the analyst include

- graphic editors that support drawing the different system views and that capture relevant textual information
- precise semantics that allow analysis of the different views and the relationships among them
- algorithms that support the testing of the consistency and completeness of the evolving model
- methods and tools that simulate the dynamic behavior of the system as represented by the system's modes and the transitions between them, as well as the behavior of the dynamic processes
- a database supporting the creation and maintenance of data dictionaries
- interrogation facilities that allow the investigation of various aspects of the evolving model and the requirements associated with it
- report-generating facilities that document the analysis results using varying sets of templates suitable for various applications

ECSAM models and reports can be created using widely available word processing programs that provide graphic support. A major disadvantage to such programs, however, is their inability to analyze the consistency and completeness of a model. They are also unable to enforce

the precise semantics needed in various diagrams, often making the resulting specifications ambiguous, difficult to analyze and maintain, and their reuse in other systems highly impractical.

CASE tools, providing varying levels of support of the ECSAM method, have been developed over the years. An advanced CAS^{2}E tool, called Statemate®[5], supports most aspects of the ECSAM method.

Comprehension of the complex dynamic aspects of a system's model can be significantly enhanced during early stages of development by tools that support simulation or animation of the modeled system's behavior. Our experience shows that the effort invested in modeling and simulation provides significant gains in later phases of development by reducing rework needed to correct erroneous system behavior caused by incorrect specifications.[6] These gains increase when tools based on precise and rich semantics—allowing detailed modeling and simulation of behavior—are used.

The successful implementation of ECSAM depends heavily on the availability of appropriate CAS^{2}E tools during the early phases of the project, when the method is first put into use.

8. The history of the ECSAM modeling method

An early version of the ECSAM modeling method was first used in 1980 at Israel Aircraft Industries (IAI) by its principal developer the First author of this paper and his team as they sought an effective method to model, analyze, and design complex, reactive, embedded systems and their software, as well as a method to communicate effectively with systems users and stakeholders who participated in specifying requirements for the systems. The method evolved over the years, drawing upon experience gained during application of the ECSAM method on projects performed by IAI.

Further development of the ECSAM generic model led in 1983 to a cooperative effort between Dr. Lavi and Professor David Harel of the Weizmann Institute of Science. The collaboration consequently resulted in development of the Statecharts formalism and the subsequent development of Statemate.

Following Dr. Lavi's retirement from IAI the authors continued to develop ECSAM. Significant extensions of the model and the analysis method—such as the introduction of the E-level and S-level models concept, and systematic derivation of operational scenarios—were added by the authors in the late nineties, in response to needs expressed by industrial projects.

9. Conclusions and future research

Present-day systems developers frequently face growing challenges, as the complexity of the systems they are tasked to develop increases from year to year, while the required time-to-market and development budgets shrink. One of the consequences of this situation is the need to use methods that allow rapid development of "the right system", one that will meet the stakeholders' needs and expectations, on the first attempt. Failure to do so frequently results in cancellation of projects, in expensive budget and schedule overruns, or in deficient functionality and performance of systems that are eventually completed. Analysis of the causes that brought about such failures usually identifies inadequate requirements and specifications as the root cause. A Standish Group study, quoted in the IEEE Software Magazine [9], found out that in 1995 America spent $81 billion for cancelled development projects and $59 billion for projects completed late, over budget, and lacking essential functionality – in the area of software systems alone! Among the primary causes for those gigantic losses, the study identified incomplete and changing requirements and specifications, as well as insufficient user involvement in the specification of the systems.

ECSAM describes a systematic approach to the analysis of systems that provides a solution to these problems, by allowing practitioners to identify the properties that a system must possess to be able to carry out its mission, and translate these properties into requirements and specifications.

Certain themes are stressed in the ECSAM approach:

- The starting point for the specification and analysis of a system is the identification of its mission(s). Specification of the system's mission(s) determines the external behavior the system must exhibit to carry out these missions. Once the missions were specified, analysis commences with the identification of the system's boundaries, its functions and other properties the system must possess to implement that behavior.
- Stakeholders must be involved in the specification of the system's mission(s), its boundaries and other essential properties, as well as in the review of the static and dynamic aspects of the system's conceptual model from the onset of development. Their involvement in the specification and analysis of the system's externally observable properties and behavior and in the specification of its

requirements is vital in assuring that "the right system" is being specified.

- Identical views and similar methods are used in the modeling and analysis of the external properties of a system as well as in the derivation and modeling of its internals.
- Analysis of the conceptual model provides the basis for the identification of essential properties the modeled system must possess to be able to carry out its missions. Furthermore, the conceptual model provides a framework for systematic derivation, evaluation and refinement of the system requirements.
- Operational scenarios, derived from a system's conceptual model, describe its external behavior and its interaction with its environment. The hierarchical organization of the model allows verification of consistency between scenarios that describe the external behavior of the system and the properties of its subsystems at various levels.
- Formal languages must be used in the specification of conceptual models to enable automatic checking of their internal consistency and completeness, and provide for simulation of the dynamic behavior specified by the model, using computer-based tools.

Although the ECSAM method was originally developed to solve problems encountered in the analysis of embedded systems, it has proven very effective in the analysis of systems of all kinds, including systems that contained no computers. This is hardly surprising, since the method is, actually, a general way of thinking about systems, as attested by former students who have learned the method over the years.

Future research related to ECSAM will involve the use of the UML-2 modeling language in the creation of ECSAM models.

10. Acknowledgements

Thanks are due to our many colleagues in various plants of Israel Aircraft Industries and its Corporate Embedded Computer-Systems R&D Department, whose technical needs, suggestions, and discussions stimulated development of many of the ideas and methods that make up ECSAM and whose projects provided the necessary testbed for evaluation of the methods in real-world situations. Special thanks are due to Prof. David Harel, who developed Statecharts at IAI as part of the ECSAM development effort, and to Prof. Amir Pnueli, whose farsighted thinking and advice have contributed immeasurably to our work. Thanks go as well to the many students who participated in ECSAM courses—from the early 1980s at IAI through the latest series of classes at the Jerusalem College of Technology.

11. References