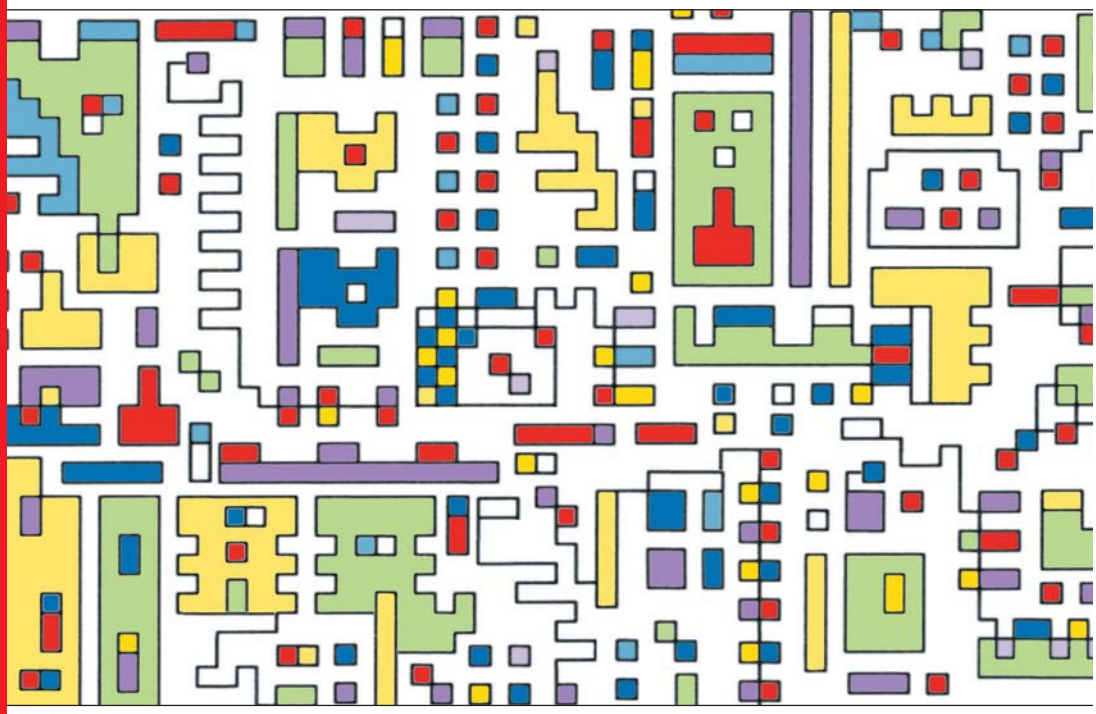
MARINA CABRINI, GIOVANNI FRANZA, PAOLO SCHGÖR,
EUGENIO SCHININÀ, GIUSEPPE ZERBI

THE ALL-ROUND IT PROFESSIONAL

Part B. Build Knowledge Area: Acquisition, Development and Implementation of Information Systems



AICA
Associazione Italiana
per l'Informatica ed
il Calcolo Automatico

EUCIP
European Certification of Informatics Professionals
Core Component

FrancoAngeli

MARINA CABRINI,
GIOVANNI FRANZA,
PAOLO SCHGÖR,
EUGENIO SCHININÀ,
GIUSEPPE ZERBI

THE ALL-ROUND IT PROFESSIONAL

Part B
Build Knowledge Area:
Acquisition, Development and Implementation of Information Systems

FrancoAngeli

This publication is based on:

Marina Cabrini, Giovanni Franza, Paolo Schgör, Eugenio Schininà, Giuseppe Zerbi, *Professione informatica. Vol. II – Realizzazione di sistemi informativi. Competenze interdisciplinari per l'applicazione delle tecnologie dell'informazione e della comunicazione nel mondo del lavoro*, FrancoAngeli, Milano 2004.

This publication is a **working translation** of the above text in the English language and do not pretend to do credit to the original Italian text. It has been prepared for the following purpose and target groups:
• Trainers and Lecturers: to allow them to prepare didactic material and courseware for the EUCIP Core training courses.
• Candidates for EUCIP Core certification: to be used as learning material to supplement EUCIP Core training.

Development:
The EUCIP Core Syllabus at [www.eucip.com] specifies the content of the EUCIP Core certification domain. This English language working translation can be used in conjunction with the EUCIP Core Syllabus as a basis to prepared further courseware in any language and in particular English language courseware.

ISBN: 978-88-464-8560-1

Summary

Introduction

The Need for Common Competences and References

The spread of IT has brought a very wide public to face arguments (ranging from the concept of *bit* to the JPEG format) which were, up to not too many years ago, a prerogative of a few specialists.

The growing IT literacy that characterizes advanced societies does not mean however that specialized competences are not needed anymore. On the contrary, the demarcation line between "users" and "professionals" is thicker and thicker, and the differences between distinct professional specializations are such that a network administrator for a large company has very little in common with a Java programmer involved in the integration of information systems at some other company or with a pre-sale consultant working for a company that develops and commercializes CAD systems.

The risk that exists in this context is a great confusion, in which many individuals think they have good IT competences, but they are unable to communicate (due to language issues to begin with) with other groups of theoretically analogous people, who are also specialists in fields that may differ from the IT sector only for some marginal details.

The task of assessing IT competences turns out to be even more difficult (maybe) for those who are alien to the field. Suffice it to think of those who are called to promote learning initiatives in order to favour social development, or of human resource managers attempting to select candidate employees or to establish criteria for internal appraisal and stimulations of professional excellence.

It is therefore important to define a reference outline for helping one to identify some firm points of cross-sectional competences

common to all IT professionals, that is those who do not simply use IT for their work, but are instead IT craftsmen themselves. All the technicians who work for companies of the IT industry as well as all the staff dedicated to IT support at companies and organizations operating in other industries fall into this category.

The definition of such a reference outline for IT competences is the objective that has carried in 2000 to the formation of an international working group promoted by CEPIS (the Council of European Professional Informatics Societies). The outcome is the EUCIP (European Certification of Informatics Professionals) program, described in the last two sections of this introductory chapter.

Book Structure

The contents of this book correspond, also in the organization of arguments, to the Syllabus (included as an appendix) that defines the basic competence requirements necessary to get the EUCIP certification. Due to the vastness of the arguments which are dealt with, it has been chosen to subdivide the book in three volumes. Such a subdivision, in addition to corresponding to the EUCIP base level structure (which consists of three distinct examinations in the “Plan”, “Build” and “Operate” areas), also reflects a logical distinction between discipline contexts.

This first volume deals with topics relating to the planning, to the use, and to the management of information systems and it therefore exposes a number of elementary concepts on information processing service “clients”. An overview is thus given on organizations, on business process management, on project management, on legal and economic implications of IT investments, often from a “consulting” point of view, in the conviction that IT specialists must understand the real requirements and the context to which technology is intended.

The second volume deals with arguments related to the realization of information systems, with particular emphasis on software, meant as a development object.

Finally, the third and last volume of the series deals with problems related to operation and operating support of the information systems, emphasizing hardware components, operating systems, communication

networks, and the delivery modalities of a support service oriented to a customer-supplier logic, which is already indicated as necessary in this first volume.

Acknowledgements

Many people have contributed with several suggestions and comments to writing this text: Raffaele Brmabilla, Cino Bocchi and Enrico Carrara. The authors also wish to acknowledge the entire AICA (Italian Association for IT and Automated Processing) structure, for sponsorship and support. A particularly warm thanks goes to AICA president Giulio Occhini, who has the merit of having been the first to realize the necessity of defining reference criteria for IT competences and of having constantly and tenaciously devoted energies and resources to the EUCIP project, as well as to several projects which led to the definition of the ECDL (European Computer Driver Licence) certification, from the core level up to the IT Administrator level, which is synergic to many of the contents of EUCIP certification itself.

EUCIP Certification Structure

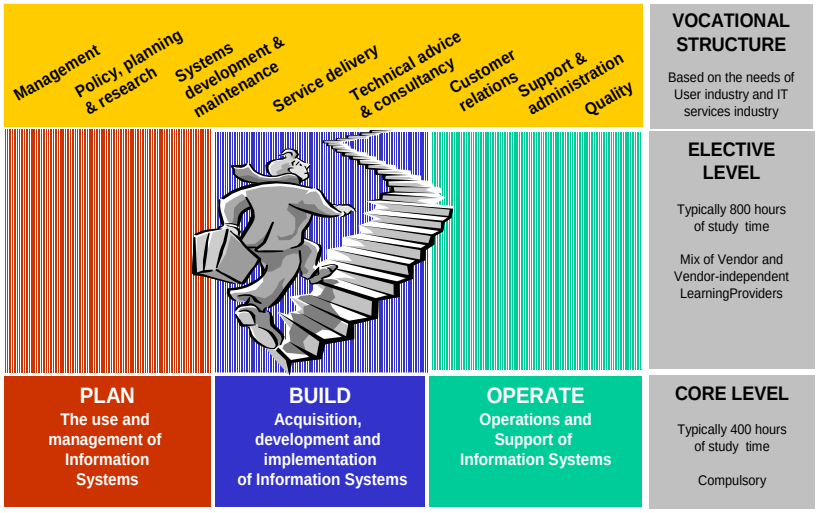


fig.I.1 –EUCIP conceptual structure

The EUCIP programme includes two certification levels, the second of which refers to a vocational structure.

- **Core level**: includes necessary competences, common to all of the paths, and covers the three fundamental processes: plan, build, and operate. A 60 minute exam corresponds to each of these three areas with multiple choice questions extracted from a question and test base (QTB). Preparation for the core level exam is estimated to require about 400 study hours, equally distributed among the three processes cited, for a university level student.
- **Elective level**: the elective level permits the choice of a specialised competence. The student has the power to arrange, with some level of freedom, elements of different areas of competence, including vendor modules as well as independent courses and modules. The overall path is expected to take about 800 hours of study, and the final exam for the certification is taken in the presence of an EUCIP examination board. The candidate provides the board with documentation that certifies the path studied and the realized projects prior to the exam. The combination of courses chosen by the student must correspond to one of the prescribed curricula (see the next point), thus guiding the candidate to a determined professional IT category.
- **Vocational structure**: the elective profiles, like, for example, Business Analyst, Information Systems Project Manager, Software Developer or Network Manager, correspond to typical profiles within businesses of the IT industry or other sectors. This structure of the EUCIP elective profiles aims to define the correspondence between the professional figures and the concrete requirements on various areas of knowledge and within the described elective level. The form is typically that of a study curriculum that describes how one can combine available modules to satisfy the requirements of the chosen position. For example, the profile of Network Administrator requires various education modules in the “Operate” area.

The Three Knowledge Areas

A. **Plan** – It refers to requirements analysis and the planning of the use of information technologies, and it is therefore strictly connected to the management processes and to the definition of the business needs in the ICT sphere put into the context of a strategic perspective. Important elements within this area are, for example, the traditional notions of business organization, return of investments, financing, risk, etc.
B. **Build** – Includes the processes of specification, development and integration of IT systems. The central node of the area is represented by traditional aspects of development, implementation, and integration of IT systems.
C. **Operate** – This area regards the installation, supervision and maintenance of IT systems. It is characterized by arguments like network management, change management, service and delivery support, etc.

Part B
Build Knowledge Area: Acquisition, Development and Implementation of Information Systems

B.1. Systems Development Process and Methods

B.1.1. Application Software and System Software

Even though the notion of software is a concept that is in continuous evolution in step with information systems technologies, it is useful to make an important distinction.

First of all, it is worth highlighting that the main goal is to subdivide the world of programs in order to analyse essential aspects based on characteristics common to each group.

According to its specific use, software can be classified into **application software** and **system software.**

An initial way of defining the two software classifications is through a relational approach. System software is defined as any program that supports the execution of applications, without being specific to any one application.

If you picture an organisation made up of **layers and levels**, you can identify system software that is positioned at a "low" level, near to the physical resources of the machine that runs them. Application software is positioned at a "high" level, farther from the hardware itself (according to the "onion" model of operating systems Sect. C.2.1.).

It is clear that the definitions significantly depend on the type of machine on which they are executed.

Nevertheless, it is possible to identify a few of the main system functions, including:

- the management of hardware interfaces
- the scheduling of processes

- the allocation of memory
- the user interface if no applications are active.

An operating system is the most common example of system software.

There are different schools of thought regarding the basic functions in the characteristics of the system, some of which include responsibilities, like a graphical user interface. Others would, instead, tend to exclude parts like the loader, Basic Input Output System (BIOS), or boot or installation firmware.

Refer to the bibliography, concerning operating systems, like Linux[1] , or to the virtual community for systems development http://dmoz.org/Computers/Programming/Operating_Systems

On the other hand, it is possible to adopt a definition of **application software** that would include all of the programs that carry out a specific function for the user.

In this case, there are supporters of the theory that a client and a server together form a distributed application. Others argue that the editors and compilers, rather than applications, should be considered tools for construction of other applications.

It is also useful to note that the distinction between system and application software often lies in the fact that one comes from the machine executed in privileged mode, while the other in user or not-privileged mode.

Some programs do in fact fall into each of the two families.

System Software: operating system programs for use on a personal computer, examples of which are Windows and OS7; real time operating systems (RTOS) that support embedded applications; systems like Unix and Linux for professional use and/or for large computers.

Application software: also for this family wide subgroups can be identified, based on their use:

- general purpose applications (also known as individual production tools or systems for office automation):

[1] "Inside Linux : A Look at Operating System Development", *fatbrain.com*, March 1996.

    - word processing
    - spreadsheets
    - presentation tools
- applications for software development or tools for the construction of other programs:
    - assemblers
    - compilers
    - debuggers
    - linkers
    - interpreters
    - supports for the control of the product configuration
    - generators of profiles and code analysers
- information management applications (cfr. A.2.3):
    - enterprise resource planning (ERP) – that include various functionalities in an integrated way, which are sometimes sold separately, like:
    - accounting packages
    - applications for market analysis
    - customer relationship management (CRM)
    - decisions support systems
    - programs that manage projects and the coordination of groups and resources
- tools for professional use, scientific and/or engineering calculations:
    - computer-aided design (CAD)
    - programs for statistical analysis
    - tools for project management
- publishing and multimedia:
    - tools for text composition
    - tools for manipulating images and audio/video sequences
    - tools for the preparation of didactic material
    - tools for composing multimedia pages
- entertainment applications:
    - electronic games
    - audio and video players and editors

The list of subdivisions described above is far from exhaustive but is an attempted classification of programs that are widely bought and used today.

Syllabus Items

- Recognise and describe the difference between systems software and application software
- Name some examples of both categories
- Explain the use of application and systems software

B.1.2. Systems Development Principles and Methodologies

We will now consider software from the "productive" point of view, as the final object of an intense design and elaboration process to put together a "product" that meets the expectations of the "customer".

We should note that, like all commercial objects, software also undergoes the various evolutional phases of its market.

Therefore, it is possible to make reference to the time periods of introduction, growth, maturity and decline that characterize the generally valid commercial market model (see figure B1.01).

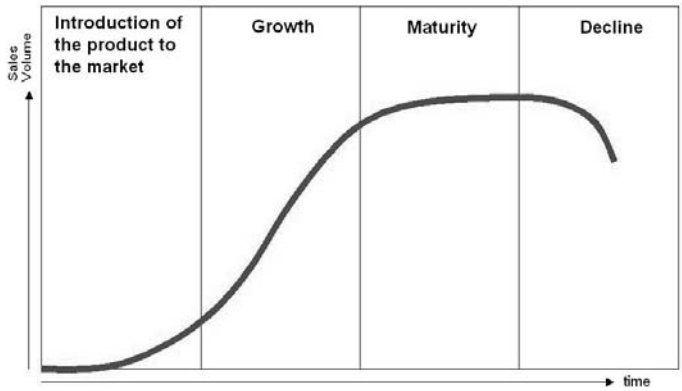


fig.B1.01 – Model applicable to the market evolution of the software product

Probably more so than other products, software is subject to significantly fast changes in the operational conditions and in the

functionalities requested by clients. This places new demands on improvements, adaptations and more generally, modifications that arise during its lifetime.

Therefore, a rationalization of the processes of inception, development, delivery, evolution and maintenance of software is necessary because of the fast changes that software must undergo and because of its complex nature.

In this context, there is a true **life cycle**, subdivided in phases where the following activities are performed (see figure B1.02): inception, study of feasibility, design, development, verification, delivery, maintenance and others, all of which are strictly correlated with the product itself.

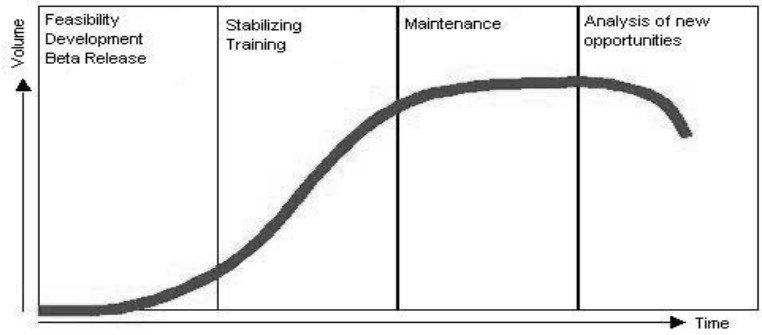


fig.B1.02 – Life cycle of the software product

In the following discussion, the more encompassing term of **system** will be used to include the complexity of systems. In this sense, we will refer to **systems development**.

The subjects which will be covered in depth can often be applied, not only to software products, but also more generally, in fields where technology with systems theories are present.

In the last decades, various schools of thought and approaches to the activity of software development have arisen.

For each of these, we can recall a few of the most significant **models** of reference proposed in the past:

- waterfall
- spiral
- prototyping
- incremental delivery

In the **waterfall** model (see figure B1.03), the various phases are connected in sequence. Each phase transfers to the next phase all of the necessary information to continue the activities.

Also, the handover from one phase to another consists of a revision activity and approval (or a design review), conducted by not only the authors of the activities themselves, but also the people in charge of the next phase.

Remember that we are focusing on the development of programs which are much larger than what can be accomplished by a single person. It is, therefore, necessary to work in teams with adequate organisation structures.

The spiral model (see figure B1.04), illustrated from Boehm[2], and initially applied in structured environments, like the military, allows development to occur in several "turns". With each turn, the classic phases, similar to those of the waterfall model are undergone in a simplified way. One of the major advantages of this approach is that it offers the possibility to produce, at every "turn" a prototype that allows for refinement, for example, of the complete outline of the application, or to focus of a few essential aspects, like the interfaces between other systems.

A particular case of this last consideration is without a doubt that in which the physical machine is still undergoing phases of development and updating requiring a long enough time period to recommend that the development of the application occur in parallel, in order to reduce the overall time-to-market.

[2] Boehm B.W., TRW Defense Systems Group, "A Spiral Model of Software Development and Enhancement", *IEEE Computer*, May 1988.

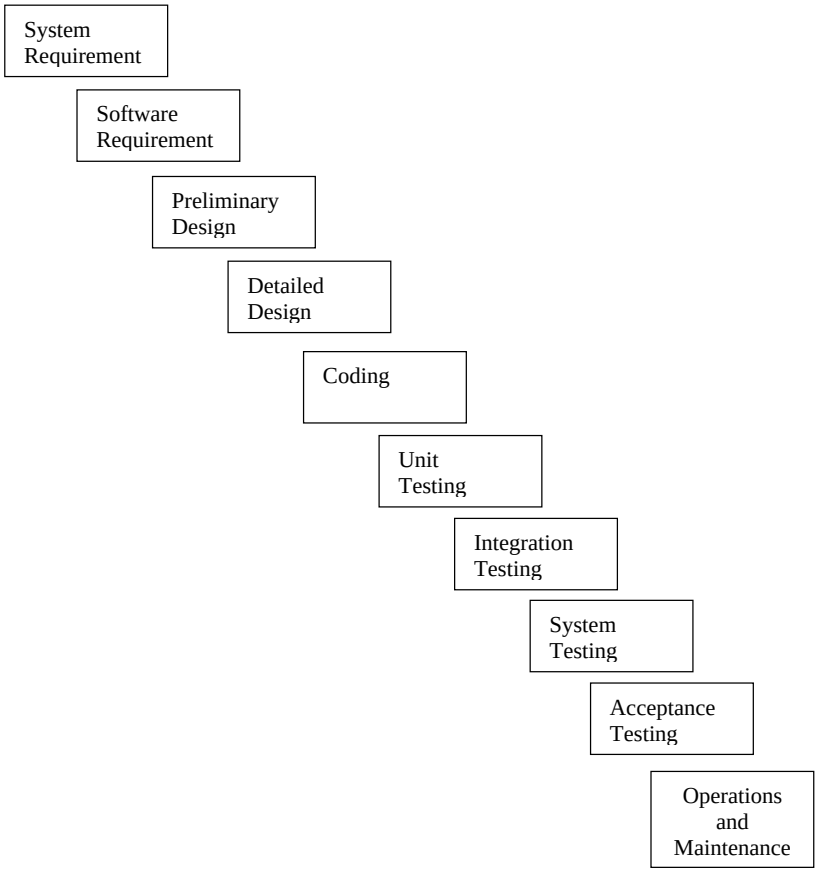


fig.B1.03 – Waterfall development model

Figure B1.04 shows that the main part dedicated to testing resides in the last turn, moving toward the official delivery phase.

Another observation regarding planning activities: these activities generally occur in the final part of each turn; planning activities contribute to the identification of resources necessary for the next phase.