Co-funded by the
Erasmus+ Programme
of the European Union

D5.1 Curriculum development of the short cycle program JAVA DEVELOPER

Project Acronym:	PT&SCHE
Project full title:	Introduction of part-time and short-cycle studies in Serbia
Project No:	561655-EPP-1-2015-1-EE-EPPKA2-CBHE-SP(2015-3431/001-001)
Funding Scheme:	ERASMUS+
Coordinator:	TLU – Tallinn University
Project start date:	October 15, 2015
Project duration:	36 months

Abstract	This report provides the information on developed curriculum of the pilot implementation of the online short-cycle in higher education (SCHE) program JAVA DEVELOPER. Its aim is to provide the qualification of a Java Developer after 12 months with 600 online and F2F hours of education and training, It consists of 13 courses and a Internship lasting two months. The students that successfully submit all assignments and projects for 13 courses and complete it two months internship, is awared with a Certificate. As a pilot program, the curriculum and organization of the SCHE program has been developed according to deliverables of *WP2. Development of legal frameworks for implementation for PT&SCHE*

The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein

DOCUMENT CONTROL SHEET

Title of Document:	D5.1 Curriculum development of the short cycle program JAVA DEVELOPER
Work Package:	WP5. Pilot implementation of online PT & SCHE programs
Last version date:	11/07/2017
Status :	Draft
Document Version:	1.0
File Name	Development of online SCHE JAVA DEVELOPER
Number of Pages	92
Dissemination Level	Institutional

VERSIONING AND CONTRIBUTION HISTORY

Version	Date	Revision Description	Responsible Partner
0.9	1.5.2017	Concept development	Dragan Domazet, BMU
1.0	11.7.2017	Completed draft version	Dragan Domazet, BMU

TABLE OF CONTENT

CURRICULUM DEVELOPMENT OF SCHE "JAVA DEVELOPER"

1 SPECIFICATION OF THE ICT JOB PROFILE: DEVELOPER

1.1 Relevant EU Policy Documents

1.1.1 European ICT Professional Profiles

"**European ICT Professional Profiles**", CWA 16458,is the second relevant document that is the CEN Workshop Agreement document (CEN stands for European Committee for Standardization). This Workshop Agreement has been endorsed by the National Members of CEN, but this is not t an official standard developed by CEN and its Members. The following paragraphs are the citations from this document:

"As a response to the huge number of ICT Profile Frameworks and Profile descriptions used today in European ICT Business and Qualification systems, it was decided to create a number of representative ICT Profiles covering, at their level of granularity, the full ICT Business process.

The profiles may be used for reference, or for the basis to develop further profile generations, by European stakeholders. Structured from six main **ICT Profile families**, these Profiles reflect the top of a **European ICT Profiles family tree** (Figure 1.1.). The concept devised is broadly analogous to human genetics where the genes of one generation pass down to the next. In the same way it is envisaged that the core components of the 23 Generation 2 Profiles will pass down to profiles constructed to meet specific stakeholder requirements. The 23 Profiles constructed in this CWA combined with e-competences from the e-CF, provide a gene pool for the development of tailored profiles that may be developed by European ICT sector players in specific contexts and with higher levels of granularity.

The 23 multi-stakeholders agreed that ICT Profile descriptions are based on the European e-Competence Framework (e-CF). European ICT Profiles and e-Competence are complementary concepts that can significantly support the development and management of a world class ICT professional community within Europe.

Applied at the same level of granularity as the e-CF, the European ICT Profiles provide generic skeletons of the most representative Profile prototypes currently used in ICT Business structures."

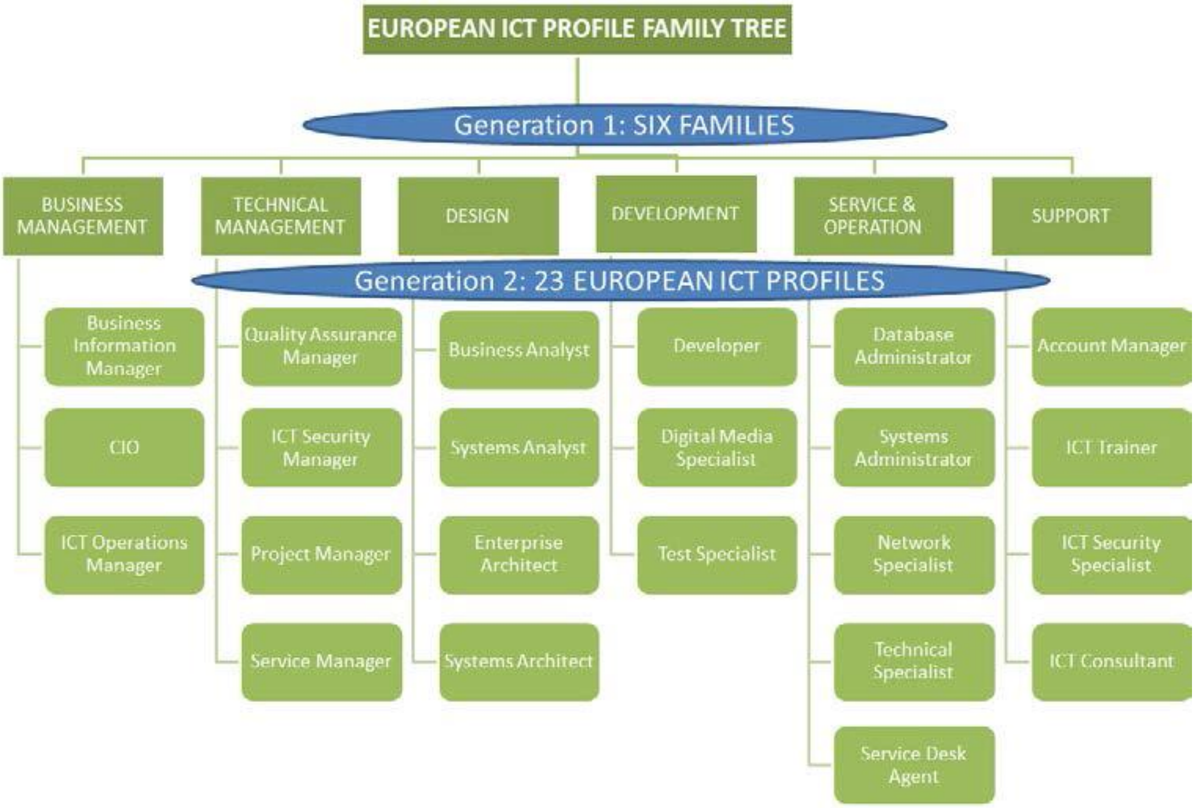


Figure 1.1 European ICT Profile Family Tree – Generation 1 and 2 as a shared European reference

“To add value, the European ICT Profiles must be adaptable to the employment environment. They are not useful if, on the contrary, the employer has to change practices to meet profile descriptions.

The European ICT Profile descriptions are therefore reduced to core components and constructed to clearly differentiate one from each other. Further context-specific elements can be added to the Profiles according to the specific environments in which the Profiles are to be integrated. Clause 4 explains how the European ICT Profiles can be used and adapted by any European stakeholder from a business, qualification or from a research perspective.

The 23 Profiles cover the full ICT Business process; positioning them into the e-CF Dimension 1 demonstrates this. Figure 1.2 below illustrates this together with the ICT Profiles family structure.

The European ICT Profiles build a consistent *bridge between existing competence and profile approaches*. In some European Countries, job **profile creation** is deployed as the traditional methodology for identifying and driving both organisational career paths and educational curriculum. Other countries deploy **a competence-oriented approach,** appreciating that the competence approach provides more flexibility.

In the European ICT Profiles development, the advantages of both approaches have been combined. The European ICT Profiles present e-Competences in an operational context. e-Competences provide the European ICT Profiles with core content in terms of capabilities needed to successfully perform a role. This provides the flexibility to make Profiles applicable EU-wide yet usable in a workplace environment.

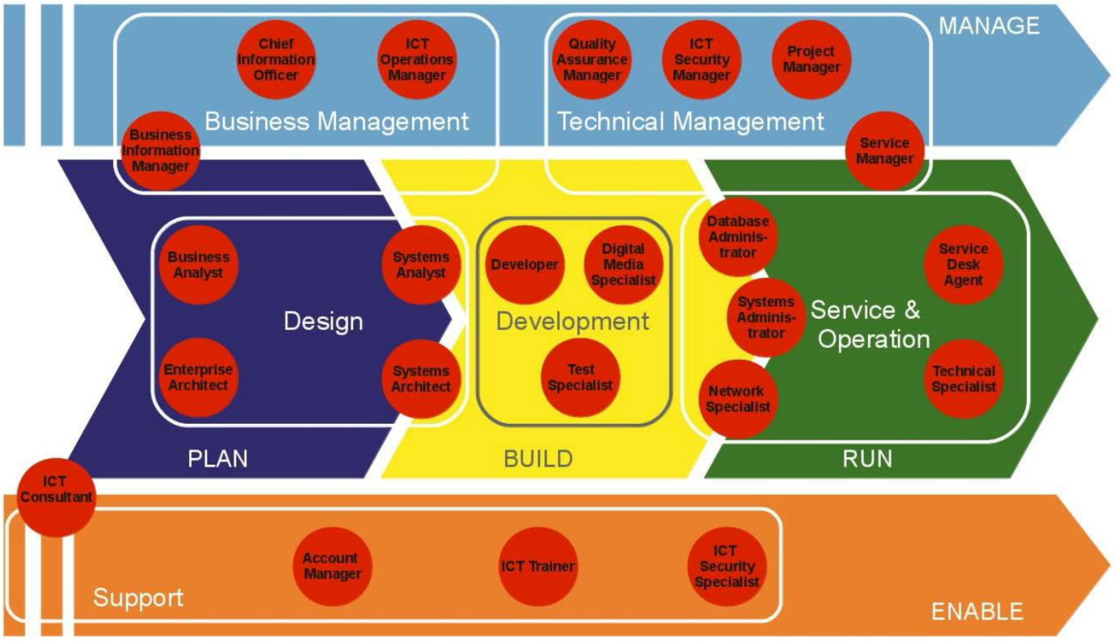


Figure 1.2 European ICT Professional Profiles structured by six families and positioned within the ICT Business Process (e-CF Dimension 1)

By embedding e-Competence within ICT Profiles, which can be readily understood by experts or laymen, the European ICT Profile Family provides a universally applicable solution for communication between stakeholders with interests in ICT skills, knowledge and attitude development.”

ICT Profiles are not totally isolated from each other. Those that interact with each other more closely, create a Profile Cluster. Figure 1.3 shows some of Profiles Clusters from the Design and Development Profile families.

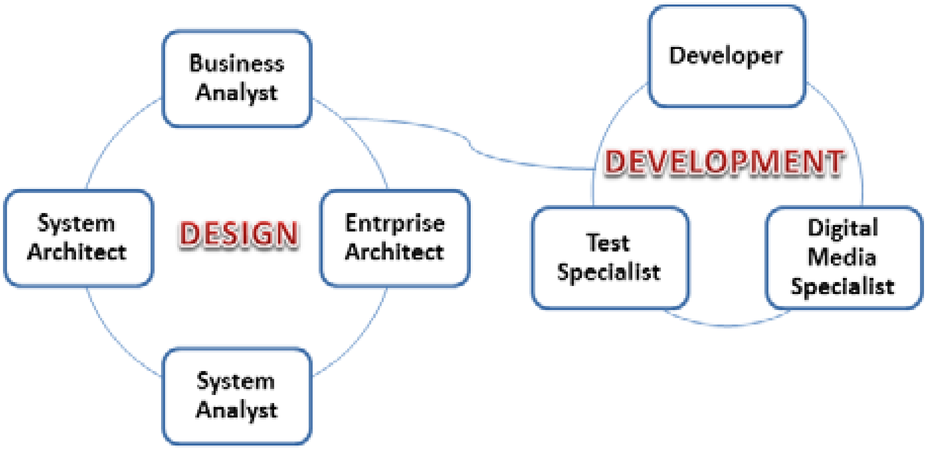
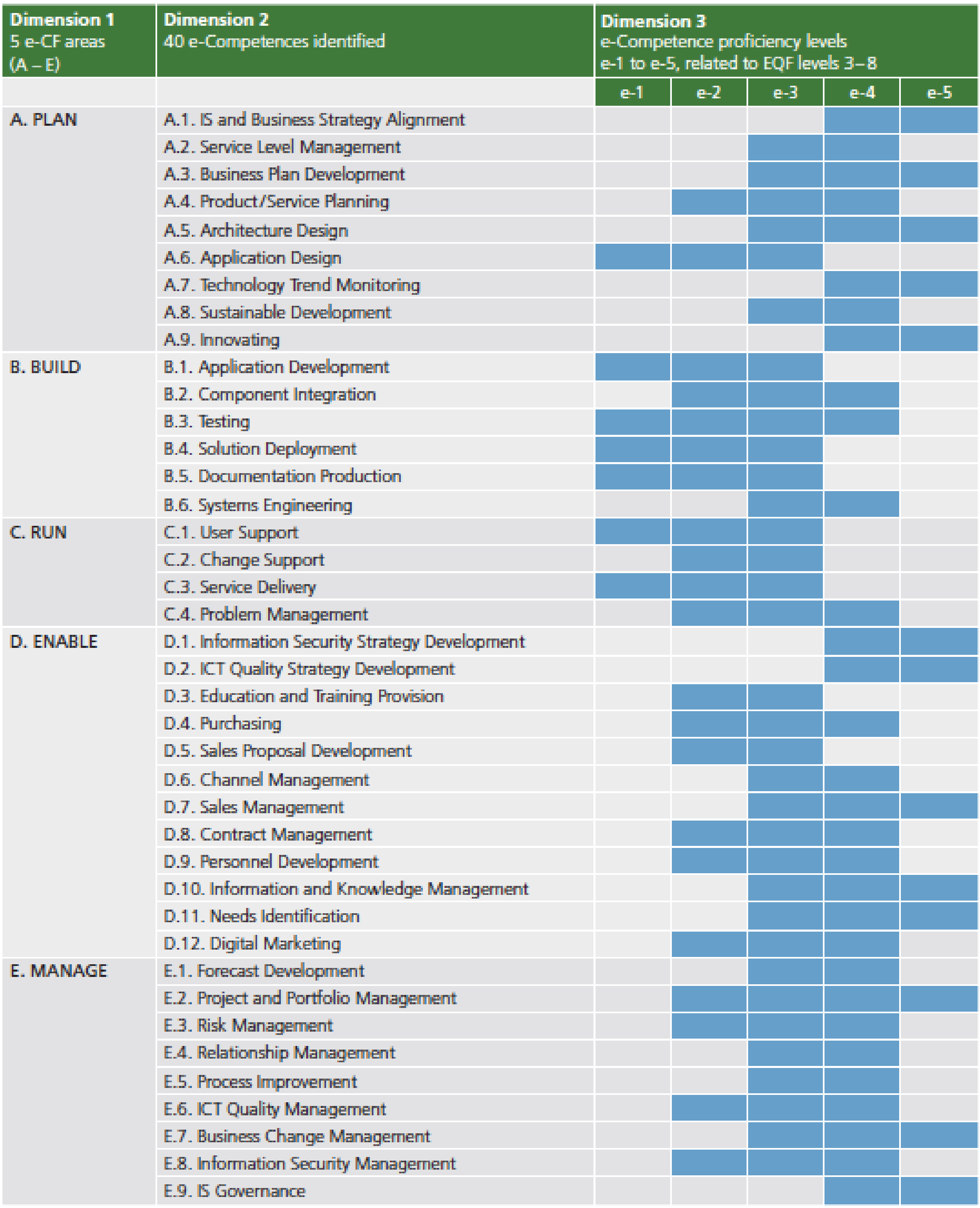


Figure 1.3 ICT Profile Clusters related to Design and Developmenti Profile families.

1.1.2 The European e-Competence Framework

The CWA (CEN Workshop Agreement) document: “**The European e-Competence Framework (e-CF) version 3.0**” is the result of 8 years continuing effort and commitment by multi-stakeholders from the European ICT sector.



Dimension 1 5 e-CF areas (A – E)	**Dimension 2** 40 e-Competences identified	**Dimension 3** e-Competence proficiency levels e-1 to e-5, related to EQF levels 3–8				
		e-1	e-2	e-3	e-4	e-5
A. PLAN	A.1. IS and Business Strategy Alignment				■	■
	A.2. Service Level Management			■	■	
	A.3. Business Plan Development			■	■	■
	A.4. Product/Service Planning		■	■	■	
	A.5. Architecture Design			■	■	■
	A.6. Application Design	■	■	■		
	A.7. Technology Trend Monitoring				■	■
	A.8. Sustainable Development			■	■	
	A.9. Innovating				■	■
B. BUILD	B.1. Application Development	■	■	■		
	B.2. Component Integration		■	■	■	
	B.3. Testing	■	■	■	■	
	B.4. Solution Deployment	■	■	■		
	B.5. Documentation Production	■	■	■		
	B.6. Systems Engineering			■	■	
C. RUN	C.1. User Support	■	■	■		
	C.2. Change Support		■	■		
	C.3. Service Delivery	■	■	■		
	C.4. Problem Management		■	■	■	
D. ENABLE	D.1. Information Security Strategy Development				■	■
	D.2. ICT Quality Strategy Development				■	■
	D.3. Education and Training Provision		■	■		
	D.4. Purchasing		■	■	■	
	D.5. Sales Proposal Development		■	■		
	D.6. Channel Management			■	■	
	D.7. Sales Management			■	■	■
	D.8. Contract Management		■	■	■	
	D.9. Personnel Development		■	■	■	
	D.10. Information and Knowledge Management			■	■	■
	D.11. Needs Identification			■	■	■
	D.12. Digital Marketing		■	■	■	
E. MANAGE	E.1. Forecast Development			■	■	
	E.2. Project and Portfolio Management		■	■	■	■
	E.3. Risk Management		■	■	■	
	E.4. Relationship Management			■	■	
	E.5. Process Improvement			■	■	
	E.6. ICT Quality Management		■	■	■	
	E.7. Business Change Management			■	■	■
	E.8. Information Security Management		■	■	■	
	E.9. IS Governance				■	■

Figure 1.4: 40 e-Competences defined by the European e-Competence Framework

The European e-Competence Framework (e-CF) version 3.0 provides a reference of 40 competences as required and applied at the Information and Communication Technology (ICT) workplace, using *a common language for competences, skills and capability levels t*hat can be understood across Europe. As the first sector-specific implementation of the European Qualifications Framework (EQF), the e-CF was created for application by ICT service, user and supply companies, for managers and human resource (HR) departments, for education institutions and training bodies including higher education, for market watchers and policy makers, and other organisations in public and private sectors.

"The e-CF supports the definition of jobs, training courses, qualifications, career paths, formal and non-formal learning paths, certifications etc. in the ICT sector. In this way, local, national, European and global ICT vendor and user companies as well as qualification and certification providers have access to a shared reference."

The European e-Competence Framework is structured from four dimensions (Figure 1.4). These dimensions reflect different levels of business and human resource planning requirements in addition to job / work proficiency guidelines and are

specified as follows:

Dimension 1: 5 e-Competence areas, derived from the ICT business processes PLAN – BUILD – RUN – ENABLE – MANAGE (see Figure 1.2)

Dimension 2: A set of reference e-Competences for each area, with a generic description for each competence. 40 competences identified in total provide the European generic reference definitions of the e-CF 3.0.

Dimension 3: Proficiency levels of each e-Competence provide European reference level specifications on e-Competence levels e-1 to e-5, which are related to the EQF levels 3 to 8. (Table 1.1)

Dimension 4: Samples of knowledge and skills relate to e-Competences in dimension 2. They are provided to add value and context and are not intended to be exhaustive.

Whilst competence definitions are explicitly assigned to dimension 2 and 3 and knowledge and skills samples appear in dimension 4 of the framework, attitude is embedded in all three dimensions.

Table 1.1.

EQF Levels	EQF	e-CF Levels	e-CF Levels descriptions	Typical Tasks
8	Knowledge at the most advanced frontier, the most advanced and specialised skills and techniques tosolve critical problems in research and/or innovation, demonstrating substantial authority, innovation, autonomy, scholarly or professional integrity.	**e-5**	**Principal** Overall accountability and responsibility; recognised inside and outside the organisation for innovative solutions and for shaping the future using outstanding leading edge thinking and knowledge.	IS strategy or programme management
7	Highly specialised knowledge, some of which is at the forefront of knowledge in a field of work or study, as the basis for original thinking, critical awareness of knowledge issues in a field and at the interface between different fields, specialised problem-solving skills in research and/or innovation to develop new knowledge and procedures and to integrate knowledge from different fields, managing and transforming work or study	**e-4**	**Lead Professional / Senior Manager** Extensive scope of responsibilities deploying specialised integration capability in complex environments; full responsibility for strategic	IS strategy/ holistic solutions

	contexts that are complex, unpredictable and require new strategic approaches, taking responsibility for contributing to professional knowledge and practice and/or for reviewing the strategic performance of teams		development of staff working in unfamiliar and unpredictable situations	
6	Advanced knowledge of a field of work or study, involving a critical understanding of theories and principles, advanced skills, demonstrating mastery and innovation in solving complex and unpredictable problems in a specialised field of work or study, management of complex technical or professional activities or projects, taking responsibility for decision-making in unpredictable work or study contexts, for continuing personal and group professional development.	**e-3**	**Senior Professional / Manager** Respected for innovative methods and use of initiative in specific technical or business areas; providing leadership and taking responsibility for team performances and development in unpredictabl environments**.**	Consulting
5	Comprehensive, specialised, factual and theoretical knowledge within a field of work or study and an awareness of the boundaries of that knowledge, expertise in a comprehensive range of cognitive and practical skills in developing creative solutions to abstract problems, management and supervision in contexts where there is unpredictable change, reviewing and developing performance of self and others.	**e-2**	**Professional** Operates with capability and ndependence in specified boundaries and may supervise others in this environment; conceptual and abstract model building using creative thinking; uses theoretical knowledge and practical skills to solve complex problems within a predictable and sometimes unpredictable context.	Concepts / Basic principles
4	Factual and theoretical knowledge in broad contexts within a field of work or study, expertise in a range of cognitive and practical skills in generating solutions to specific problems in a field of work or study, self-manageme nt within the guidelines of work or study contexts that are usually predictable, but are subject to change, supervising the routine work of others, taking some responsibility for the evaluation and improvement of work or study activities.			
3	Knowledge of facts, principles, processes and general concepts, in a field of work or study, a range of cognitive and practical skills in accomplishing tasks. Problem solving with basic methods, tools, materials and information, responsibility for completion of tasks in work or study, adapting own behaviour to circumstances in solving problems.	**e-1**	Associate Able to apply knowledge and skills to solve straight forward problems; responsible for own actions; operating in a stable environment.	Support / Service

1.2 The role and competences of a Developer

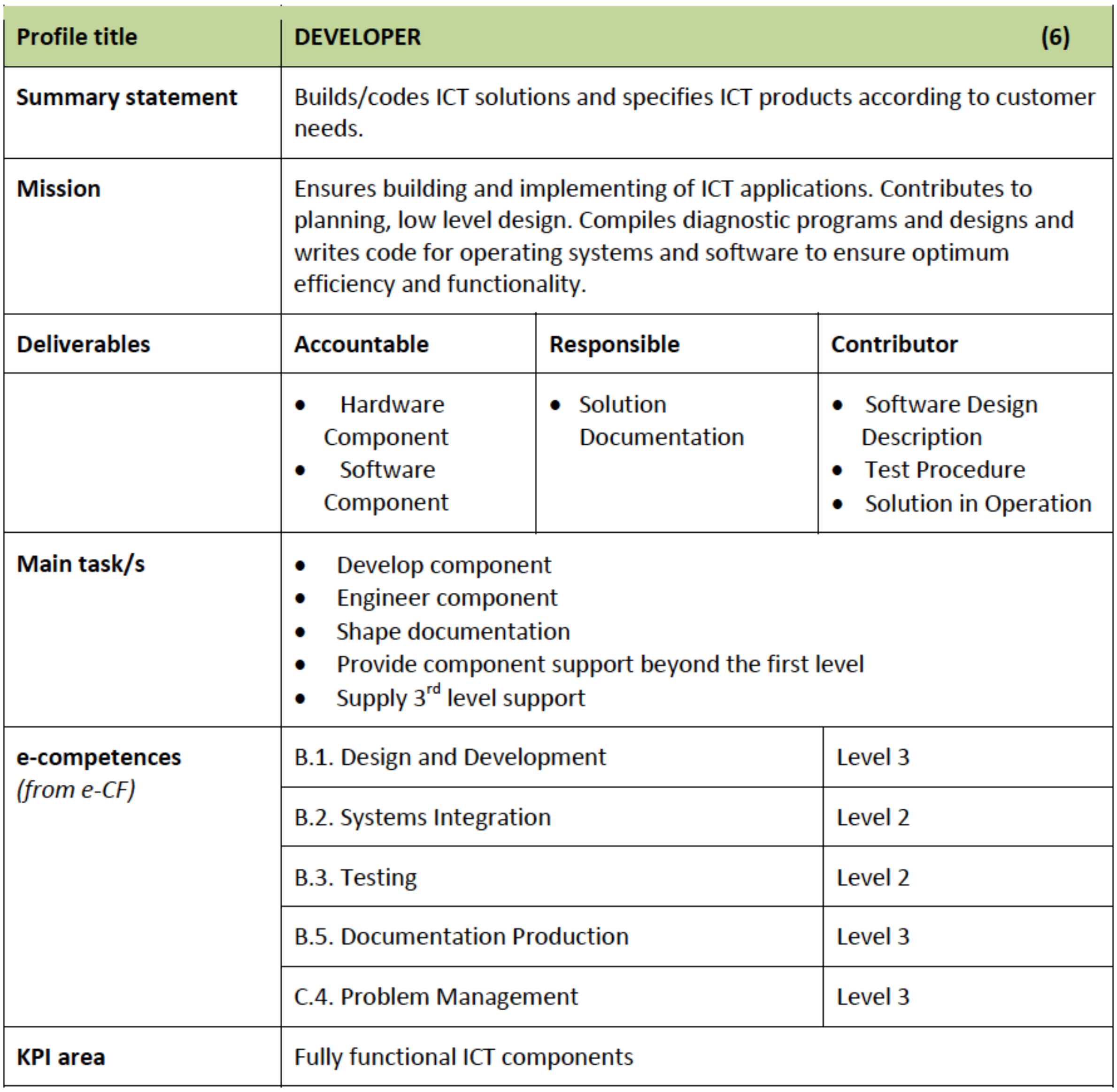
1.2.1 The specification of the profile

ICT Profile Summary statement:

Builds/codes ICT solutions and specifis ICT products according to the customer needs.

Alternative titles:

- Component Developer
- Application Developer
- Programmer



Profile title	**DEVELOPER**		**(6)**
Summary statement	Builds/codes ICT solutions and specifies ICT products according to customer needs.		
Mission	Ensures building and implementing of ICT applications. Contributes to planning, low level design. Compiles diagnostic programs and designs and writes code for operating systems and software to ensure optimum efficiency and functionality.		
Deliverables	**Accountable**	**Responsible**	**Contributor**
	• Hardware Component • Software Component	• Solution Documentation	• Software Design Description • Test Procedure • Solution in Operation
Main task/s	• Develop component • Engineer component • Shape documentation • Provide component support beyond the first level • Supply 3rd level support		
e-competences *(from e-CF)*	B.1. Design and Development		Level 3
	B.2. Systems Integration		Level 2
	B.3. Testing		Level 2
	B.5. Documentation Production		Level 3
	C.4. Problem Management		Level 3
KPI area	Fully functional ICT components		

Figure 1.5: Job profile specification of a Developer

1.2.2 e-competences required

A Developer must have the following e-competence specified jn the European e-Competence Framework 3.0:

B.1. Design and Development (Level 3)

B.2. System Integration (Level 2)

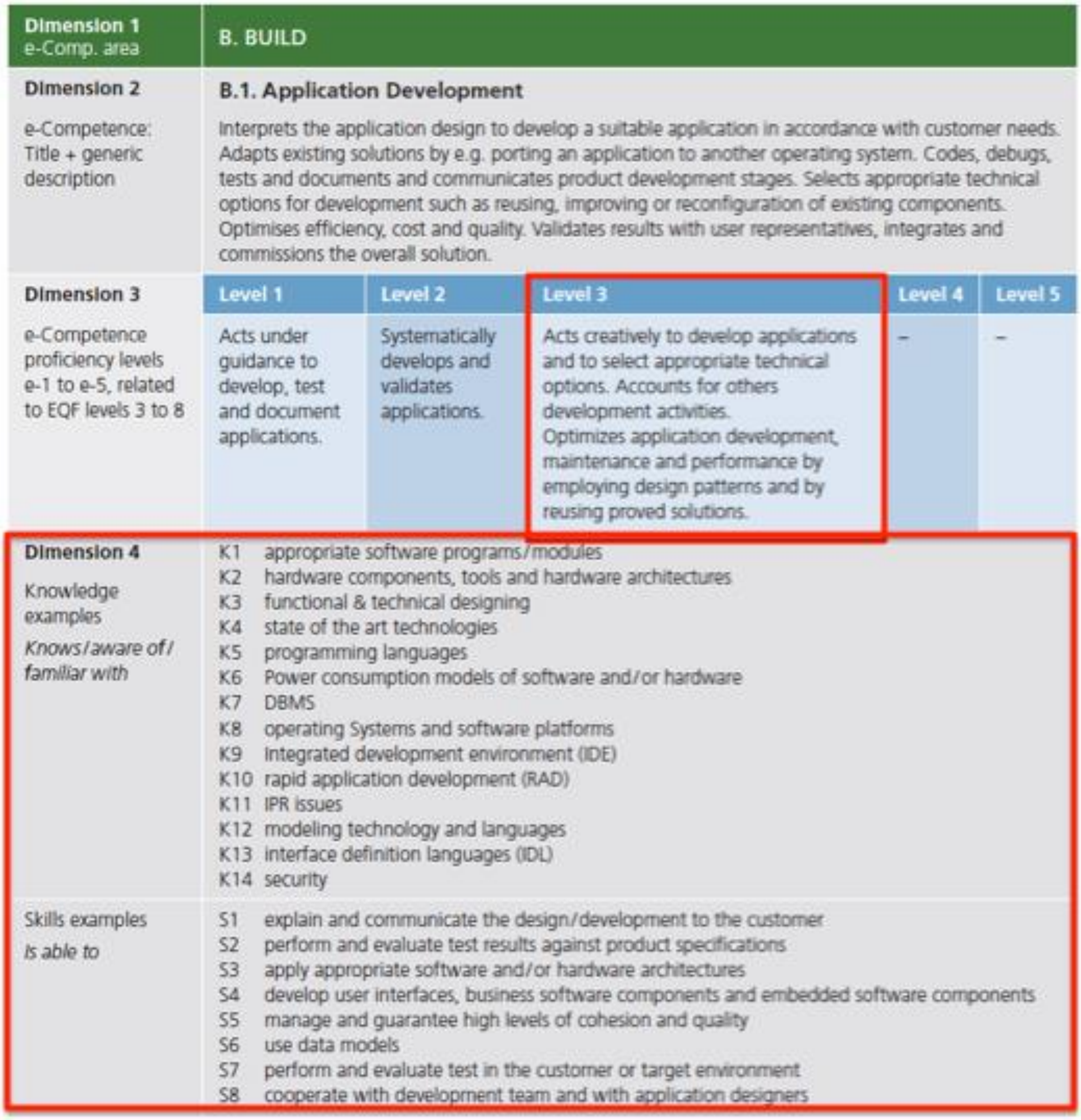
B.3.Testing (Level 2)

B.5. Documentation Production (Level 3)

C.4. Problem Management (Level 3)

For each of these e-competences we cite its specification from the document European e-Competence Framework 3.0.

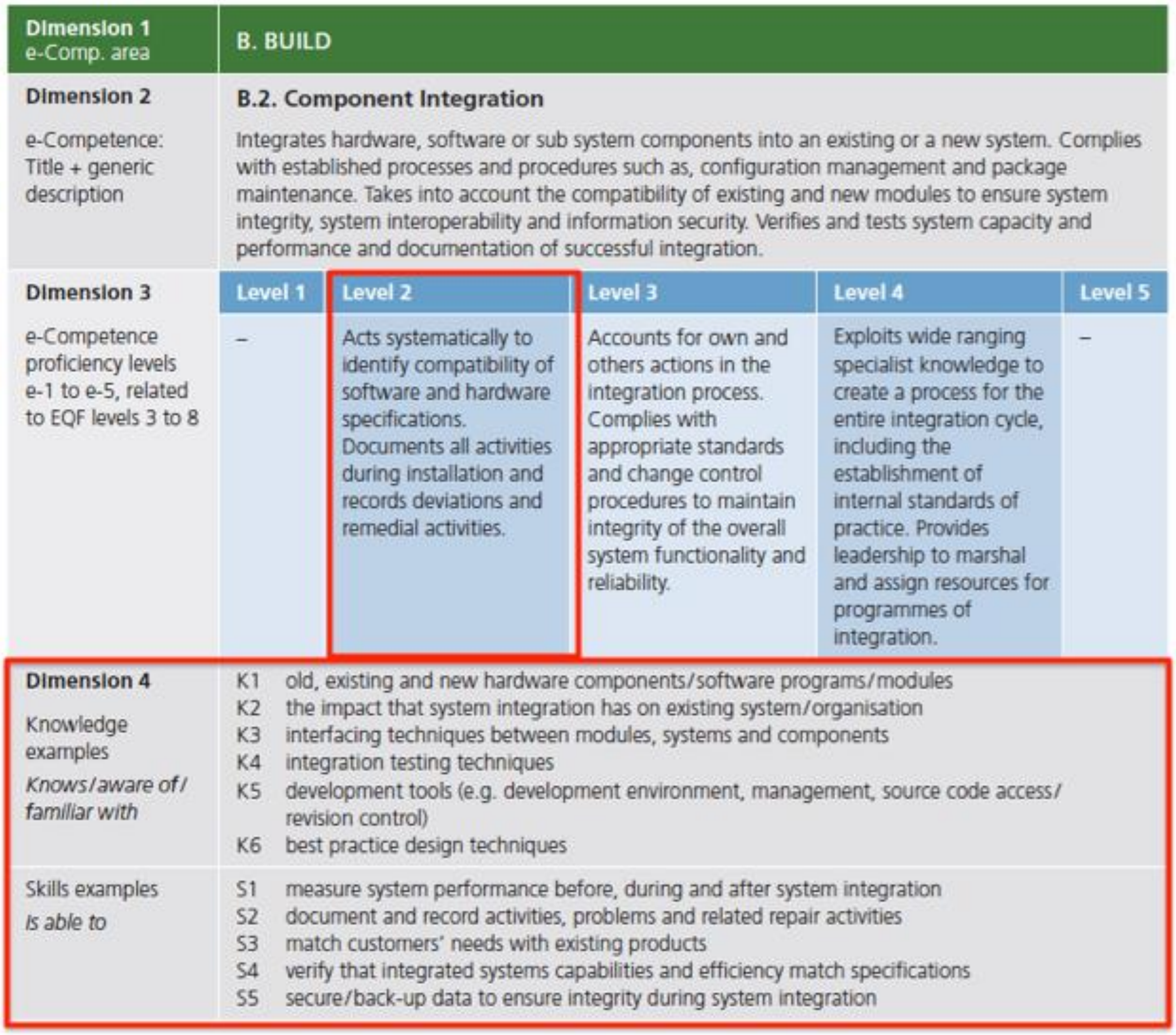
B.1. Design and Development (Level 3)



Dimension 1 e-Comp. area	**B. BUILD**				
Dimension 2 e-Competence: Title + generic description	**B.1. Application Development** Interprets the application design to develop a suitable application in accordance with customer needs. Adapts existing solutions by e.g. porting an application to another operating system. Codes, debugs, tests and documents and communicates product development stages. Selects appropriate technical options for development such as reusing, improving or reconfiguration of existing components. Optimises efficiency, cost and quality. Validates results with user representatives, integrates and commissions the overall solution.				
Dimension 3 e-Competence proficiency levels e-1 to e-5, related to EQF levels 3 to 8	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
	Acts under guidance to develop, test and document applications.	Systematically develops and validates applications.	Acts creatively to develop applications and to select appropriate technical options. Accounts for others development activities. Optimizes application development, maintenance and performance by employing design patterns and by reusing proved solutions.	–	–
Dimension 4 Knowledge examples *Knows/aware of/ familiar with*	K1 appropriate software programs/modules K2 hardware components, tools and hardware architectures K3 functional & technical designing K4 state of the art technologies K5 programming languages K6 Power consumption models of software and/or hardware K7 DBMS K8 operating Systems and software platforms K9 integrated development environment (IDE) K10 rapid application development (RAD) K11 IPR issues K12 modeling technology and languages K13 interface definition languages (IDL) K14 security				
Skills examples *Is able to*	S1 explain and communicate the design/development to the customer S2 perform and evaluate test results against product specifications S3 apply appropriate software and/or hardware architectures S4 develop user interfaces, business software components and embedded software components S5 manage and guarantee high levels of cohesion and quality S6 use data models S7 perform and evaluate test in the customer or target environment S8 cooperate with development team and with application designers				

Figure 1.6: Knowledge and skills needed for e-competence B.1. Application Development

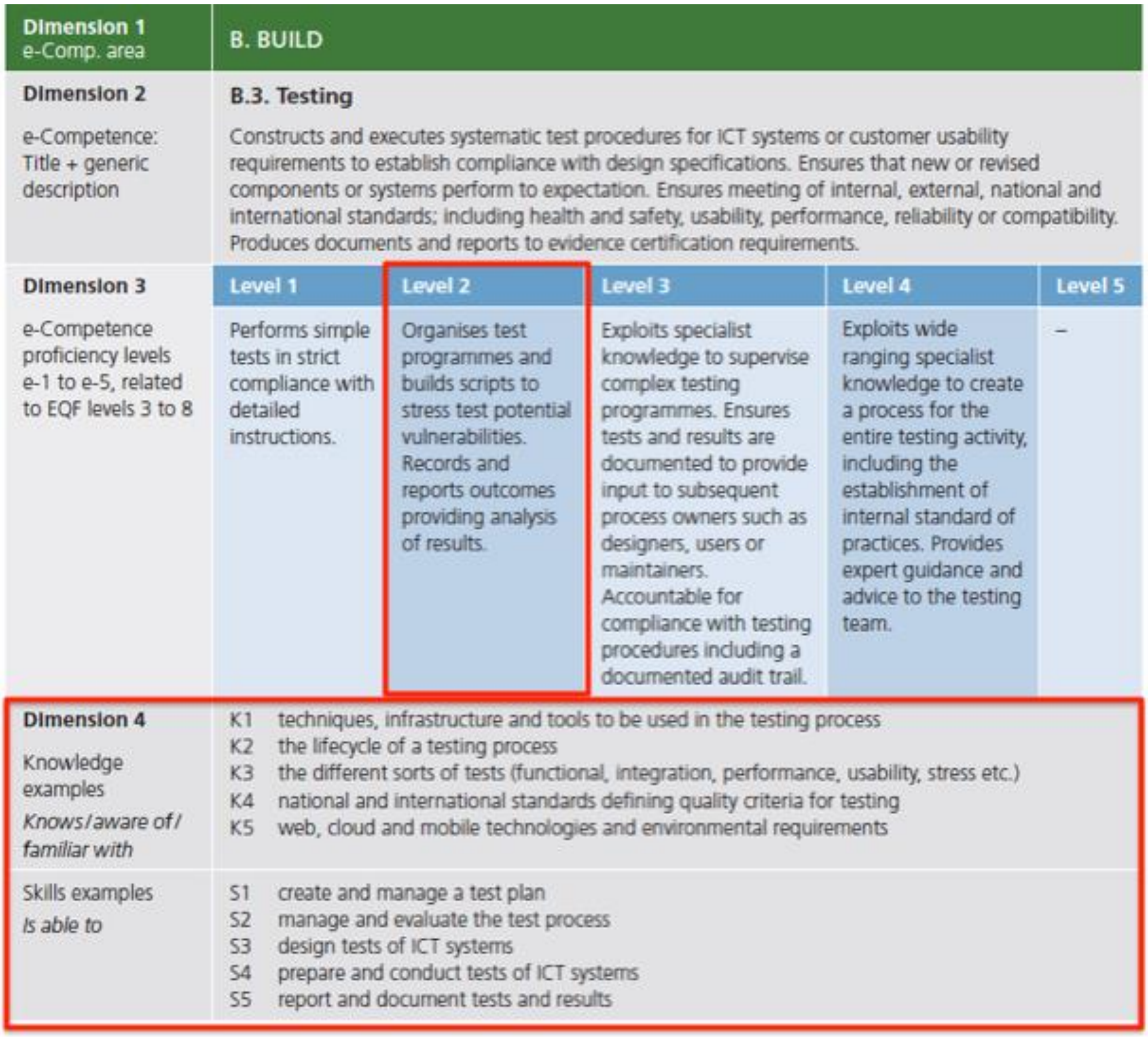
B.2. System Integration (Level 2):



Dimension 1 e-Comp. area	**B. BUILD**				
Dimension 2 e-Competence: Title + generic description	**B.2. Component Integration** Integrates hardware, software or sub system components into an existing or a new system. Complies with established processes and procedures such as, configuration management and package maintenance. Takes into account the compatibility of existing and new modules to ensure system integrity, system interoperability and information security. Verifies and tests system capacity and performance and documentation of successful integration.				
Dimension 3	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
e-Competence proficiency levels e-1 to e-5, related to EQF levels 3 to 8	–	Acts systematically to identify compatibility of software and hardware specifications. Documents all activities during installation and records deviations and remedial activities.	Accounts for own and others actions in the integration process. Complies with appropriate standards and change control procedures to maintain integrity of the overall system functionality and reliability.	Exploits wide ranging specialist knowledge to create a process for the entire integration cycle, including the establishment of internal standards of practice. Provides leadership to marshal and assign resources for programmes of integration.	–
Dimension 4 Knowledge examples *Knows/aware of/ familiar with*	K1 old, existing and new hardware components/software programs/modules K2 the impact that system integration has on existing system/organisation K3 interfacing techniques between modules, systems and components K4 integration testing techniques K5 development tools (e.g. development environment, management, source code access/ revision control) K6 best practice design techniques				
Skills examples *Is able to*	S1 measure system performance before, during and after system integration S2 document and record activities, problems and related repair activities S3 match customers' needs with existing products S4 verify that integrated systems capabilities and efficiency match specifications S5 secure/back-up data to ensure integrity during system integration				

Figure 1.7: Knowledge and skills needed for e-competence B.2. Component Integration

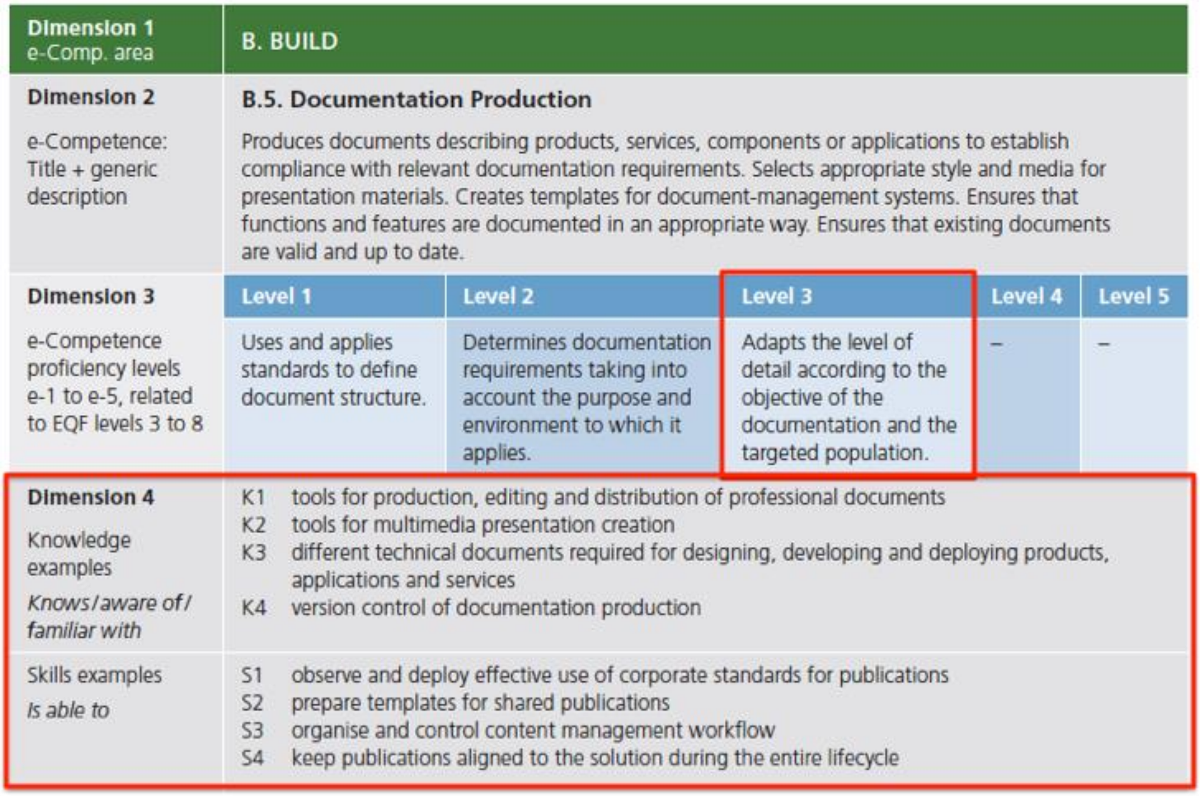
B.3.Testing (Level 2):



Dimension 1 e-Comp. area	**B. BUILD**				
Dimension 2 e-Competence: Title + generic description	**B.3. Testing** Constructs and executes systematic test procedures for ICT systems or customer usability requirements to establish compliance with design specifications. Ensures that new or revised components or systems perform to expectation. Ensures meeting of internal, external, national and international standards; including health and safety, usability, performance, reliability or compatibility. Produces documents and reports to evidence certification requirements.				
Dimension 3	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
e-Competence proficiency levels e-1 to e-5, related to EQF levels 3 to 8	Performs simple tests in strict compliance with detailed instructions.	Organises test programmes and builds scripts to stress test potential vulnerabilities. Records and reports outcomes providing analysis of results.	Exploits specialist knowledge to supervise complex testing programmes. Ensures tests and results are documented to provide input to subsequent process owners such as designers, users or maintainers. Accountable for compliance with testing procedures including a documented audit trail.	Exploits wide ranging specialist knowledge to create a process for the entire testing activity, including the establishment of internal standard of practices. Provides expert guidance and advice to the testing team.	–
Dimension 4 Knowledge examples *Knows/aware of/ familiar with*	K1 techniques, infrastructure and tools to be used in the testing process K2 the lifecycle of a testing process K3 the different sorts of tests (functional, integration, performance, usability, stress etc.) K4 national and international standards defining quality criteria for testing K5 web, cloud and mobile technologies and environmental requirements				
Skills examples *Is able to*	S1 create and manage a test plan S2 manage and evaluate the test process S3 design tests of ICT systems S4 prepare and conduct tests of ICT systems S5 report and document tests and results				

Figure 1.8: Knowledge and skills needed for e-competence B.3. Testing

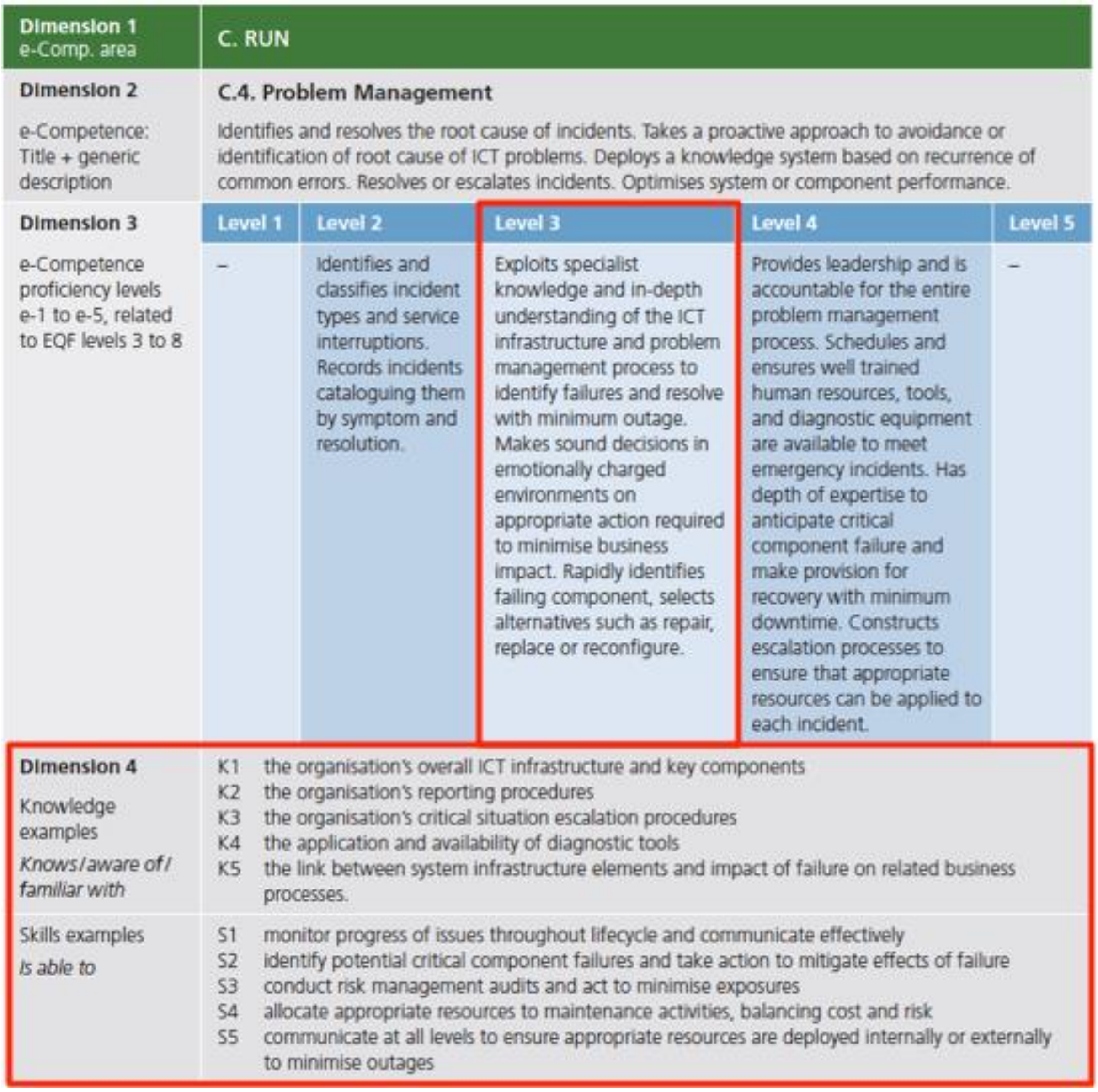
B.5. Documentation Production (Level 3):



Dimension 1 e-Comp. area	**B. BUILD**				
Dimension 2 e-Competence: Title + generic description	**B.5. Documentation Production** Produces documents describing products, services, components or applications to establish compliance with relevant documentation requirements. Selects appropriate style and media for presentation materials. Creates templates for document-management systems. Ensures that functions and features are documented in an appropriate way. Ensures that existing documents are valid and up to date.				
Dimension 3 e-Competence proficiency levels e-1 to e-5, related to EQF levels 3 to 8	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
	Uses and applies standards to define document structure.	Determines documentation requirements taking into account the purpose and environment to which it applies.	Adapts the level of detail according to the objective of the documentation and the targeted population.	–	–
Dimension 4 Knowledge examples *Knows/aware of/ familiar with*	K1 tools for production, editing and distribution of professional documents K2 tools for multimedia presentation creation K3 different technical documents required for designing, developing and deploying products, applications and services K4 version control of documentation production				
Skills examples *Is able to*	S1 observe and deploy effective use of corporate standards for publications S2 prepare templates for shared publications S3 organise and control content management workflow S4 keep publications aligned to the solution during the entire lifecycle				

Figure 1.9: Knowledge and skills needed for e-competence B.5. Document Production

C.4. Problem Management (Level 3):



Dimension 1 e-Comp. area	**C. RUN**				
Dimension 2 e-Competence: Title + generic description	**C.4. Problem Management** Identifies and resolves the root cause of incidents. Takes a proactive approach to avoidance or identification of root cause of ICT problems. Deploys a knowledge system based on recurrence of common errors. Resolves or escalates incidents. Optimises system or component performance.				
Dimension 3	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
e-Competence proficiency levels e-1 to e-5, related to EQF levels 3 to 8	–	Identifies and classifies incident types and service interruptions. Records incidents cataloguing them by symptom and resolution.	Exploits specialist knowledge and in-depth understanding of the ICT infrastructure and problem management process to identify failures and resolve with minimum outage. Makes sound decisions in emotionally charged environments on appropriate action required to minimise business impact. Rapidly identifies failing component, selects alternatives such as repair, replace or reconfigure.	Provides leadership and is accountable for the entire problem management process. Schedules and ensures well trained human resources, tools, and diagnostic equipment are available to meet emergency incidents. Has depth of expertise to anticipate critical component failure and make provision for recovery with minimum downtime. Constructs escalation processes to ensure that appropriate resources can be applied to each incident.	–
Dimension 4 Knowledge examples *Knows/aware of/ familiar with*	K1 the organisation's overall ICT infrastructure and key components K2 the organisation's reporting procedures K3 the organisation's critical situation escalation procedures K4 the application and availability of diagnostic tools K5 the link between system infrastructure elements and impact of failure on related business processes.				
Skills examples *Is able to*	S1 monitor progress of issues throughout lifecycle and communicate effectively S2 identify potential critical component failures and take action to mitigate effects of failure S3 conduct risk management audits and act to minimise exposures S4 allocate appropriate resources to maintenance activities, balancing cost and risk S5 communicate at all levels to ensure appropriate resources are deployed internally or externally to minimise outages				

Figure 1.10: Knowledge and skills needed for e-competence B.5. Problem Management

1.3 The Body of Knowledge

Specification of knowledge units and skills provided for each e-competence in the previous section is not enough to specify the curriculum for a short cycle program for a profile. The specifies required knowledge and skills are of very high level and need to be specified at lower levels. This is the mission of a Body of Knowledge of a study program. In our case we can use:

- **The Foundation ICT Body of Knowledge**, Version 1, 22 February 2015, a report prepared for the European Commission, DG Internal Market, Industry, Entrepreneurship and SMEs by the Service Contract: *e-Skills: Promotion of ICT Professionalism in Europe | No 290/PP/ENT/CIP/13/C/N01C011* prepared by Capgemini Consulting and Ernst & Young.
- **The Software Engineering Body of Knowledge – SWEBOK 3.0,** specified by the IEEE Computer Society - see P. Bourque and R.E. Fairley, eds., **Guide to the Software Engineering Body of Knowledge, Version 3.0**, IEEE Computer Society, 2014; www.swebok.org.

1.3.1 The European Foundational ICT Body of Knowledge

The European Foundational ICT Body of Knowledge is the base-level knowledge required to enter the ICT profession and acts as the first point of reference for anyone interested in working in ICT'.

The ultimate objective is to create a recognised and supported Foundational ICT Body of Knowledge that:

- Serves as an entry point to get into ICT for anyone contemplating a career in ICT and entering from other professions or wanting to digitise their current job;
- Facilitates communication between and understanding of ICT professionals in Europe in whatever sector they are active, thereby reducing risks and strengthening ICT professionalism;
- Increases the supply and pool of ICT professionals and enhances the image of ICT.

The definition of an ICT Professional is defined, as someone who should:

- Possess a comprehensive and up-to-date understanding of a relevant body of knowledge;
- Demonstrate on-going commitment to professional development via an appropriate combination of qualifications, certifications, work experience, non-formal and / or informal education;
- Adhere to an agreed code of ethics / conduct and / or applicable regulatory practices; and
- Through competent practice deliver value for stakeholders.

Some of the key challenges for the near future are to:

- Ensure that as many ICT professionals as possible have the necessary relevant knowledge, skills and competence to deliver professional products and service in today's digital economy;
- Improve the quality of the ICT profession;
- Close the ICT resource and skills gap;
- Enhance growth in digital jobs in Europe;

- Improve general ICT knowledge among professionals in other fields of expertise.

The nature of ICT jobs is also changing. It is no longer enough to merely be a technical expert. The industry needs professionals with a diversity of ICT knowledge and skillsx. ICT professionals are also required to understand the business, operational and HR management aspects. Industry is looking for multidisciplinary ICT professionals, dual thinkers (i.e. people who have a good understanding of both business and Technology) or T-shaped persons (see below). ICT is no longer a back office support tool or one department within a company but permeates all the layers and units of a company. ICT has moved itself to the forefront and become a key strategic asset in everyday (professional) life. Therefore, it is no longer sufficient only to have knowledge of one specific ICT domain.

The need for a broad IT systems viewpoint is essential, with the ability to understand the possibilities and constraints of the various technologies and to talk a common language with the diversity of people involved. This was expressed as a concept for the first time by David Guest in 1991xi through the use of the T-shape metaphor, which has been widely adopted since (Figure 1.11).

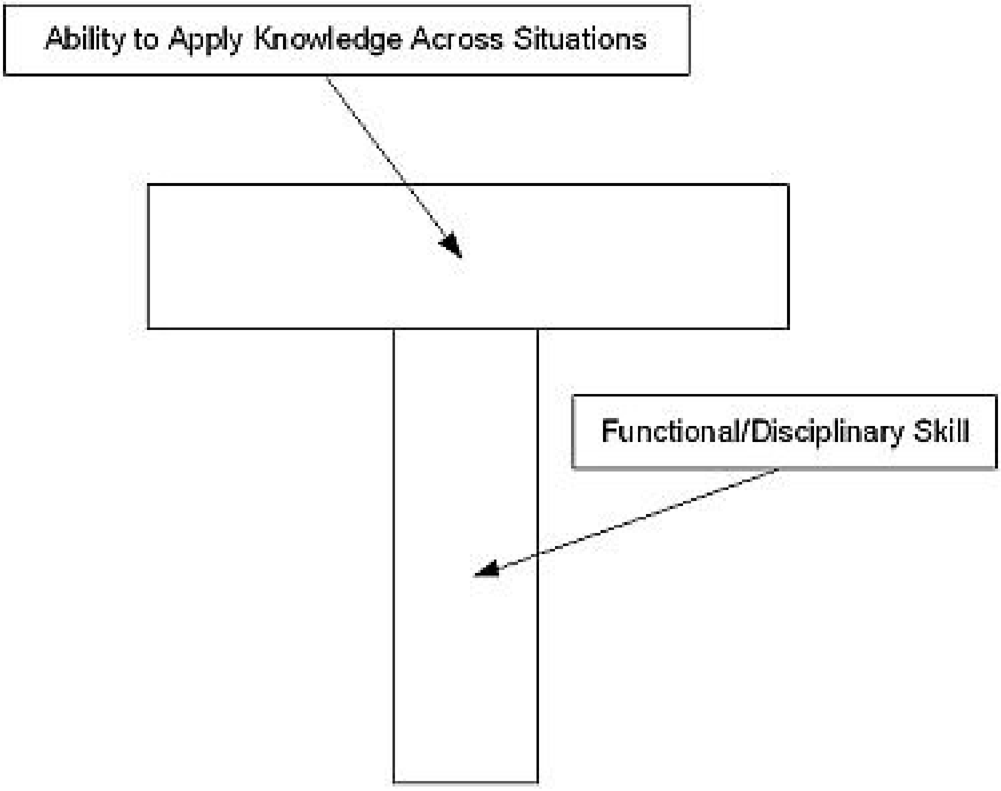

Ability to Apply Knowledge Across Situations
Functional/Disciplinary Skill

Figure 1.11 Shaped Skills Model

The vertical line of the T represents the depth of related skills and expertise in a single field, whereas the horizontal bar is the ability to collaborate across disciplines with experts in other areas and to apply knowledge in areas of expertise other than one's own. This model thus differs from another classic type: “I-shaped” – with a deep understanding of one specific discipline, but not necessarily of any other. In the current ICT environment, employers find themselves trying to do a “T” job with “I” people.

However, a professional who combines specialisation in a specific ICT domain with relevant breadth of ICT knowledge is more easily employable and has a competitive position on the market. Given that there has in the past been a particular focus on depth, it is necessary to look more closely at the issue of breadth of knowledge. It is all a matter of creating the right balance between the two.

The objective is to create T-shaped persons with as much as possible the same elements in the horizontal bar. All ICT professionals should have the same DNA. It is however often the case that ICT professionals have much in common, but have different (job) profiles. The objective of a Body of Knowledge (BOK) is to define the ‘chromosomes’, or building blocks of the horizontal bar, in the ICT field and act as a guide to the breadth of ICT knowledge required.

The EU Foundational ICT Body of Knowledge thus aims to provide guidance for individuals, academia and industry, and hence contribute to developing tomorrow’s multidisciplinary ICT professionals.

The structure of the Foundational ICT Body of Knowledge could be described as an ‘inverted T-model’, in which the horizontal axis shows the knowledge areas of the ICT domain running from a predominantly strategic to a predominantly technological perspective. The vertical axis corresponds to specific knowledge and skills an individual should develop to specialise in one domain. We can assume that any ICT professional wanting to go into a field different from that of their existing specialisation should come down to the horizontal bar (the base-level) and find a connection to other knowledge areas in order to expand their breadth of knowledge.

The Foundational ICT Body of Knowledge provides the base-level knowledge that ICT professionals require. However, considering the wide range of knowledge in the ICT field, it has to be intended as a “permissive model” where every ICT professional will acquire as much breadth as possible in terms of knowledge

In addition to the dimension of ICT core knowledge defined above, the European Foundational ICT Body of Knowledge c<u>onsists of a second dimension of complementary base-level knowledge</u> required to enter the ICT profession. This dimension includes cross-cutting knowledge that cannot be considered purely in relation to one ICT knowledge area but can be referred to, at different levels, in relation to all core knowledge areas, i.e.:

- **Legal, ethical, social and professional practices:** including this knowledge in the Foundational ICT Body of Knowledge serves to provide key reference points for everyone interested in the ICT profession, as they are strongly linked to the definition of the ICT profession itself. Legal, ethical, social and professional practices need to be addressed at different levels at different stages of professional development. Thevery nature of professional work means that some knowledge and skills are best developed through experience and that an understanding of complex issues, such as ethics, grows with maturity. Further development will be provided at a full professional level through participation in certification programmes.
- **Soft skills:** including soft skills in the Foundational ICT Body of Knowledge provides a concrete contribution to the evolution of the ICT profession. Soft skills integrate the technical skills, providing a sound basis for developing

"dual thinker" profiles, which are oriented towards team building, collaboration, negotiation, e-leadership, etc.

- **Emerging / disruptive technologies:** given the fast growth in the disruptive technologies of cloud, mobile, social and big data, which are predicted to constitute 40% of the global market and 98% of growth by 2020, and the expected creation of 4.4 million IT jobs globally to support big data – base-level knowledge should be provided to improve an understanding of these technologies and their impacts on business and society.

The BOK illustrated below (Figure 1.12) and expanded on in the following sections presents the taxonomy of **the high-level areas of knowledge** that represent the base level that starting ICT professionals should understand. These knowledge areas are then broken down and described in further detail, including with a general definition of the knowledge area, a detailed list foundational knowledge, reference to the e-CF, potential job profiles and examples of specific Bodies of Knowledge, certification and training opportunities.

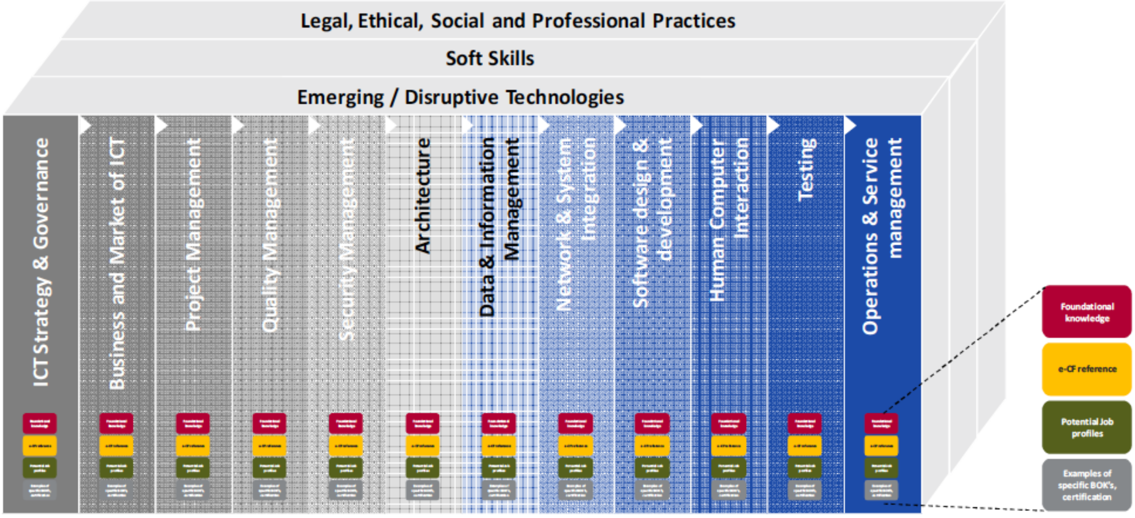

Legal, Ethical, Social and Professional Practices
Soft Skills
Emerging / Disruptive Technologies
ICT Strategy & Governance
Business and Market of ICT
Project Management
Quality Management
Security Management
Architecture
Data & Information Management
Network & System Integration
Software design & development
Human Computer Interaction
Testing
Operations & Service management
Foundational knowledge
e-CF reference
Potential Job profiles
Examples of specific BOK's, certification

Figure 1.12: Taxonomy of Foundational ICT Body of Knowledge

This Body of Knowledge aims to develop the next generation of ICT professionals, e.g. young, rounded ICT professionals with a significant breadth of base-level knowledge of ICT that allows them to further specialize within a particular discipline.

This Version 1.0 of the European Foundational ICT Body of Knowledge presents the taxonomy of high-level areas of knowledge that represent the base level starting ICT professionals should understand.

The following section presents **12 Knowledge Areas**:

1. ICT Strategy & Governance
2. Business and Market of ICT
3. Project Management
4. Security Management
5. Quality Management
6. Architecture
7. Data and Information Management
8. Network and Systems Integration
9. Software Design and Development
10. Human Computer Interaction

11. Testing
12. Operations and Service Management.

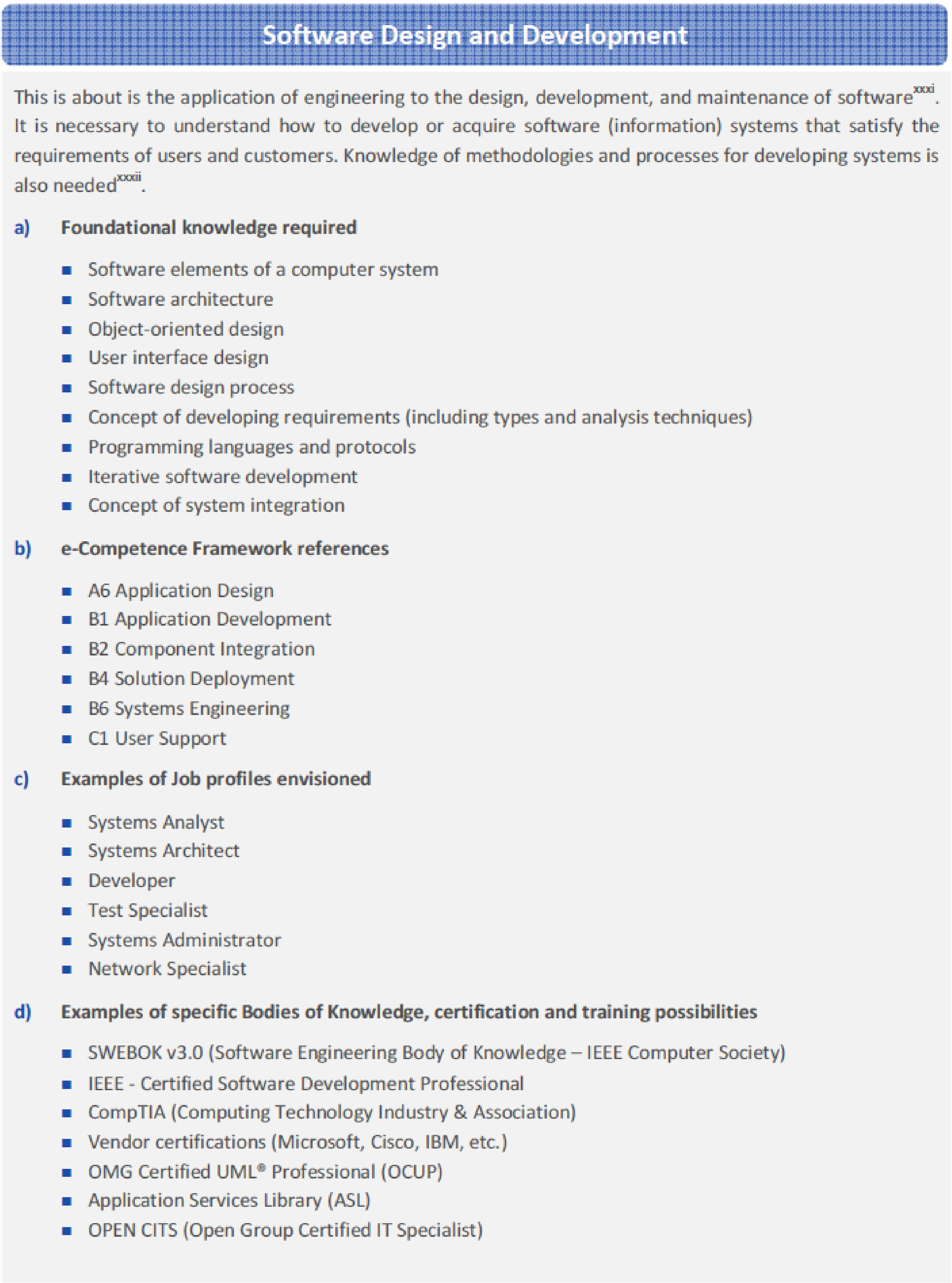
Each **Knowledge Area is further detailed**, including a:

1. Definition of the Knowledge Area;
2. List of items required as foundational knowledge necessary under this Knowledge Area;
3. List of references to the e-Competence Framework (dimension 4: knowledge);
4. List of possible job profiles that require having an understanding of the Knowledge Area;
5. List of examples of specific Bodies of Knowledge, certification and training possibilities.

Figures 1.13-1.116 summarize the content of few Knowledge Areas, the most relevant for the profile Developer:

- Software Design and Development
- Human Computer Interaction
- Data and Information Management
- Testing

These Knowledge Areas provide broader knowledge then needed for the Developer profile, as it is related only to a part of one of five (Build) phases of the ICT Business Process, as shown in Figure 1.2 earlier.



Software Design and Development

This is about is the application of engineering to the design, development, and maintenance of software[xxxi]. It is necessary to understand how to develop or acquire software (information) systems that satisfy the requirements of users and customers. Knowledge of methodologies and processes for developing systems is also needed[xxxii].

a) Foundational knowledge required

- Software elements of a computer system
- Software architecture
- Object-oriented design
- User interface design
- Software design process
- Concept of developing requirements (including types and analysis techniques)
- Programming languages and protocols
- Iterative software development
- Concept of system integration

b) e-Competence Framework references

- A6 Application Design
- B1 Application Development
- B2 Component Integration
- B4 Solution Deployment
- B6 Systems Engineering
- C1 User Support

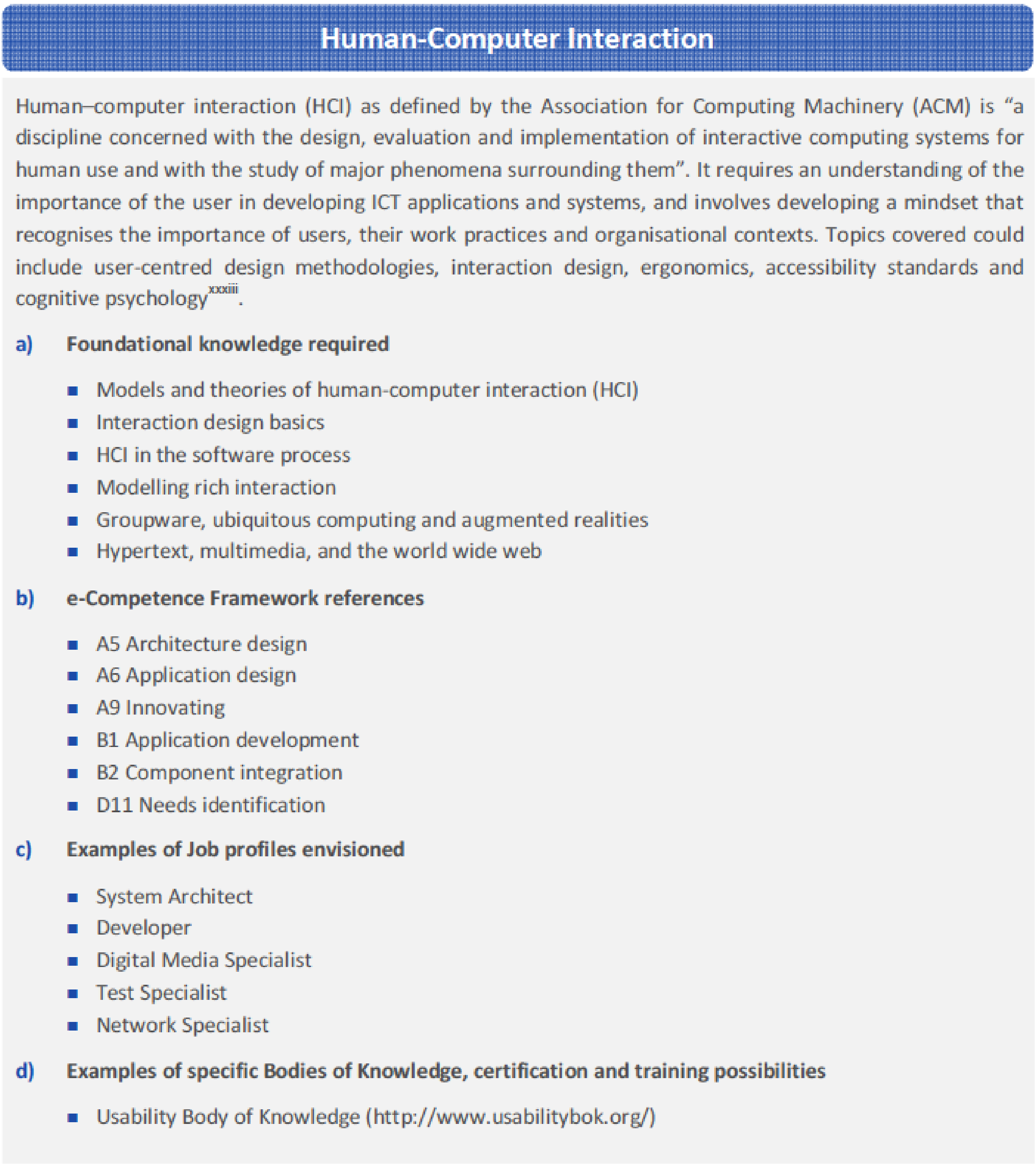
c) Examples of Job profiles envisioned

- Systems Analyst
- Systems Architect
- Developer
- Test Specialist
- Systems Administrator
- Network Specialist

d) Examples of specific Bodies of Knowledge, certification and training possibilities

- SWEBOK v3.0 (Software Engineering Body of Knowledge – IEEE Computer Society)
- IEEE - Certified Software Development Professional
- CompTIA (Computing Technology Industry & Association)
- Vendor certifications (Microsoft, Cisco, IBM, etc.)
- OMG Certified UML® Professional (OCUP)
- Application Services Library (ASL)
- OPEN CITS (Open Group Certified IT Specialist)

Figure 1.13: Software Design and Development Knowledge Area



Human-Computer Interaction

Human–computer interaction (HCI) as defined by the Association for Computing Machinery (ACM) is “a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them”. It requires an understanding of the importance of the user in developing ICT applications and systems, and involves developing a mindset that recognises the importance of users, their work practices and organisational contexts. Topics covered could include user-centred design methodologies, interaction design, ergonomics, accessibility standards and cognitive psychology[xxxiii].

a) Foundational knowledge required

- Models and theories of human-computer interaction (HCI)
- Interaction design basics
- HCI in the software process
- Modelling rich interaction
- Groupware, ubiquitous computing and augmented realities
- Hypertext, multimedia, and the world wide web

b) e-Competence Framework references

- A5 Architecture design
- A6 Application design
- A9 Innovating
- B1 Application development
- B2 Component integration
- D11 Needs identification

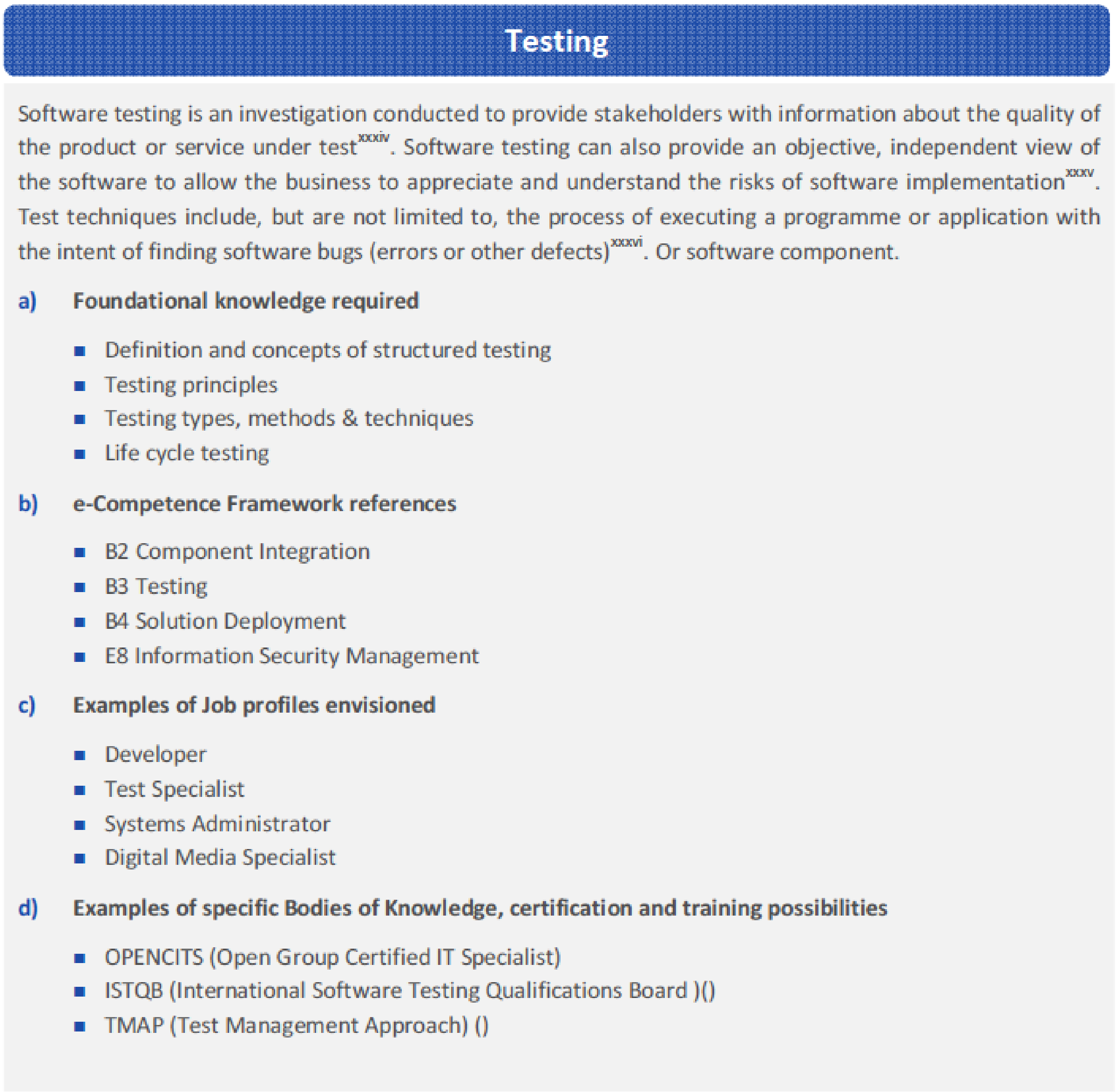
c) Examples of Job profiles envisioned

- System Architect
- Developer
- Digital Media Specialist
- Test Specialist
- Network Specialist

d) Examples of specific Bodies of Knowledge, certification and training possibilities

- Usability Body of Knowledge (http://www.usabilitybok.org/)

Figure 1.14: Human-Computer Interaction Knowledge Area



Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test[xxxiv]. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation[xxxv]. Test techniques include, but are not limited to, the process of executing a programme or application with the intent of finding software bugs (errors or other defects)[xxxvi]. Or software component.

a) Foundational knowledge required

- Definition and concepts of structured testing
- Testing principles
- Testing types, methods & techniques
- Life cycle testing

b) e-Competence Framework references

- B2 Component Integration
- B3 Testing
- B4 Solution Deployment
- E8 Information Security Management

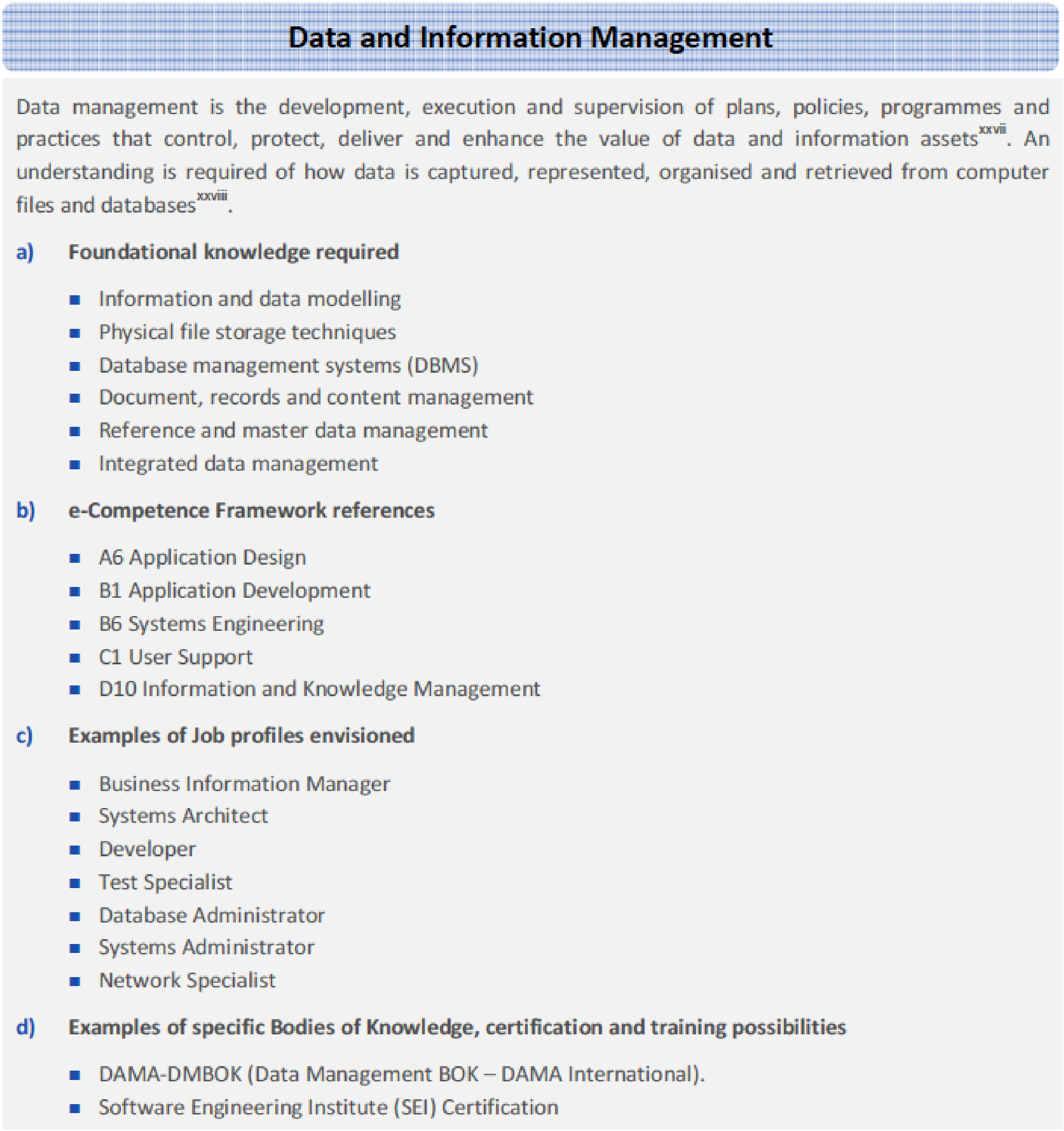
c) Examples of Job profiles envisioned

- Developer
- Test Specialist
- Systems Administrator
- Digital Media Specialist

d) Examples of specific Bodies of Knowledge, certification and training possibilities

- OPENCITS (Open Group Certified IT Specialist)
- ISTQB (International Software Testing Qualifications Board)()
- TMAP (Test Management Approach) ()

Figure 1.15: Testing Knowledge Area



Data and Information Management

Data management is the development, execution and supervision of plans, policies, programmes and practices that control, protect, deliver and enhance the value of data and information assets[xxvii]. An understanding is required of how data is captured, represented, organised and retrieved from computer files and databases[xxviii].

a) Foundational knowledge required

- Information and data modelling
- Physical file storage techniques
- Database management systems (DBMS)
- Document, records and content management
- Reference and master data management
- Integrated data management

b) e-Competence Framework references

- A6 Application Design
- B1 Application Development
- B6 Systems Engineering
- C1 User Support
- D10 Information and Knowledge Management

c) Examples of Job profiles envisioned

- Business Information Manager
- Systems Architect
- Developer
- Test Specialist
- Database Administrator
- Systems Administrator
- Network Specialist

d) Examples of specific Bodies of Knowledge, certification and training possibilities

- DAMA-DMBOK (Data Management BOK – DAMA International).
- Software Engineering Institute (SEI) Certification

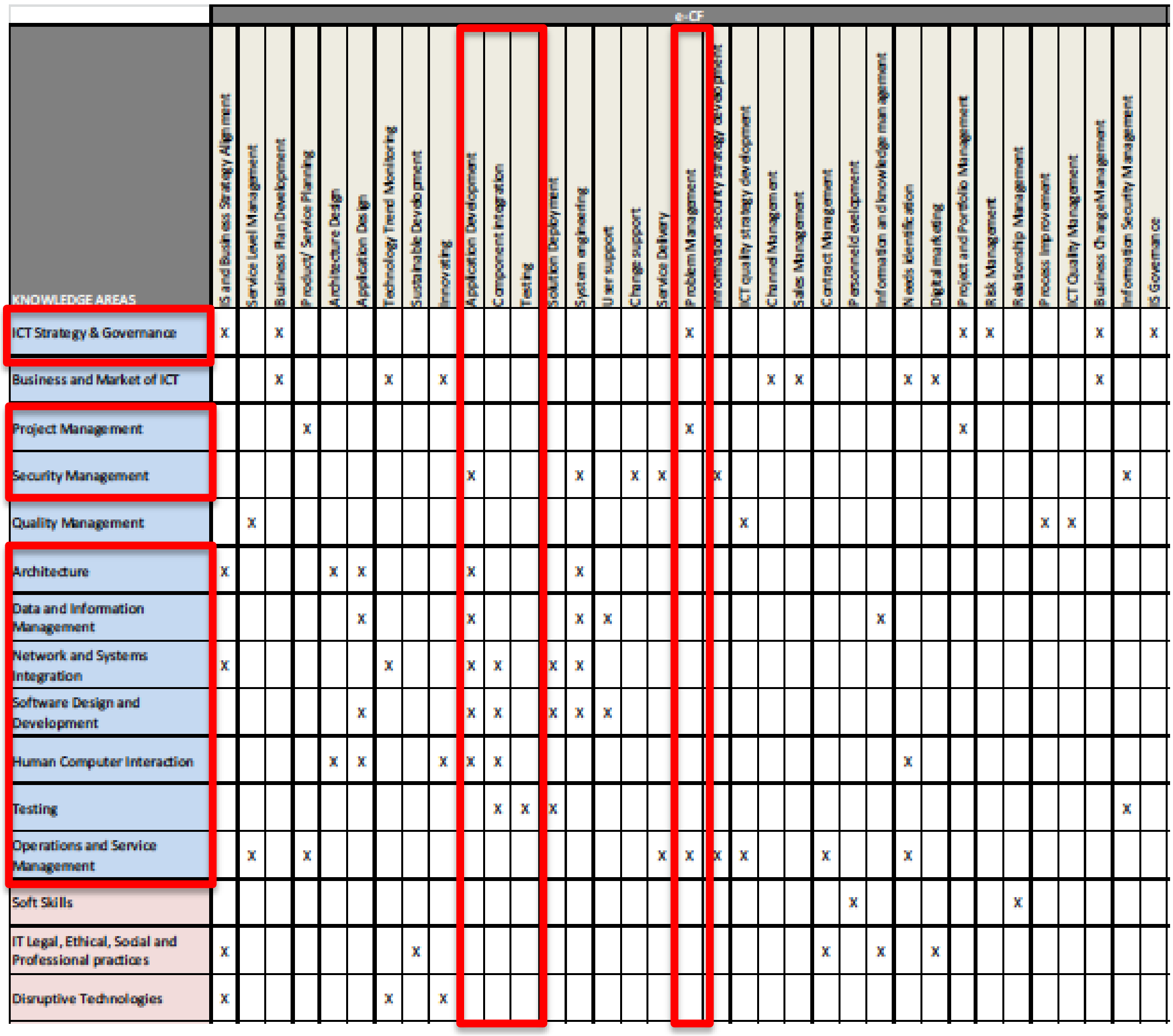
Figure 1.16: Data and Information Management Knowledge Area

As specified earlier, five ICT e-competences are required for the profile Developer:

B.1. Design and Development (Level 3)
B.2. System Integration (Level 2)
B.3.Testing (Level 2)
B.5. Documentation Production (Level 3)
C.4. Problem Management (Level 3)

Figure 1.17 shows relationships of these five e-competences and 10 Knowledge Areas of the ICT Foundation Body of Knowledge. It does nit mean the profile Developer must know everything specified in these 10 Knowledge Areas. In so

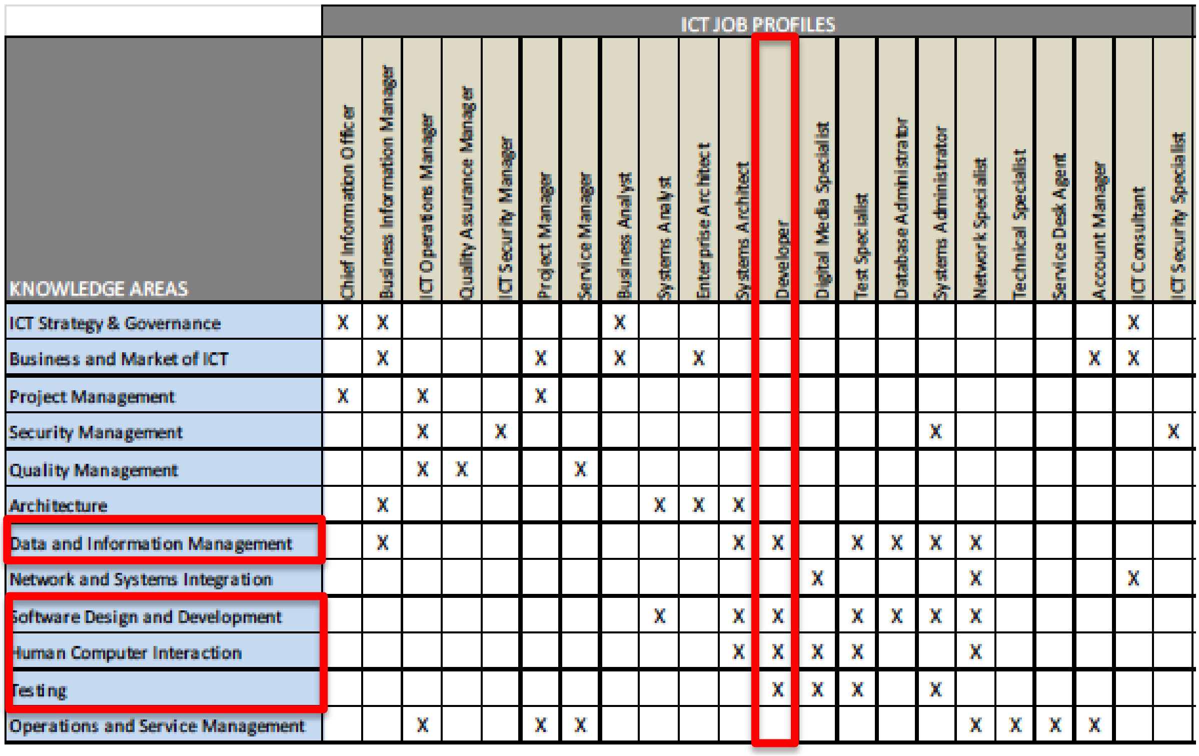
me of them it is almost true, but in most of other Knowledge Areas is not the case, as only a small portion of the Knowledge Area is needed. It will be the task of curriculum development to be more specific and specify lower level knowledge units and skills.



	e-CF																																		
KNOWLEDGE AREAS	IS and Business Strategy Alignment	Service Level Management	Business Plan Development	Product/ Service Planning	Architecture Design	Application Design	Technology Trend Monitoring	Sustainable Development	Innovating	Application Development	Component Integration	Testing	Solution Deployment	System engineering	User support	Change support	Service Delivery	Problem Management	Information security strategy development	ICT quality strategy development	Channel Management	Sales Management	Contract Management	Personnel development	Information and knowledge management	Needs identification	Digital marketing	Project and Portfolio Management	Risk Management	Relationship Management	Process Improvement	ICT Quality Management	Business Change Management	Information Security Management	IS Governance
ICT Strategy & Governance	X		X															X										X	X				X		X
Business and Market of ICT			X				X		X												X	X				X	X						X		
Project Management				X														X										X							
Security Management										X				X		X	X		X															X	
Quality Management		X																		X											X	X			
Architecture	X				X	X				X				X																					
Data and Information Management						X				X				X	X										X										
Network and Systems Integration	X						X			X	X		X	X																					
Software Design and Development						X				X	X		X	X	X																				
Human Computer Interaction					X	X			X	X	X															X									
Testing											X	X	X																					X	
Operations and Service Management		X		X													X	X	X	X			X			X									
Soft Skills																								X						X					
IT Legal, Ethical, Social and Professional practices	X							X															X		X		X								
Disruptive Technologies	X						X		X																										

Figure 1.17: Relationships between Developer’s e-competences and Knowledge Areas of the ICT Foundation Body of Knowledge

More specific, four Knowledge Areas of the profile Developer are shown in Figure 1.18 that shows relationships of the European ICT Professional Profiles and Knowledge Areas of the ICT Foundation Body of Knowledge.



	ICT JOB PROFILES																					
KNOWLEDGE AREAS	Chief Information Officer	Business Information Manager	ICT Operations Manager	Quality Assurance Manager	ICT Security Manager	Project Manager	Service Manager	Business Analyst	Systems Analyst	Enterprise Architect	Systems Architect	Developer	Digital Media Specialist	Test Specialist	Database Administrator	Systems Administrator	Network Specialist	Technical Specialist	Service Desk Agent	Account Manager	ICT Consultant	ICT Security Specialist
ICT Strategy & Governance	X	X						X													X	
Business and Market of ICT		X				X		X		X										X	X	
Project Management	X		X			X																
Security Management			X		X											X						X
Quality Management			X	X			X															
Architecture		X							X	X	X											
Data and Information Management		X									X	X		X	X	X	X					
Network and Systems Integration													X				X				X	
Software Design and Development									X		X	X		X	X	X	X					
Human Computer Interaction											X	X	X	X			X					
Testing												X	X	X		X						
Operations and Service Management			X			X	X										X	X	X	X		

Figure 1.19: Relationships between ICT Job Profiles and Knowledge Areas of the ICT Foundation Body of Knowledge

Unfortunately, the ICT Foundation Body of Knowledge does not provide yet lower levels of knowledge and it is not sufficient for a curriculum development. Therefore, additional extensions (sub-topics) of the Bodies of Knowledge are needed.

1.3.2 The Body of Knowledge for Developer SCHE Programme

IEEE Computer Society specified two Bodies of Knowledge (BOK) that are relevant for ICT Profile Developer:

1. **Computer Science Curricula 2013** - Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, December 20, 2013, The Joint Task Force on Computing Curricula of Association for Computing Machinery (ACM) and IEEE Computer Society
2. **SWEBOK 3**.0 – Guide to the Software Engineering Body of Knowledge, Editors Pierre Bourque, École de technologie supérieure (ÉTS) and Richard E. (Dick) Fairley, Software and Systems Engineering Associates (S2EA), IEEE Computer Society

Knowledge areas and topics from these two Bodies of Knowledge are to be selected according to specified of Knowledge Areas and e-competences required for ICT Profile Developer specified in previous sections.

Figure 1.1 showed European ICT Profile Family Tree with Generation 1 and 2 of ICT Profiles. 23 in total). As this SCHE Programme aims to educate and train Java

Developers, i.e. developers of applications written in Java, we will create a **Generation 3 ICT Profile – Java Developer**. We have to provide all competences specified for ICT Profile Developer specified in previous sections, but extended with specific competences of Java Developers.

2 THE SHORT CYCLE PROGRAMME FOR THE PROFILE ICT JAVA DEVELOPER

2.1 Organisation structure of a Short Cycle Program

In order to develop the required competences of a ICT Profile, such as Developer, a learner must learn all knowledge units (such as topics and sub-topics of a Knowledge Area) specified for the Profile and develop necessary skills. A course is the basic set of knowledge and skills that a student must verify that he or she acquired the specified knowledge and skills by passing an exam. To acquire all competences required, a student must complete a number of courses by passing their exams. The granularity of courses my be different and smaller courses are usually preferable, as student can easier complete their assignment specified by their syllabi and pass their exams.

In some cases courses are inter-related and can be grouped in modules. A short cycle program may have any number of courses and modules. Figure 2.1 shows the general structure of a short cycle program.

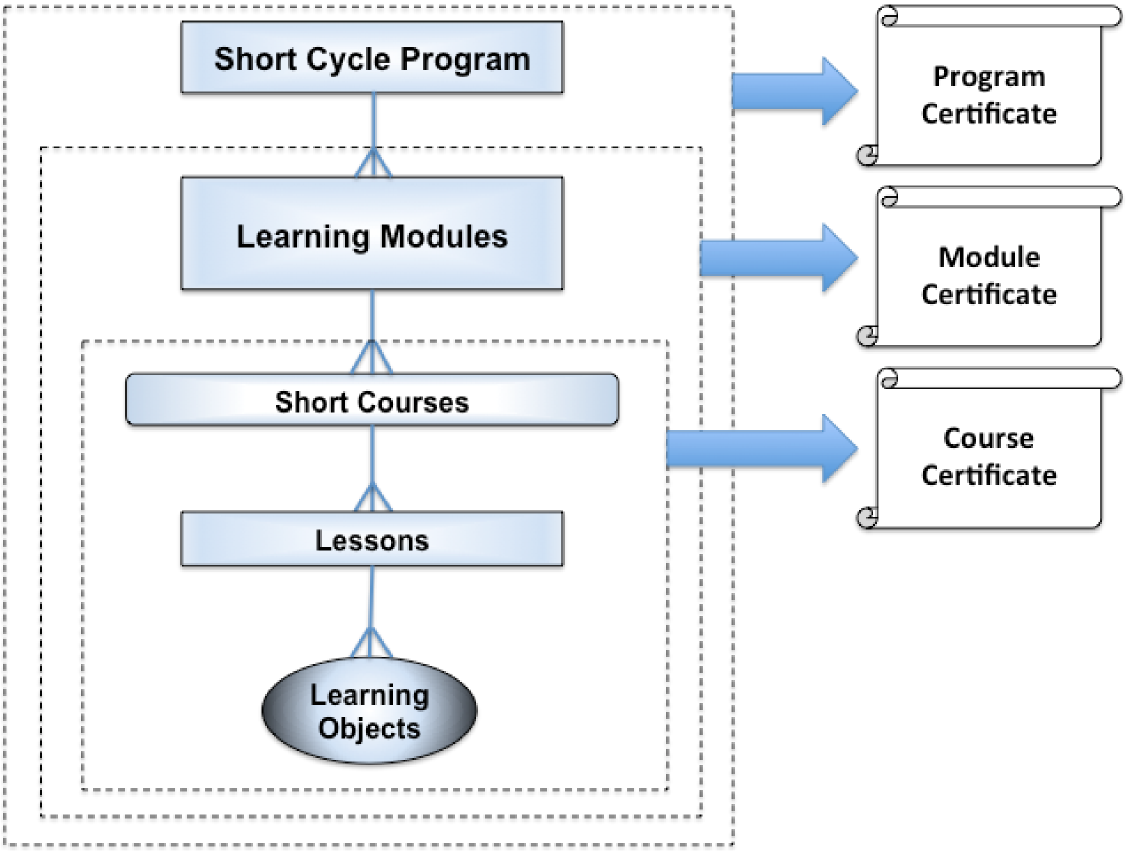


Figure 2.1: A typical organization structure of a Short Cycle Program

A Short Cycle Program must provide students with the required competences and must qualify them for the specified job. In our case here, the job is the job of a **Java Developer**, specified in the previous chapter. The Short Cycle Courses will be defined in groups (Modules) related to the specific e-competences listed for the ICT Profile ***Developer*.** Each Short Course contains a number of Lessons created by Learning Objects (LO). BMU is using LO of fine granularity needed for personalized e-learning (BMU is strategically oriented to develop and implement personalized e-learning). Small size LOs support LO reusability among different courses.

As shown in Figure 2.1, BMU offers three levels of Certificates:

1. *Course Certificate* - for all students that pass the final exams of a course.
2. *Module Certificate* - for all students that pass the final exams of a all course of a Module planned for a SC Program.
3. *Programme Certificate* - for all students that pass the final exams of all modules of a SC course.

If a Short Cycle Programme does no contain modules, it provides only two certificates: *Course Certificate* and *Programme Certificate* (Figure 2.2)

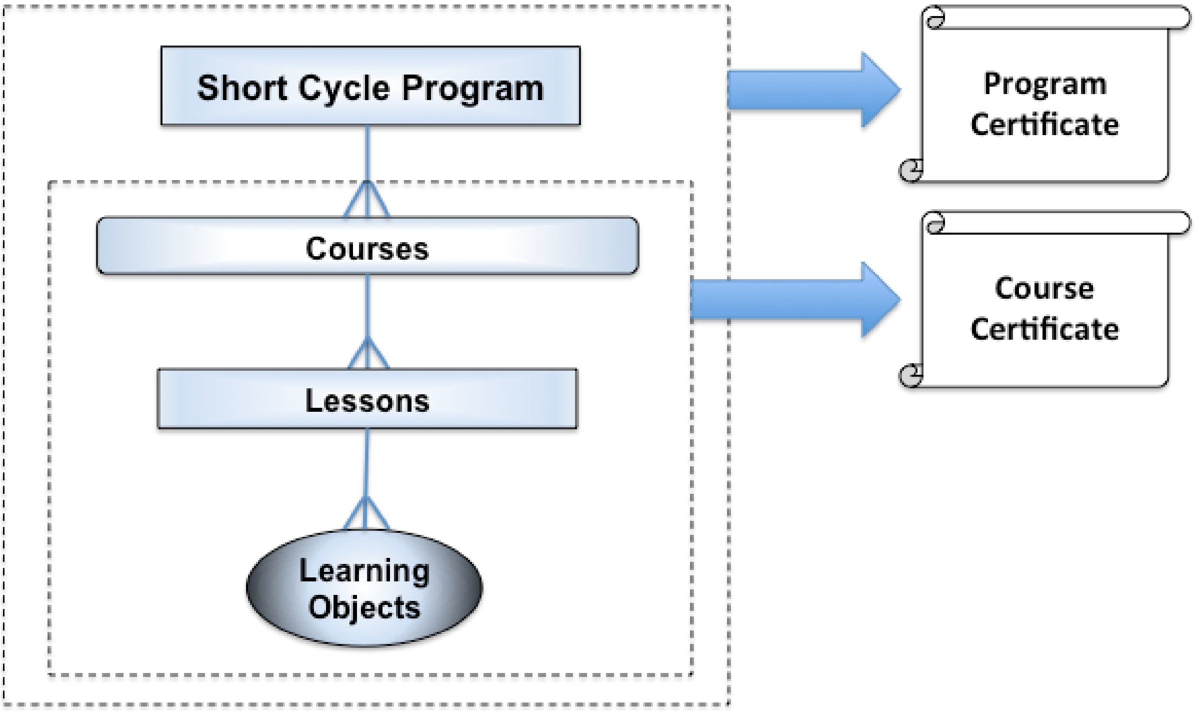
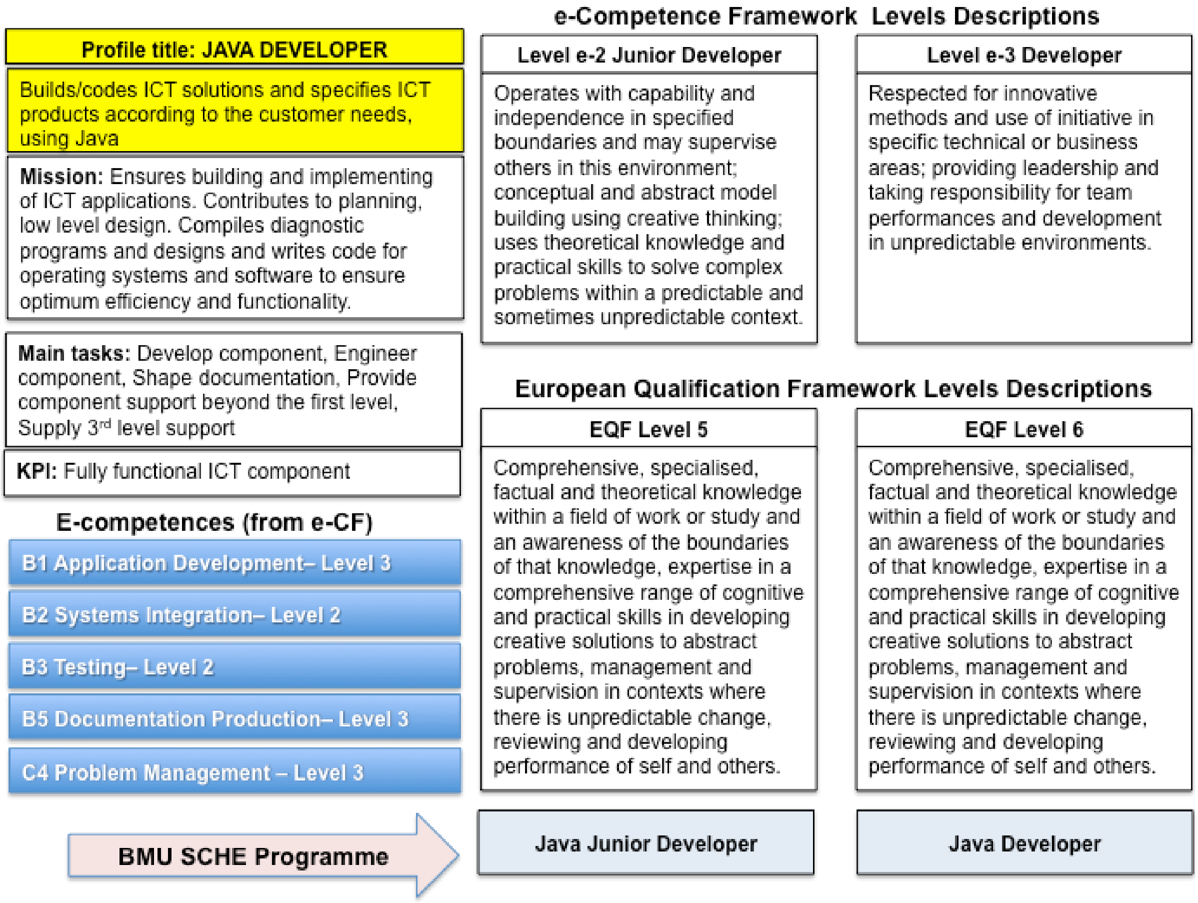


Figure 2.2: A Short-Cycle Programme without modules

Having in mind Figures 1.5 – 1.10, Figure 2.3 was created. It relates e-competence levels (e-2 and e-3) with EQF levels (5 and 6) with five required e-competences (B1, B2, B3, B5 and C4) for a ICT job profile JAVA DEVELOPER, as a 3rd level specialization of ICT job profile DEVELOPER. The difference is that all e-competences must be implemented with Java technology. The job profile short description, mission, and main tasks are the same as for the ICT job profile DEVELOPER.

Depending of the achieved e-competence level (e-2 or e-3) and EQF level (5 or 6), a SCHE program may educate and train a Java Junior Developer or a Java Developer (Figure 2.4). The pilot implementation od the SCHE JAVA DEVELOPER is developed for Java Developer level (e-3 and EQF level 6).



Profile title: JAVA DEVELOPER

Builds/codes ICT solutions and specifies ICT products according to the customer needs, using Java

Mission: Ensures building and implementing of ICT applications. Contributes to planning, low level design. Compiles diagnostic programs and designs and writes code for operating systems and software to ensure optimum efficiency and functionality.

Main tasks: Develop component, Engineer component, Shape documentation, Provide component support beyond the first level, Supply 3rd level support

KPI: Fully functional ICT component

E-competences (from e-CF)

- B1 Application Development– Level 3
- B2 Systems Integration– Level 2
- B3 Testing– Level 2
- B5 Documentation Production– Level 3
- C4 Problem Management – Level 3

e-Competence Framework Levels Descriptions

Level e-2 Junior Developer

Operates with capability and independence in specified boundaries and may supervise others in this environment; conceptual and abstract model building using creative thinking; uses theoretical knowledge and practical skills to solve complex problems within a predictable and sometimes unpredictable context.

Level e-3 Developer

Respected for innovative methods and use of initiative in specific technical or business areas; providing leadership and taking responsibility for team performances and development in unpredictable environments.

European Qualification Framework Levels Descriptions

EQF Level 5

Comprehensive, specialised, factual and theoretical knowledge within a field of work or study and an awareness of the boundaries of that knowledge, expertise in a comprehensive range of cognitive and practical skills in developing creative solutions to abstract problems, management and supervision in contexts where there is unpredictable change, reviewing and developing performance of self and others.

EQF Level 6

Comprehensive, specialised, factual and theoretical knowledge within a field of work or study and an awareness of the boundaries of that knowledge, expertise in a comprehensive range of cognitive and practical skills in developing creative solutions to abstract problems, management and supervision in contexts where there is unpredictable change, reviewing and developing performance of self and others.

BMU SCHE Programme

Java Junior Developer

Java Developer

Figure 2.3: ICT job profile description for Java Junior Developer and Java Developer

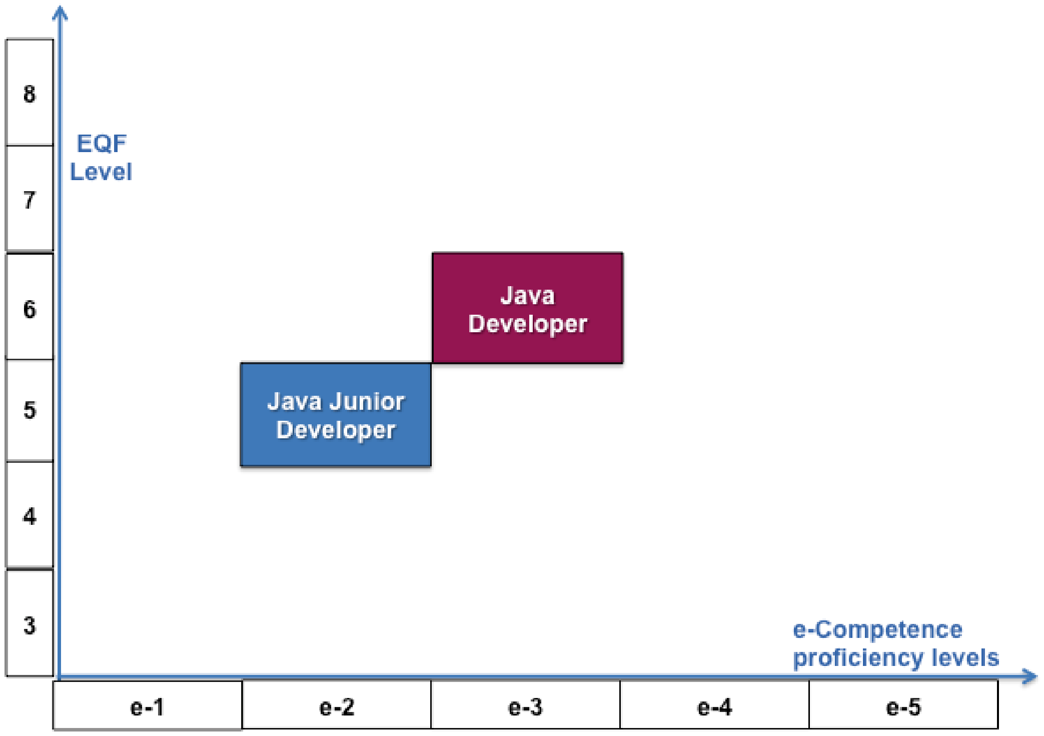


Figure 2.4: Positioning of Java Junior Developer and Java Developer SCHE programs in relation to EQF levels and e-Competence proficiency levels

2.2 Relationships between e-competences and BMU e-courses

At this stage we need to identify the existing BMU e-courses that can be used in Short Cycle HE Program JAVA DEVELOPER (or shorter, SCHE JAVA DEVELOPER) for development of its Courses. It can significantly reduce the effort of developing SCP JAVA DEVELOPER and its courses (Figure 2.5). As BMU bachelor courses are based on SWEBOK, their parts of the Body of Knowledge are to be mapped into BMU SCHE courses

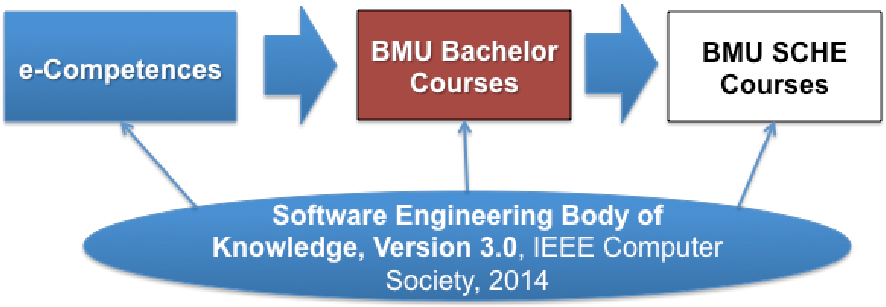
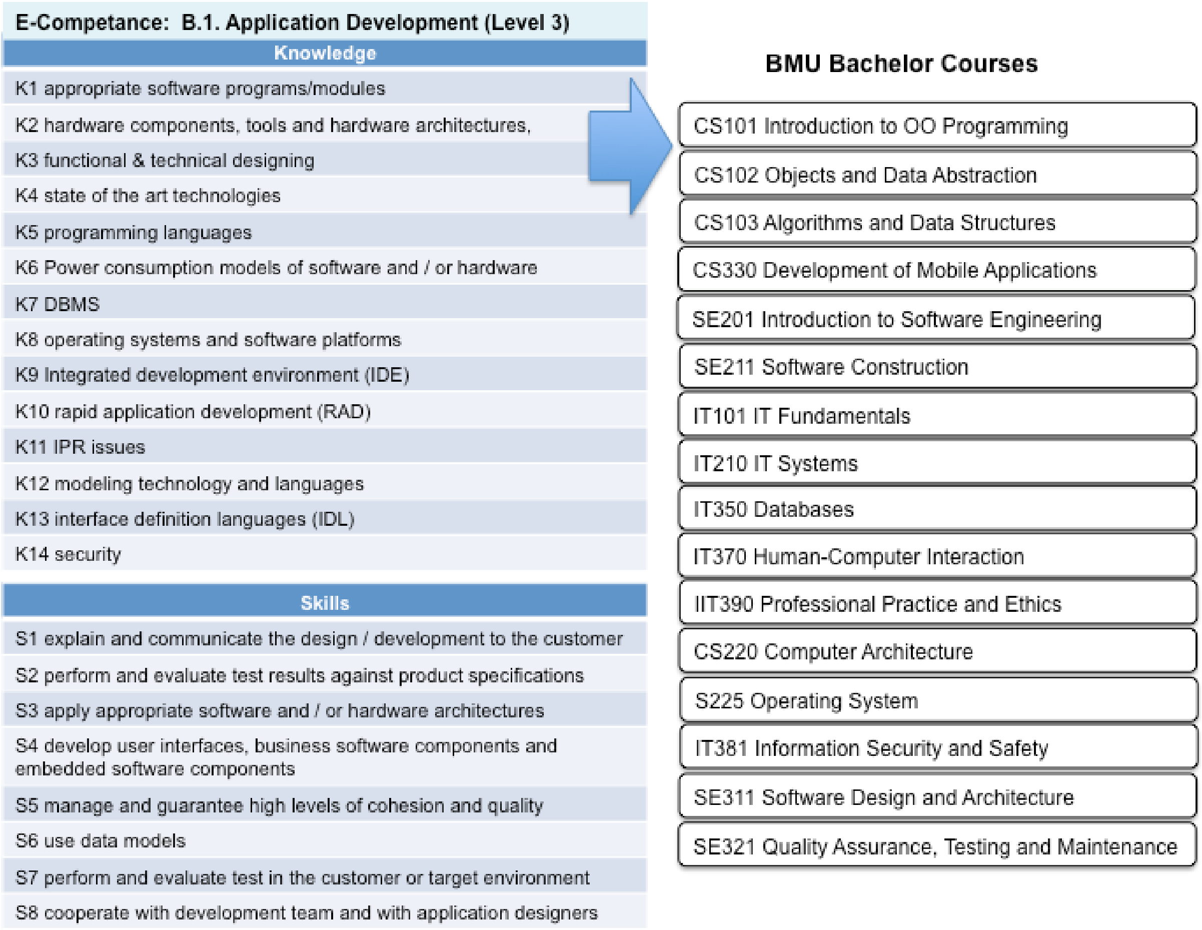

e-Competences
BMU Bachelor Courses
BMU SCHE Courses
Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014

Figure 2.5: Mapping of required e-competences into BMU bachelor courses and courses of the BMU SCHE Java Developer

2.2.1 Acquiring the e-competence B.1. Design and Development (Level 3)

Figure 2.6 shows the list of knowledge areas required for ICT e-competence **B.1. Application Developmen**t, as well as the BMU e-courses that offer learning objects (learning contents) corresponding to these knowledge areas. Using the Software Engineering Body of Knowledge (SWEBOK 3.0) we will specify all needed learning units that constitute each of the listed learning areas. The listed BMU e-courses were developed to implement SWEBOK 3.0 , they provide learning objects for all knowledge units that are part of SWEBOK 3.0 Knowledge Areas.



E-Competance: B.1. Application Development (Level 3)
Knowledge
K1 appropriate software programs/modules
K2 hardware components, tools and hardware architectures,
K3 functional & technical designing
K4 state of the art technologies
K5 programming languages
K6 Power consumption models of software and / or hardware
K7 DBMS
K8 operating systems and software platforms
K9 Integrated development environment (IDE)
K10 rapid application development (RAD)
K11 IPR issues
K12 modeling technology and languages
K13 interface definition languages (IDL)
K14 security
Skills
S1 explain and communicate the design / development to the customer
S2 perform and evaluate test results against product specifications
S3 apply appropriate software and / or hardware architectures
S4 develop user interfaces, business software components and embedded software components
S5 manage and guarantee high levels of cohesion and quality
S6 use data models
S7 perform and evaluate test in the customer or target environment
S8 cooperate with development team and with application designers

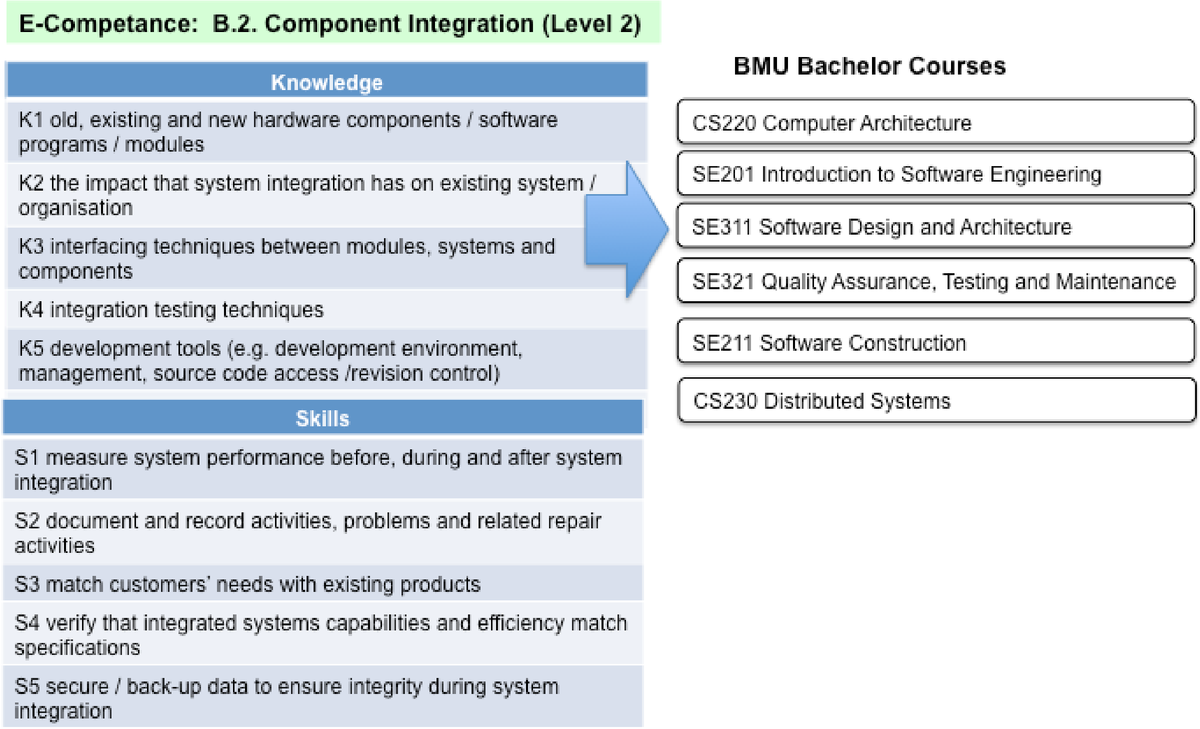
BMU Bachelor Courses

- CS101 Introduction to OO Programming
- CS102 Objects and Data Abstraction
- CS103 Algorithms and Data Structures
- CS330 Development of Mobile Applications
- SE201 Introduction to Software Engineering
- SE211 Software Construction
- IT101 IT Fundamentals
- IT210 IT Systems
- IT350 Databases
- IT370 Human-Computer Interaction
- IIT390 Professional Practice and Ethics
- CS220 Computer Architecture
- S225 Operating System
- IT381 Information Security and Safety
- SE311 Software Design and Architecture
- SE321 Quality Assurance, Testing and Maintenance

Figure 2.6: Knowledge areas of e-competence B.1. and related BMU e-courses

2.2.2 Acquiring the e-competence B.2. System Integration (Level 2)

Figure 2.7 shows the knowledge areas required for the **B.2. System Integration** e-competence and the BMU e-courses that provide learning objects corresponding to the learning units of the listed knowledge areas. These learning units are specified in the SWEBOK 3.0 (specified by IEEE Computer Society and AIS) for each learning area.



E-Competance: B.2. Component Integration (Level 2)

Knowledge

K1 old, existing and new hardware components / software programs / modules

K2 the impact that system integration has on existing system / organisation

K3 interfacing techniques between modules, systems and components

K4 integration testing techniques

K5 development tools (e.g. development environment, management, source code access /revision control)

Skills

S1 measure system performance before, during and after system integration

S2 document and record activities, problems and related repair activities

S3 match customers' needs with existing products

S4 verify that integrated systems capabilities and efficiency match specifications

S5 secure / back-up data to ensure integrity during system integration

BMU Bachelor Courses

CS220 Computer Architecture

SE201 Introduction to Software Engineering

SE311 Software Design and Architecture

SE321 Quality Assurance, Testing and Maintenance

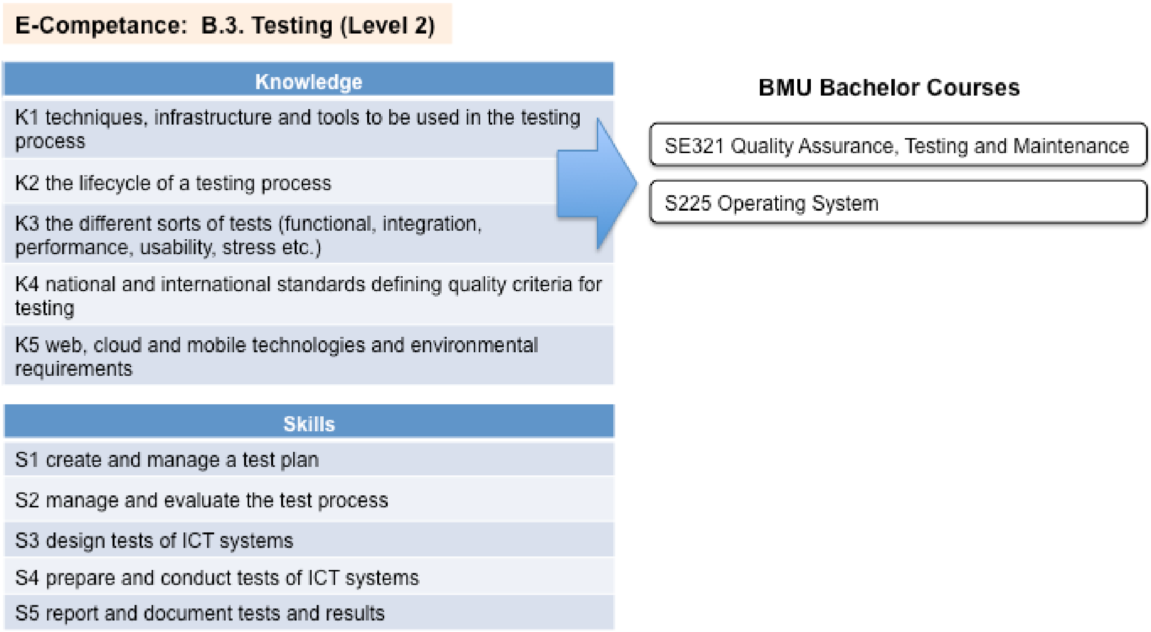
SE211 Software Construction

CS230 Distributed Systems

Figure 2.7: The knowledge areas specified for the e-competence B.2. Component Integration and related BMU e-courses.

2.2.3 Acquiring the e-competence B.3.Testing (Level 2)

Figure 2.8 shows the knowledge areas required for the **B.3. Testing** e-competence and the BMU e-courses that provide learning objects corresponding to the learning units of the listed knowledge areas. These learning units are specified in the SWEBOK 3.0 (specified by IEEE Computer Society and AIS) for each learning area.



E-Competance: B.3. Testing (Level 2)

Knowledge
K1 techniques, infrastructure and tools to be used in the testing process
K2 the lifecycle of a testing process
K3 the different sorts of tests (functional, integration, performance, usability, stress etc.)
K4 national and international standards defining quality criteria for testing
K5 web, cloud and mobile technologies and environmental requirements

Skills
S1 create and manage a test plan
S2 manage and evaluate the test process
S3 design tests of ICT systems
S4 prepare and conduct tests of ICT systems
S5 report and document tests and results

BMU Bachelor Courses

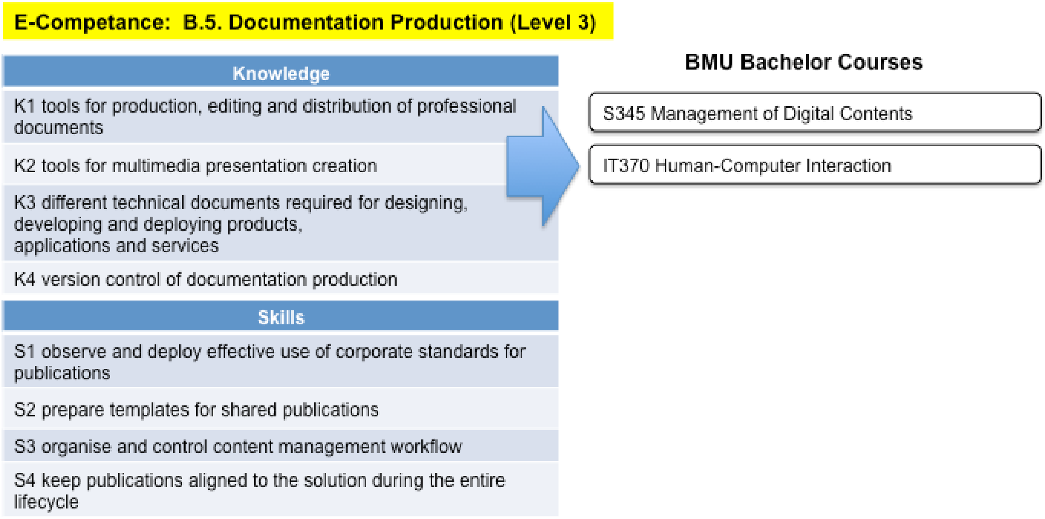
SE321 Quality Assurance, Testing and Maintenance

S225 Operating System

Figure 2.8: The knowledge areas specified for the e-competence B.3. Testing and related BMU e-courses.

2.2.4 Acquiring the e-competence B.5. Documentation Production (Level 3)

Figure 2.9 shows the knowledge areas required for the **B.5. Documentation Production** e-competence and the BMU e-courses that provide learning objects corresponding to the learning units of the listed knowledge areas. These learning units are specified in the SWEBOK 3.0 (for each learning area.



E-Competance: B.5. Documentation Production (Level 3)

Knowledge
K1 tools for production, editing and distribution of professional documents
K2 tools for multimedia presentation creation
K3 different technical documents required for designing, developing and deploying products, applications and services
K4 version control of documentation production

Skills
S1 observe and deploy effective use of corporate standards for publications
S2 prepare templates for shared publications
S3 organise and control content management workflow
S4 keep publications aligned to the solution during the entire lifecycle

BMU Bachelor Courses

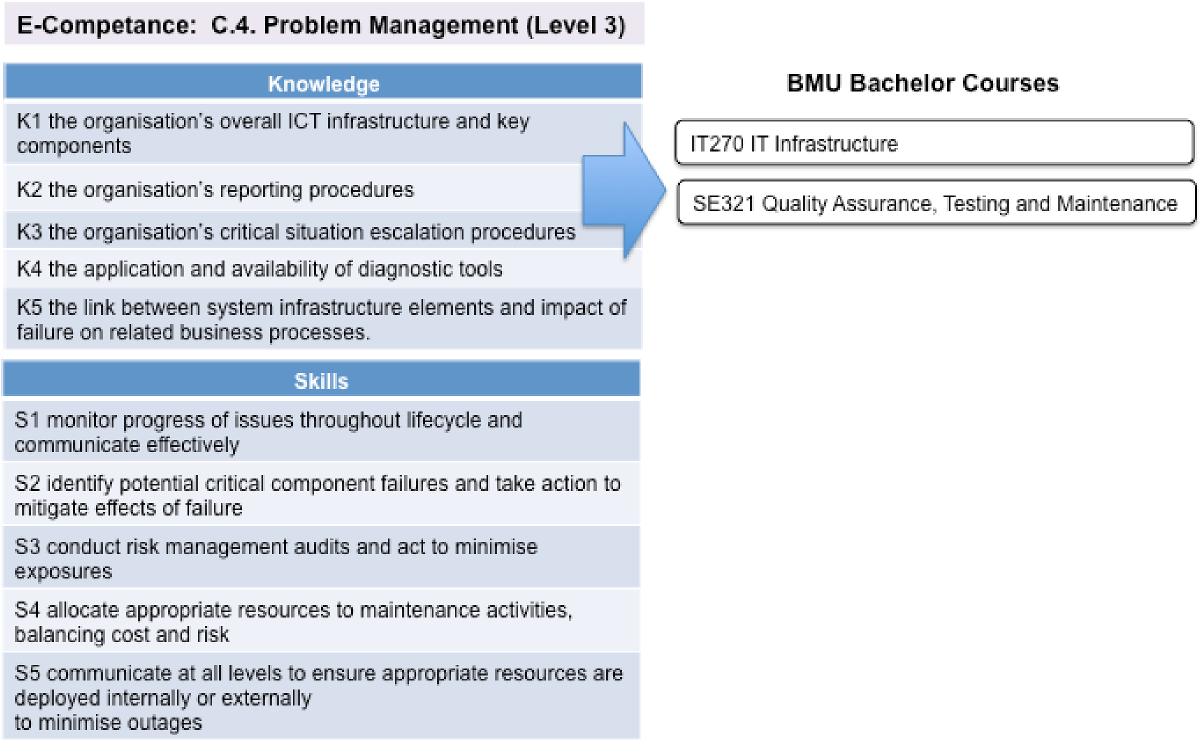
S345 Management of Digital Contents

IT370 Human-Computer Interaction

Figure 2.9: The knowledge areas specified for the e-competence B.5. Documentation Production and related BMU e-courses.

2.2.5 Acquiring the e-competence C.4. Problem Management (Level 3)

Figure 2.10 shows the knowledge areas required for the **C.4. Problem Management** e-competence and the BMU e-courses that provide learning objects corresponding to the learning units of the listed knowledge areas. These learning units are specified in the SWEBOK 3.0 (for each learning area.



E-Competance: C.4. Problem Management (Level 3)
Knowledge
K1 the organisation's overall ICT infrastructure and key components
K2 the organisation's reporting procedures
K3 the organisation's critical situation escalation procedures
K4 the application and availability of diagnostic tools
K5 the link between system infrastructure elements and impact of failure on related business processes.
Skills
S1 monitor progress of issues throughout lifecycle and communicate effectively
S2 identify potential critical component failures and take action to mitigate effects of failure
S3 conduct risk management audits and act to minimise exposures
S4 allocate appropriate resources to maintenance activities, balancing cost and risk
S5 communicate at all levels to ensure appropriate resources are deployed internally or externally to minimise outages

BMU Bachelor Courses

- IT270 IT Infrastructure
- SE321 Quality Assurance, Testing and Maintenance

Figure 2.10: The knowledge areas specified for the e-competence **C.4. Problem Management** and related BMU e-courses.

2.2.6 The List of BMU e-Courses Related to c-competences Specified for the ICT Profile Developer

After analyzing Figures 2.6 -2.10, Figure 2.11 was created showing the BMU e-courses corresponding to all five e-competence specified for the ICT Profile **Java Developer**.

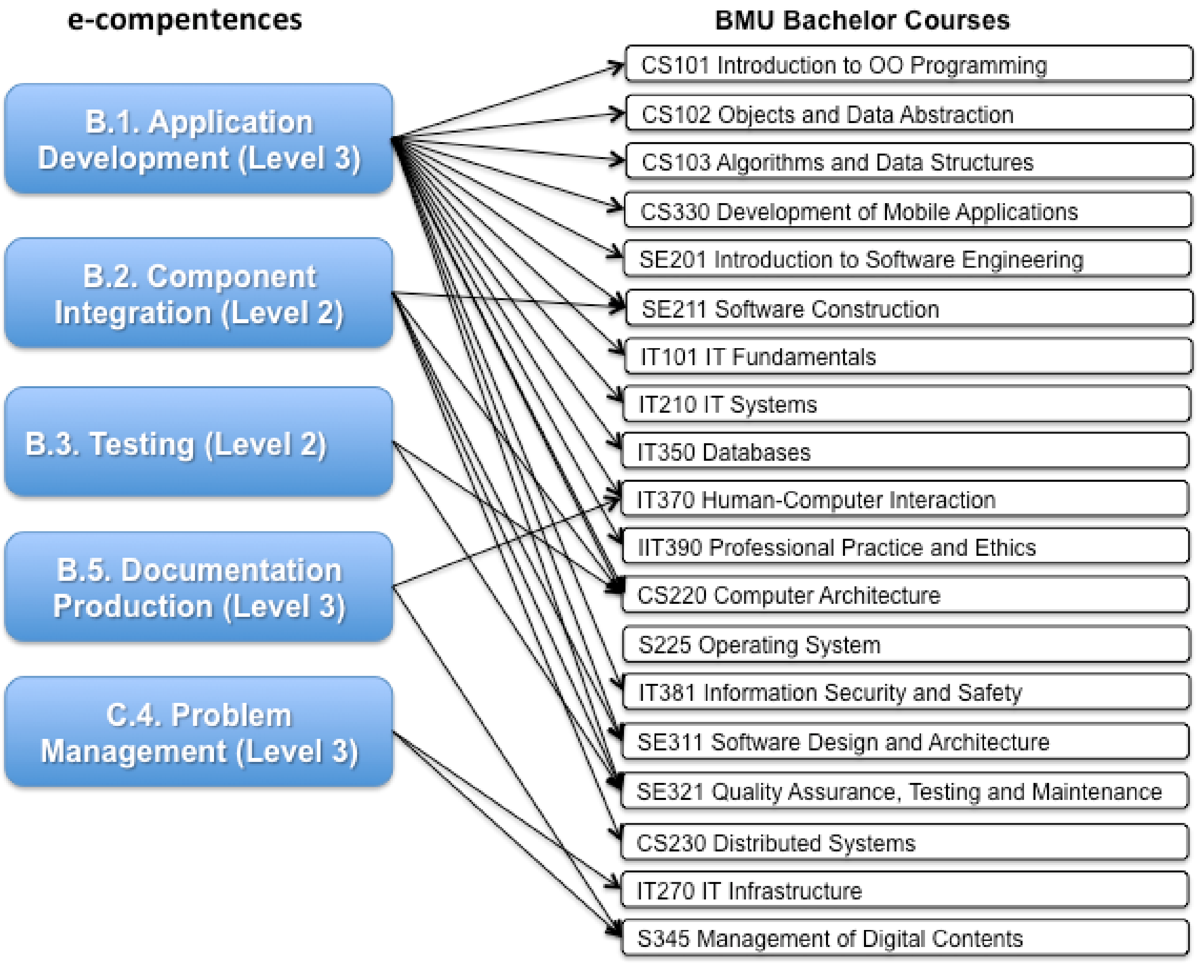

e-compentences
BMU Bachelor Courses
B.1. Application Development (Level 3)
B.2. Component Integration (Level 2)
B.3. Testing (Level 2)
B.5. Documentation Production (Level 3)
C.4. Problem Management (Level 3)
CS101 Introduction to OO Programming
CS102 Objects and Data Abstraction
CS103 Algorithms and Data Structures
CS330 Development of Mobile Applications
SE201 Introduction to Software Engineering
SE211 Software Construction
IT101 IT Fundamentals
IT210 IT Systems
IT350 Databases
IT370 Human-Computer Interaction
IIT390 Professional Practice and Ethics
CS220 Computer Architecture
S225 Operating System
IT381 Information Security and Safety
SE311 Software Design and Architecture
SE321 Quality Assurance, Testing and Maintenance
CS230 Distributed Systems
IT270 IT Infrastructure
S345 Management of Digital Contents

Figure 2.8: The BMU e-courses related to five e-competences specified for the ICT Profile Java Developer

2.2.7 Mapping of BMU Bachelor Courses into SCHE Java Developer Courses

Next step in development process of SCHE Java Developer courses if mapping of BMU e-courses into SCHE Java Developer e-courses (Figure 2.9).

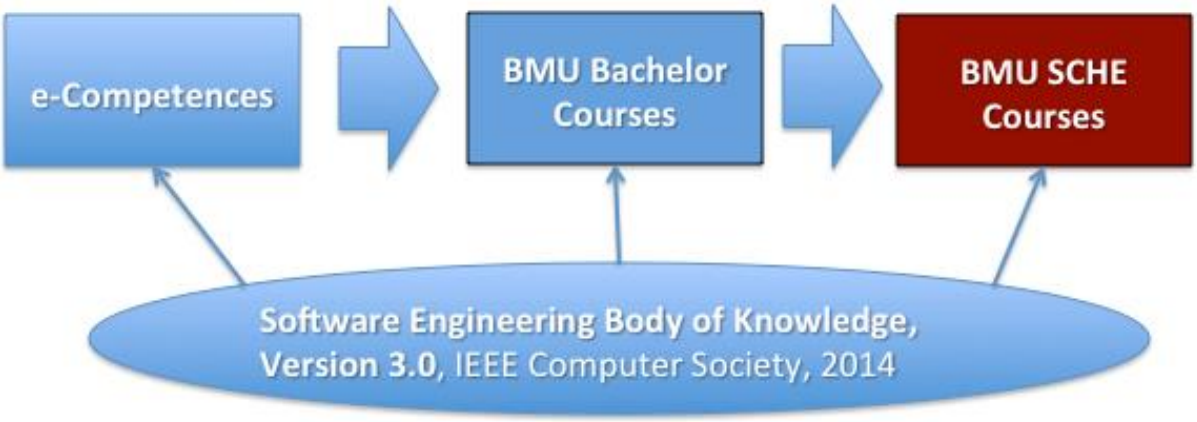

e-Competences
BMU Bachelor Courses
BMU SCHE Courses
Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014

Figure 2.9: Mapping of BMU bachelor courses into SCHE Java Developer courses

Figure 2.10 shows created SCHE Java Developer courses. These courses takes into account specifics of SCHE Java Developer. They have to provide more practical and simpler explanation of programming concepts, more elaborated shown examples, and many assignments for individual exercise of each student. In the next chapter, syllabi of these courses will be specified.

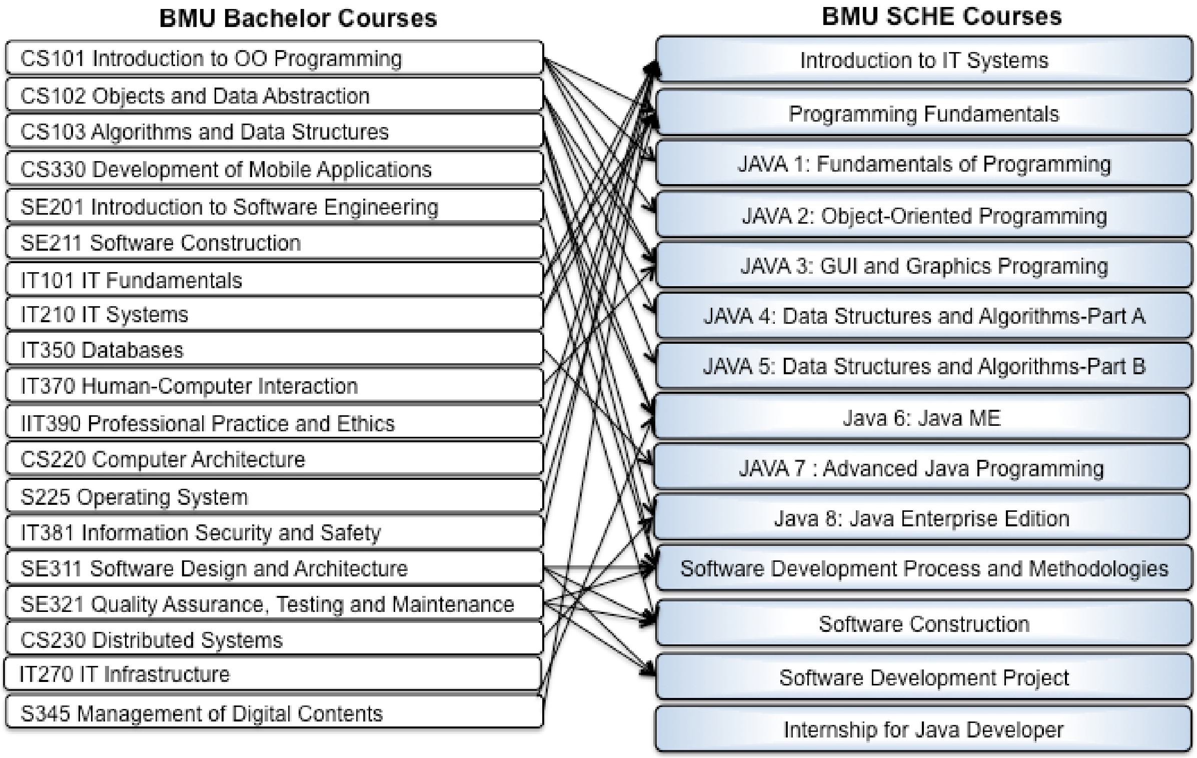

BMU Bachelor Courses
CS101 Introduction to OO Programming
CS102 Objects and Data Abstraction
CS103 Algorithms and Data Structures
CS330 Development of Mobile Applications
SE201 Introduction to Software Engineering
SE211 Software Construction
IT101 IT Fundamentals
IT210 IT Systems
IT350 Databases
IT370 Human-Computer Interaction
IIT390 Professional Practice and Ethics
CS220 Computer Architecture
S225 Operating System
IT381 Information Security and Safety
SE311 Software Design and Architecture
SE321 Quality Assurance, Testing and Maintenance
CS230 Distributed Systems
IT270 IT Infrastructure
S345 Management of Digital Contents
BMU SCHE Courses
Introduction to IT Systems
Programming Fundamentals
JAVA 1: Fundamentals of Programming
JAVA 2: Object-Oriented Programming
JAVA 3: GUI and Graphics Programing
JAVA 4: Data Structures and Algorithms-Part A
JAVA 5: Data Structures and Algorithms-Part B
Java 6: Java ME
JAVA 7 : Advanced Java Programming
Java 8: Java Enterprise Edition
Software Development Process and Methodologies
Software Construction
Software Development Project
Internship for Java Developer

Figure 2.10 Created SCHE Java Developer courses

3 COURSES OF SCP JAVA DEVELOPER

3.1 Sequence of courses of SCHE Java Developer

The following table shows all courses and their planned sequence.

#	Course	Starting Date 1	Exam Date 1
1	Introduction to IT systems	02.10.2017	18.10.2017
2	Programming Fundamentals	23.10.2017	03.11.2018
3	Java 1: Fundamentals of Programming	07.11.2017	24.11.2017
4	Java 2: Object-oriented programming	27.11.2017	11.12.2017
5	Java 3: GUI Programming	12.12.2017	30.12.2017
6	Java 4: Data Structures and Algorithms – Part A	08.01.2018	25.01.2018
7	Java 5: Data Structures and Algorithms – Part B	29.01.2018	15.02.2018
8	Java 6: Java ME	19.02.3018	07.03.2018
9	Java 7: Advanced Java Programming	125.3.2018	28.03.2018
10	Java 8: Java Enterprise Edition	02.04.2018	21.04.2018
11	Software Development Process and Methodologies	30.04.2018	19.05.2018
12	Software Construction	21.05.2018	13.06.2018
13	Software Development Project	18.06.2018	05.07.2918
14	Intership (8 weeks)	06.08.2018	29.09.2018

The following section specifies syllabi of these courses.

3.2 Syllabi of Programming Module Courses

The Programming Module provides the following SC Courses:

1. Introduction of IT Systems
2. Programming Fundamentals
3. JAVA 1: Fundamentals of Programming
4. JAVA 2: Object-Oriented Programming
5. JAVA 3: GUI Programming
6. JAVA 4: Data Structures and Algorithms – Part A
7. JAVA 5: Data Structures and Algorithms – Part A
8. JAVA 6: JAVA ME
9. JAVA 7: Advanced Java programming
10. JAVA 8: Java Enterprise Edition
11. Software Development Process and Methodologies
12. Software Construction
13. Software Development Project
14. Internship (8 weeks)

3.2.1 Course 1: Introduction to IT Systems

Duration: 15 days, 12 online teaching days, 2 day workshop days

Number of hours: 3 hours per online/workshop day, Total: 42 hours

ECTS: 4

Day	Hours	Teaching units	Topics	Results – knowledge or skills that the students should receive
1	3	Model of IT Systems	Components of computer systems Computer system System software Operating system Utilities Application software Computer Hardware Central processing unit Input / output devices Memory Data and information Input and output devices	
2	3	Operating Systems	Overview of the operating system functions Operating system roles Types of operating systems and their characteristics Operating systems of personal computers Operating systems server Real-time operating systems Mainframe operating systems File system Comparison of Windows and OS Unis	
3	3	Concepts and Fundamentals of Information Management Architecture of Data Organisation	Information systems: purpose, use, value Characteristics of data (quality, accuracy, changes with time) Challenges in data management Life cycle of data Database systems Knowledge management Data models Relational model Normal forms Functional dependencies 1NF, 2NF, 3NF	

4	3	Data Modelling	Conceptual model Entity Relationship Diagrams Logical models Physical models Standardized modeling in IDEF1 and UML	
		DDL i basic form of statement SELECT	DDL: CREATE TABLE, CREATE INDEX; ALTER TABLE, DROP TABLE; Commands CREATE TABLE, CREATE INDEX; ALTER TABLE, DROP TABLE; Commands: INSERT, UPDATE, DELETE Examples of DDL commands for creating database elements Examples of applying the basic form of the SELECT command to display the unchanged table contents DMS: INSERT, UPDATE, DELETE Queries over one table showing the unchanged content of the table: SELECT ... FROM;	
5 6	6	Web Technologies Development of Web Sites Architecture of Information Digital Media	Preged web technology: HTTP Protocol, HTML / XHTML XML Web interface Availability issue Web Accessibility Initiative Web services Hypertext / hypermedia: Effective Communication, Interfaces, Navigation Schemes, Media Types Web design process: Design by user, Web design templates, Organization of information Digital libraries Media formats Tools for recording, creating and producing Compression Broadcast media (Streaming media) Implementation and integration Integration with the database	
7	3	Inter-	Architecture for System	

		Systems Communication	Integration DCOM, CORBA, RMI Web Services and Middleware Network programming Messaging and routing services Data transfer to lower.	
8	3	Mapping and Exchange of Data	Meta data Presentation and encoding of data XML, DTD, XML Schema XML document parsing XSL, XSLT and Xpath Client-server programming	
9	3	Integrative Coding Scripting Technics Techics of Code Writing Integrations	IPT3. Integrated coding: MVC, singleton, factory method, façade, proxy, decorator and observer Writing a script and the role of a scripting language Comparative presentation of Adopt and Adapt techniques compared to make Versions and version management Components, interfaces and integration Infrastructure, middleware and platforms	
10 11	6	**(HCI)Human-Computer Interacion:** Human Factors Ascpects of HCI of Application Domains Human-Centered Evaluation Development of effective interfaces	Cognitive principles - perception, memory, problem solving Understanding the users Design for man Ergonomics Types of environment Cognitive models Approach Usability testing Usability standards User experience Interaction styles Matching interface elements to user requirements Biometrics The stress syndrome caused by repetition of the same operations PHP language. Writing, analysis and testing a script that includes	

			selection, repetition, and forwarding Create a PHP document for your purpose	
12	3	Basics of Computer Networks Routing Physical Layer	KStandardization bodies OSI model Internet model Nodes and connections IEEE 802.1 Routing algorithms Routing protocols Wireless and mobile connections Commuted and packet transfer Physical media Satellite communications Shannon's law Multimedia technologies WWW Databases and file servers	
13 14	6	**Information Security and Safty:** Fundamental Aspects Security Mechanisms Ataks Security Domains Forensics Information States Model of Risk Analysis Security Services	History and terminology Security way of thinking Model for information security (threats, vulnerability, attacks, countermeasures) Cryptography and cryptosystems Types of attack Security domains Give an overview of possible attacks on network and computer resources Legal system Digital investigation and its relationship with other investigations Rules of record Media analysis Searching and seizing the device Transfer Storage Processing Risk assessment Costs Availability Integrity Secrecy Authentication Non-repudiation	
15	3	**Final**	Students get examination questions	To evaluate knowledge and

		examination (in BMU computer rooms)	and problems Exam duration - 3 hours	skills acquired during the course

3.2.2 Course 2: Programming Fundamentals

Duration: 11 days, 8 online teaching days, 2 day workshop days

Number of hours: 3 hours per online/workshop day, Total: 30 hours

ECTS: 3

Day	Hours	Teaching units	Topics	Results – knowledge or skills that the students should receive
1,2	6	Problem Solving Techniques Programming Fundamentals	What Is a computer? Definition of Problem Solving Formulating the Real Problem Analyze the Problem Design a Solution Search Strategy Problem Solving Using Programs The Programming Process Programming Paradigms	To formulate and analyse programmimg problems To design a solution search strategy To understand the programming process To understand programming paradigmes
2,3	6	Programming Language Basics	Programming Language Overview Operating Systems Syntax and Semantics of Programming Languages Low-Level Programming Languages High-Level Programming Languages Declarative vs. Imperative Programming Languages	To understane the role of operating systems To difirentiate the syntax and semantics of programming languages To understabd the difference between low- and high-level languages To understand the difference between declarative and imperative programming languages
4,5	6	Introduction of algorithms and problem-solving	Problem-solving strategies; the role of algorithms in the problem-solving process; implementation strategies for algorithms; the concept and properties of algorithms	To understabd the roel of algorithms To implement alogoritmes in porgramming To understand the concept and properties of algorithms
5,6	6	Implementation of algorithms	Examples of algoritmic problem-solving processes Exercises and student assignments	To implement algorithms in solving different problems
7	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course

8	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students’ project. Presentation of the project report
13	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.3 Course 3: JAVA 1: Fundamentals of Programming

Duration: 17 days, 14 online teaching days, 2 day workshop days

Number of hours: 3 hours per online/workshop day, Total: 48 hours

ECTS: 4

Day	Ho-urs	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1	3	**Introduction to Java**	What is Java? Specification, API, JDK, and iDE A simple Java program Creating, compiling, and executing a java program Programming style and documentation Programming errors Developing java programs using NetBeans Programming exercises Programming assignment	To understand computer basics, programs, and operating systems To describe the relationship between Java and the World Wide Web To understand the meaning of Java language specification, API, JDK, and IDE To write a simple Java program To display output on the console To explain the basic syntax of a Java program To create, compile, and run Java programs To use sound Java programming style and document programs properly To explain the differences between syntax errors, runtime errors, and logic errors To develop Java programs using NetBeans
2,3	6	**Elementary programming in Java**	Writing a simple program Reading input from the console Identifiers Variables Assignment statements and assignment expressions Named constants Naming conventions Numeric data types and operations Numeric literals Evaluating expressions and operator precedence Case study: displaying the current time Augmented assignment operators Increment and decrement	To write Java programs to perform simple computations To obtain input from the console using the Scanner class To use identifiers to name variables, constants, methods, and classes To use variables to store data To program with assignment statements and assignment expressions To use constants to store permanent data To name classes, methods, variables, and constants by following their naming conventions To explore Java numeric primitive data types: byte, short, int, long, float, and double To read a byte, short, int, long, float, or double value from the keyboard To perform operations using operators +, -, *, /, and % To perform exponent operations using

			operators Numeric type conversions Software development process Case study: counting monetary units Common errors and pitfalls Programming exercises Programming assignment	Math.pow(a, b) To write integer literals, floating-point literals, and literals in scientific notation (To write and evaluate numeric expressions To obtain the current system time using System.currentTimeMillis() To use augmented assignment operators To distinguish between postincrement and preincrement and between postdecrement and predecrement To cast the value of one type to another type To describe the software development process and apply it to develop the loan payment program To write a program that converts a large amount of money into smaller units To avoid common errors and pitfalls in elementary programming
4,5	6	**Selections (program branching)**	Boolean data type If statements Two-way if-else statements Nested if and multi-way if-else statements Common errors and pitfalls Generating random numbers Case study: computing body mass index Case study: computing taxes Logical operators Case study: determining leap year Case study: lottery Switch statements Conditional expressions Operator precedence and associativity Debugging Programming exercises Programming assignment	To declare boolean variables and write Boolean expressions using relational operators To implement selection control using one-way if statements To implement selection control using two-way if-else statements To implement selection control using nested if and multi-way if statements To avoid common errors and pitfalls in if statements To generate random numbers using the Math.random() method To program using selection statements for a variety of examples (SubtractionQuiz, BMI, ComputeTax) To combine conditions using logical operators (!, &&, ||, and ^) To program using selection statements with combined conditions (LeapYear, Lottery) To implement selection control using switch statements To write expressions using the conditional expression To examine the rules governing operator precedence and associativity To apply common techniques to debug

				errors
6,7	6	**Loops**	The while loop The do-while loop The for loop Which loop to use? Nested loops Minimizing numeric errors Case studies Keywords break and continue Case study: checking palindromes Case study: displaying prime numbers Programming exercises Programming assignment	To write programs for executing statements repeatedly using a while loop To follow the loop design strategy to develop loops To control a loop with a sentinel value To obtain large input from a file using input redirection rather than typing from the keyboard To write loops using do-while statements To write loops using for statements To discover the similarities and differences of three types of loop statements To write nested loops To learn the techniques for minimizing numerical errors To learn loops from a variety of examples (GCD, FutureTuition, Dec2Hex) To implement program control with break and continue To process characters in a string using a loop in a case study for checking palindrome To write a program that displays prime numbers
8,9	6	**Mathematical functions, characters and strings**	Common mathematical functions Character data type and operations The string type Case studies Formatting console output Programming exercises Programming assignment	To solve mathematical problems by using the methods in the Math class To represent characters using the char type To encode characters using ASCII and Unicode To represent special characters using the escape sequences To cast a numeric value to a character and cast a character to an integer To compare and test characters using the static methods in the Character class. To introduce objects and instance methods To represent strings using the String object To return the string length using the length() method To return a character in the string using the charAt(i) method To use the + operator to concatenate

				strings To return an uppercase string or a lowercase string and to trim a string To read strings from the console To read a character from the console To compare strings using the equals method and the compareTo methods To obtain substrings To find a character or a substring in a string using the indexOf method To program using characters and strings (GuessBirthday) To convert a hexadecimal character to a decimal value (HexDigit2Dec) To revise the lottery program using strings (LotteryUsingStrings) To format output using the System.out.printf method
10 11	6	**Methods**	Defining a method Calling a method void method example Passing arguments by values Modularizing code Case study: converting hexadecimals to decimals Overloading methods The scope of variables Case study: generating random characters Method abstraction and stepwise refinement Programming exercises Programming assignment	To define methods with formal parameters To invoke methods with actual parameters (i.e., arguments) To define methods with a return value To define methods without a return value To pass arguments by value To develop reusable code that is modular, easy to read, easy to debug, and easy to maintain To write a method that converts hexadecimals to decimals To use method overloading and understand ambiguous overloading To determine the scope of variables To apply the concept of method abstraction in software development To design and implement methods using stepwise refinement
12 13	6	**Single-Dimensional Arrays**	Array basics Case study: analyzing numbers Case study: deck of cards Copying arrays Passing arrays to methods Returning an array from a method	To describe why arrays are necessary in programming To declare array reference variables and create arrays To obtain array size using arrayRefVar.length and know default values in an array To access array elements using indexes To declare, create, and initialize an array using an array initializer

			Case study: counting the occurrences of each letter Variable-length argument lists Searching arrays Sorting arrays The arrays class Command-line arguments Programming exercises Programming assignment	To program common array operations (displaying arrays, summing all elements, finding the minimum and maximum elements, random shuffling, and shifting elements) To simplify programming using the for each loops To apply arrays in application development (AnalyzeNumbers, DeckOfCards) To copy contents from one array to another To develop and invoke methods with array arguments and return valueTo define a method with a variable-length argument list To search elements using the linear or binary search algorithm. To sort an array using the selection sort approach To use the methods in the java.util.Arrays class To pass arguments to the main method from the command line
14	3	**Multi-Dimensional Arrays**	Two-dimensional array basics Processing two-dimensional arrays Passing two-dimensional arrays to methods Case study: grading a multiple-choice test Case study: finding the closest pair Case study: sudoku Multidimensional arrays Programming exercises Programming assignment	To give examples of representing data using two-dimensional arrays To declare variables for two-dimensional arrays, create arrays, and access array elements in a two-dimensional array using row and column indexes To program common operations for two-dimensional arrays (displaying arrays, summing all elements, finding the minimum and maximum elements, and random shuffling) To pass two-dimensional arrays to methods To write a program for grading multiple-choice questions using twodimensional arrays To solve the closest-pair problem using two-dimensional arrays To check a Sudoku solution using two-dimensional arrays To use multidimensional arrays
15	3	**F2F Project Workshop** (in BMU computer	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during

		rooms, optionally - online)		the course
16	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report
17	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.4 Course 4: Java 2: Object-oriented programming

Duration: 13 days, 10 online teaching days, 2 day workshop days

Number of hours: 3 hours per online/workshop day, Total: 36 hours

ECTS: 3

Day	Ho-urs	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1,2	6	**Classes and objects**	Defining classes for objects Example: defining classes and creating objects Constructing objects using constructors Accessing objects via reference variables Using classes from the java library Static variables, constants, and methods Visibility modifiers Data field encapsulation Passing objects to methods Array of objects Immutable objects and classes The scope of variables The this reference Programming exercises Programming assignment	To describe objects and classes, and use classes to model objects To use UML graphical notation to describe classes and objects To demonstrate how to define classes and create objects To create objects using constructors To access objects via object reference variables To define a reference variable using a reference type To access an object's data and methods using the object member access operator (.) To define data fields of reference types and assign default values for an object's data fields To distinguish between object reference variables and primitive data type variables To use the Java library classes Date, Random, and Point2D To distinguish between instance and static variables and methods To define private data fields with appropriate getter and setter methods To encapsulate data fields to make classes easy to maintain To develop methods with object arguments and differentiate between primitive-type arguments and object-type arguments To store and process objects in arrays To create immutable objects from immutable classes to protect the contents of objects To determine the scope of variables in the context of a class To use the keyword this to refer to the calling object itself
3,4	6	**Object-oriented**	Class abstraction and encapsulation	To apply class abstraction to develop software

		thinking	Thinking in objects Class relationships Case study: designing the course class Case study: designing a class for stacks Processing primitive data type values as objects Automatic conversion between primitive types and Wrapper class types The BigInteger and BigDecimal classes The String class The StringBuilder and StringBuffer classes Programming exercises Programming assignment	To explore the differences between the procedural paradigm and object-oriented paradigm To discover the relationships between classes To design programs using the object-oriented paradigm To create objects for primitive values using the wrapper classes (Byte, Short, Integer, Long, Float, Double, Character, and Boolean) To simplify programming using automatic conversion between primitive types and wrapper class types To use the BigInteger and BigDecimal classes for computing very large numbers with arbitrary precisions To use the String class to process immutable strings To use the StringBuilder and StringBuffer classes to process mutable strings
5,6	6	**Inheritance and Polymorphism**	Superclasses and subclasses, Superclasses and subclasses methods Using super keyword Overriding methods Overriding vs overloading, Polymorphism Dynamic binding Casting objects and the instanceof operator. The Object's equals method The ArrayList class Case study: a custom stack The protected data and methods Preventing extending and overriding Programming exercises	To define a subclass from a superclass through inheritance To invoke the superclass's constructors and methods using the super keyword To override instance methods in the subclass To distinguish differences between overriding and overloading To explore the toString() method in the Object class To discover polymorphism and dynamic binding To describe casting and explain why explicit downcasting is necessary To explore the equals method in the Object class To store, retrieve, and manipulate objects in an ArrayList To construct an array list from an array, to sort and shuffle a list, andto obtain max and min element from a list

			Programming assignment	To implement a Stack class using ArrayList To enable data and methods in a superclass accessible from subclasses using the protected visibility modifier To prevent class extending and method overriding using the final
7,8	6	**Exception Handling and Text I/O**	Exception-Handling Overview Exception types More on exception handling The finally clause When to use exceptions Rethrowing exceptions Chained exceptions Defining custom exception classes The File class File input and output Reading data from the Web Case study: Web Crawler Programming exercises Programming assignment	To get an overview of exceptions and exception handling To explore the advantages of using exception handling To distinguish exception types: Error (fatal) vs. Exception (nonfatal)and checked vs. unchecked To declare exceptions in a method header To throw exceptions in a method To write a try-catch block to handle exceptions To explain how an exception is propagated To obtain information from an exception object To develop applications with exception handling To use the finally clause in a try-catch block To use exceptions only for unexpected errors To rethrow exceptions in a catch block To create chained exceptions To define custom exception classes To discover file/directory properties, to delete and rename files/ directories, and to create directories using the File class

				To write data to a file using the PrintWriter class To use try-with-resources to ensure that the resources are closed automatically To read data from a file using the Scanner class To understand how data is read using a Scanner To develop a program that replaces text in a file To read data from the Web To develop a Web Crawler
9 10	6	**Abstract Classes and Interfaces**	Abstract classes Case study: the AbstractNumber Class Case study: Calendar and GregorianCalendar Interfaces The Comparable interface The Cloneable interface Interfaces vs. abstract classes Case Study: the Rational class Class design guidelines Programming exercises Programming assignment	To design and use abstract classes To generalize numeric wrapper classes, BigInteger, and BigDecimal using the abstract Number class To process a calendar using the Calendar and GregorianCalendar classes To specify common behavior for objects using interfaces To define interfaces and define classes that implement interfaces To define a natural order using the Comparable interface To make objects cloneable using the Cloneable interface To explore the similarities and differences among concrete classes, abstract classes, and interfaces To design the Rational class for processing rational numbers To design classes that follow the class-design guidelines
11	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course
12	3	**F2F Project Workshop** (in BMU computer rooms, optionally -	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report

		online)		
13	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.5 Course 5: Java 3: GUI Programming

Duration: 17 days, 14 online teaching days, 2 day workshop days

Number of hours: 3 hours per online/workshop day, Total: 48 hours

ECTS: 4

Day	Hours	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1,2	6	**Swing Graphical User Interfaces Basics (GUI)**	Swing vs. AWT The Java GUI API Frames Layout Managers Using Panels as Subcontainers The Color Class The Font Class Common Features of Swing GUI Components Image Icons JButton JCheckBox JRadioButton Labels Text Fields Programming exercises Programming assignment	To distinguish between Swing and AWT To describe the Java GUI API hierarchy To create user interfaces using frames, panels, and simple GUI components . To understand the role of layout managers and use the FlowLayout, GridLayout, and BorderLayout managers to lay out components in a container To use JPanel to group components in a subcontainer To create objects for colors using the Color class To create objects for fonts using the Font class To apply common features such as borders, tool tips, fonts, and colors on Swing components To decorate the border of GUI components To create image icons using the ImageIcon class.To create and use buttons using the JButton class. To create and use check boxes using the JCheckBox class To create and use radio buttons using the JRadioButton class To create and use labels using the JLabel class To create and use text fields using the JTextField class
3,4	6	**Graphics in Java**	The Graphics class Drawing Strings, Lines, Rectangles, and Ovals Case study: The FigurePanel class Drawing Arcs Drawing Polygons and Polylines Centering a String using the FontMetrics class	To draw graphics using the methods in the Graphics class To override the paintComponent method to draw graphics on a GUI component To use a panel as a canvas to draw graphics To draw strings, lines, rectangles, ovals, arcs, and polygons To obtain font properties using FontMetrics and to display a text centered in a panel To display an image on a GUI component

			Case study: The MessagePanel class Case study: The StillClock class Displaying images Case study: The ImageViewer class Programming exercises Programming assignment	To develop the reusable GUI components FigurePanel, MessagePanel, StillClock, and ImageViewer
5,6	6	**Java FX - Basics**	JavaFX vs Swing and AWT The basic structure of a JavaFX program Panes, UI Controls, and Shapes Property binding Common properties and methods for Nodes The Color class The Font class The Image and ImageView classes Layout Panes Shapes Case study: The ClockPane class Programming exercises Programming assignment	To distinguish between JavaFX, Swing, and AWT To write a simple JavaFX program and understand the relationship among stages, scenes, and nodes To create user interfaces using panes, UI controls, and shapes To update property values automatically through property binding To use the common properties style and rotate for nodes To create colors using the Color class To create fonts using the Font class To create images using the Image class and to create image views using the ImageView class To layout nodes using Pane, StackPane, FlowPane, GridPane, BorderPane, HBox, and VBox To display text using the Text class and create shapes using Line,Circle, Rectangle, Ellipse, Arc, Polygon, and Polyline To develop the reusable GUI component ClockPane for displaying an analog clock
7,8	6	**Event Driven Programming**	Events and Event Sources Registering Handlers and Handling Events Inner classes Anonymous Inner class handlers Simplifying Event Handling Using Lambda Expressions Case study: Loan Calculator Mouse events	To get a taste of event-driven programming To describe events, event sources, and event classes To define handler classes, register handler objects with the source object, and write the code to handle events To define handler classes using inner classes To define handler classes using anonymous inner classes To simplify event handling using lambda

			Key events Listeners for Observable Objects Animation Case study: Bouncing ball Programming exercises Programming assignment	expressions To develop a GUI application for a loan calculator To write programs to deal with MouseEvents To write programs to deal with KeyEvents To create listeners for processing a value change in an observable object To use the Animation, PathTransition, FadeTransition, and Timeline classes to develop animations To develop an animation for simulating a bouncing ball
9 10 11 12	12	**JavaFX UI Controls and Multimedia**	Labeled and Label Button CheckBox RadioButton TextField TextArea ComboBox ListView ScrollBar Slider Case study: Developing a Tic-Tac-Toe game Video and Audio Case study: National Flags and Anthems Programming exercises Programming assignment	To create graphical user interfaces with various user-interface controls To create a label with text and graphic using the Label class and explore properties in the abstract Labeled class To create a button with text and graphic using the Button class and set a handler using the setOnAction method in the abstract ButtonBase class (§16.3). To create a check box using the CheckBox class To create a radio button using the RadioButton class and group radio buttons using a ToggleGroup To enter data using the TextField class and password using the PasswordField class To enter data in multiple lines using the TextArea class To select a single item using ComboBox To select a single or multiple items using ListView To select a range of values using ScrollBar To select a range of values using Slider and explore differences between ScrollBar and Slider To develop a tic-tac-toe game To view and play video and audio using the Media, MediaPlayer, and MediaView To develop a case study for showing the national flag and playing anthem
13	3	**Binary I/O**	How is text I/O handled in Java? Text I/O vs. binary I/O Binary I/O classes	To discover how I/O is processed in Java To distinguish between text I/O and binary I/O To read and write bytes using

			Case study: Copying files Object I/O Random-access files Programming exercises Programming assignment	FileInputStream and FileOutputStream To filter data using the base classes FilterInputStream and FilterOutputStream To read and write primitive values and strings using DataInputStream and DataOutputStream To improve I/O performance by using BufferedInputStream and BufferedOutputStream To write a program that copies a file To store and restore objects using ObjectOutputStream and ObjectInputStream To implement the Serializable interface to make objects serializable To serialize arrays To read and write files using the RandomAccessFile class
14	3	**Software Testing with JUnit**	Software unit testing. JUnit test Metods of assertions validation Testing of aggregations. Pameters in testing. Testing of exceptions. Use of @Rule Programming exercises Programming assignment	To understand what is unit testing. To learn how to use JUnit test To learn how to validate assertions. To learn how to test aggregations. To understand what are parameters in testing. To learn how to test exceptions. To learn to use @Rule.
15	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course
16	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report
17	3	**Final**	Students get examination questions and problems	To evaluate knowledge and skills acquired during the course

		examination (in BMU computer rooms)	Exam duration - 3 hours	

3.2.6 Course 6: Java 4: Data Structures and Algorithms – Part A

Duration: 17 days, 14 online teaching days, 2 day workshop days, 4 ECTS

Number of hours: 3 hours per online/workshop day, Total: 45 hours

Day	**Ho-urs**	**Teaching units**	**Topics**	**Objectives – knowledge or skills that the student should receive**
1,2	6	**Recursion**	Recursion Definition, Case Study: Computing Factorials, Case Study: Computing Fibonacci Numbers, Problem Solving Using Recursion, Recursive Helper Methods. Case Study: Tower of Hanoi, Recursion vs. Iteration, Tail Recursion. Programming exercises Programming assignment	To describe what a recursive method is and the benefits of usingrecursion To develop recursive methods for recursive mathematical functions To explain how recursive method calls are handled in a call stack To solve problems using recursion To use an overloaded helper method to design a recursive method To implement a selection sort using recursion To implement a binary search using recursion To get the directory size using recursion To solve the Tower of Hanoi problem using recursion To draw fractals using recursion To discover the relationship and difference between recursion and iteration To know tail-recursive methods and why they are desirable
3,4	6	**Generics**	Motivations and benefits Defining generic classes and interfaces Generic methods Case study: sorting an array of objects Raw types and backward compatibility Wildcard generic types Erasure and restrictions on generics Case study: generic matrix class Programming exercises Programming assignment	To describe the benefits of generics To use generic classes and interfaces To define generic classes and interfaces To explain why generic types can improve reliability and readability To define and use generic methods and bounded generic types To develop a generic sort method to sort an array of Comparableobjects To use raw types for backward compatibility To explain why wildcard generic types are necessary To describe generic type erasure and list certain restrictions and limitations on generic types caused by type erasure To design and implement generic matrix classes
5,6	6	**List, Stack, Queue and**	Collections, Iterators,	To explore the relationship between interfaces and classes in the Java Collections Framework hierarchy

		PriorityQueue	Lists, The Comparator Interface, Static Methods for Lists and Collections Case Study: Bouncing Balls, Vector and Stack Classes Programming exercises Programming assignment	To use the common methods defined in the Collection interface for operating collections To use the Iterator interface to traverse the elements in a collection To use a foreach loop to traverse the elements in a collection To explore how and when to use ArrayList or LinkedList to store a list of elements To compare elements using the Comparable interface and the Comparator interface To use the static utility methods in the Collections class for sorting, searching, shuffling lists, and finding the largest and smallest element in collections To develop a multiple bouncing balls application using ArrayList To distinguish between Vector and ArrayList and to use the Stack class for creating stacks To explore the relationships among Collection, Queue, LinkedList, and PriorityQueue and to create priority queues using the PriorityQueue class To use stacks to write a program to evaluate expressions
7,8	6	**Set and Map**	Sets, Comparing the performance of Sets and Lists, Case study: counting keywords Maps. Case study: Occurrences of words, Singleton and Unmodifiable Collections and Maps Programming exercises Programming assignment	To store unordered, nonduplicate elements using a set To explore how and when to use HashSet LinkedHashSet or TreeSet to store a set of elements. To compare the performance of sets and lists To use sets to develop a program that counts the keywords in a Java source file To tell the differences between Collection and Map and describe when and how to use HashMap, LinkedHashMap, or TreeMap to store values associated with keys To use maps to develop a program that counts the occurrence of the words in a text To obtain singleton sets, lists, and maps, and unmodifiable sets, lists, and maps, using the static methods in the Collections class
9	12	**Developing**	Measuring algorithm	To estimate algorithm efficiency using the

10 11 12		**Efficient Algorithms**	efficiency using big o notation Examples: determining big O Analyzing algorithm time complexity Finding Fibonacci numbers using dynamic programming Finding greatest common divisors using Euclid's algorithm Efficient algorithms for finding prime numbers Finding the closest pair of points using divide-and-conquer Solving the eight queens problem using backtracking Computational geometry: finding a convex hull Programming exercises Programming assignment	Big O notation To explain growth rates and why constants and nondominating terms can be ignored in the estimation To determine the complexity of various types of algorithms). To analyze the binary search algorithm To analyze the selection sort algorithm To analyze the Tower of Hanoi algorithm To describe common growth functions (constant, logarithmic, loglinear, quadratic, cubic, exponential) To design efficient algorithms for finding Fibonacci numbers using dynamic programming To find the GCD using Euclid's algorithm To find prime numbers using the sieve of Eratosthenes To design efficient algorithms for finding the closest pair of points using the divide-and-conquer approach To solve the Eight Queens problem using the backtracking approach To design efficient algorithms for finding a convex hull for a set of points
13 14	6	**Sorting**	Insertion Sort Bubble Sort Merge Sort Quick Sort Heap Sort Bucket Sort and Radix Sort External Sort Programming exercises Programming assignment	To study and analyze time complexity of various sorting algorithms To design, implement, and analyze insertion sort To design, implement, and analyze bubble sort To design, implement, and analyze merge sort To design, implement, and analyze quick sort To design and implement a binary heap To design, implement, and analyze heap sort To design, implement, and analyze bucket sort and radix sort To design, implement, and analyze external sort for files that have a large amount of data
15	3	**F2F Project Workshop** (in BMU	Distribution of projects assignments Students work on their	To learn how to specify a project To learn how to organize the project and to break-down tasks

		computer rooms, optionally - online)	project tasks with assistance of instructors	To implement acquired knowledge during the course
16	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report
17	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.7 Course 7: Java 5: Data Structures and Algorithms – Part B

Duration: 16 days, 13 online teaching days, 2 day workshop days, 4 ECTS

Number of hours: 3 hours per online/workshop day, Total: 45 hours

Day	**Ho-urs**	**Teaching units**	**Topics**	**Objectives – knowledge or skills that the student should receive**
1,2	6	**Implementing Lists, Stacks, Queues,** **and Priority Queues**	Common Features for Lists Array Lists Linked Lists Stacks and Queues Priority Queues Programming exercises Programming assignment	To design common features of lists in an interface and provide skeleton implementation in a convenience abstract class To design and implement an array list using an array To design and implement a linked list using a linked structure To design and implement a stack class using an array list and a queue class using a linked list To design and implement a priority queue using a heap
3,4	6	**Binary Search Trees**	Binary search srees Deleting elements from a BST Tree visualization and MVC Iterators Case study: data compression Programming exercises Programming assignment	To design and implement a binary search tree To represent binary trees using linked data structures To search an element in a binary search tree To insert an element into a binary search tree To traverse elements in a binary tree To design and implement the Tree interface, AbstractTree class, and the BST class To delete elements from a binary search tree To display a binary tree graphically To create iterators for traversing a binary tree To implement Huffman coding for compressing data using a binary tree

5,6	6	**AVL Trees**	Rebalancing Trees Designing Classes for AVL Trees Overriding the insert Method Implementing Rotations Implementing the delete Method The AVLTree Class Testing the AVLTree Class AVL Tree Time Complexity Analysis Programming exercises Programming assignment	To know what an AVL tree is To understand how to rebalance a tree using the LL rotation, LR rotation, RR rotation, and RL rotation To design the AVLTree class by extending the BST class To insert elements into an AVL tree To implement tree rebalancing To delete elements from an AVL tree To implement the AVLTree class To test the AVLTree class To analyze the complexity of search, insertion, and deletion operations in AVL trees
7,8	6	**Hashing**	What Is Hashing? Hash Functions and Hash Codes Handling Collisions Using Open Addressing Handling Collisions Using Separate Chaining Load Factor and Rehashing Implementing a Map Using Hashing Implementing Set Using Hashing Programming exercises Programming assignment	To understand what hashing is and what hashing is used for To obtain the hash code for an object and design the hash function to map a key to an index To handle collisions using open addressing To know the differences among linear probing, quadratic probing, and double hashing (§27.4). To handle collisions using separate chaining To understand the load factor and the need for rehashing To implement MyHashMap using hashing To implement MyHashSet using hashing

9 10 11	9	**Graphs and Applications**	Basic Graph Terminologies Representing Graphs Modeling Graphs Graph Visualization Graph Traversals Depth-First Search (DFS) Case Study: The Connected Circles Problem Breadth-First Search (BFS) Case Study: The Nine Tails Problem Programming exercises Programming assignment	To model real-world problems using graphs and explain the SevenBridges of Königsberg problem To describe the graph terminologies: vertices, edges, simple graphs, weighted/unweighted graphs, and directed/undirected graphs To represent vertices and edges using lists, edge arrays, edge objects, adjacency matrices, and adjacency lists To model graphs using the Graph interface, the AbstractGraph class, and the UnweightedGraph class To display graphs visually To represent the traversal of a graph using the AbstractGraph.Tree class To design and implement depth-first search To solve the connected-circle problem using depth-first search To design and implement breadth-first search To solve the nine-tail problem using breadth-first search
12 13	6	**Weighted Graphs and Applications**	Representing Weighted Graphs The WeightedGraph Class Minimum Spanning Trees Finding Shortest Paths Case Study: The Weighted Nine Tails Problem Programming exercises Programming assignment	To represent weighted edges using adjacency matrices and adjacency lists To model weighted graphs using the WeightedGraph class that extends the AbstractGraph class To design and implement the algorithm for finding a minimum spanning tree To define the MST class that extends the Tree class To design and implement the algorithm for finding single-source shortest paths To define the ShortestPathTree class that extends the Tree class To solve the weighted nine tails problem using the shortest-path algorithm
13	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course

14	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students’ project. Presentation of the project report
15	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.8 Course 8: Java 6: Java ME

Duration: 14 days, 11 online teaching days, 2 day workshop days, 4 ECTS

Number of hours: 3 hours per online/workshop day, Total: 39 hours

Day	Ho-urs	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1	3	**Introduction to Java ME platform**	Configurations, Profiles, Packages CLDC CDC Java Class Library to Fit the CLDC Creating CLDC/MIDP Application using NetBean Creating CDC Application	
2,3	6	**CLDC Development with MIDP**	Introducing MIDlets. Building User Interfaces Storing Data Using the Record Store Using the Java Mobile Game API	
4	3	**CDC Development**	Introducing Xlets and the Personal Basis Profile Introducing Applets and the Advanced Graphics and User Interface Using Remote Method Invocation	
5	3	**Accessing Remote Data on the Network**	Generic Connection Framework (GCF) Communicating with Sockets and Datagrams Communicating with HTTP	
6	3	**Accessing Web Services**	Looking at a Web Service from the Client Perspective Exploring XML Support for Web Services in Java ME	
7	3	**Messaging with the Wireless Messaging API**	Wireless Messaging Services Wireless Messaging API Using the Push Registry Applying the Wireless Messaging API	
8	3	**Securing Java ME Applications**	Java ME's Security and Trust Services Exploring the Bouncy Castle Solution to Security Challenges Creating Secure Commerce with Contactless Communications	

9	3	**Rendering Multimedia Content**	Introducing the MMAPI Introducing the Java Scalable 2D Vector Graphics API Putting the MMAPI and the SVGAPI to Work	
10	3	**Using Locations**	Introducing the MMAPI Introducing the Java Scalable 2D Vector Graphics API Putting the MMAPI and the SVGAPI to Work	
11	3	**Seeking a Common Platform**	Understanding the Role JSRs Play in Fragmentation Understanding the JTWI Understanding the MSA	
12	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course
13	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report
15	3	**Final 6examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

Reference: Beginning Java™ ME Platform, Ray Rischpater, Apress, Inc., 2008

3.2.9 Course 9: Java 7: Advanced Java Programming

Duration: 15 days, 12 online teaching days, 2 day workshop days, 4 ECTS

Number of hours: 3 hours per online/workshop day, Total: 42 hours

Day	Hours	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1 2 3 4	12	**Multithreading and Parallel Programming**	Thread Concepts Creating Tasks and Threads The Thread Class Case Study: Flashing Text Thread Pools Thread Synchronization Synchronization Using Locks Cooperation among Threads Case Study: Producer/Consumer Blocking Queues Semaphores Avoiding Deadlocks Thread States Synchronized Collections Parallel Programming Programming exercises Programming assignment	To get an overview of multithreading To develop task classes by implementing the Runnable interface To create threads to run tasks using the Thread class To control threads using the methods in the Thread class To control animations using threads and use Platform.runLater to run the code in the application thread To execute tasks in a thread pool To use synchronized methods or blocks to synchronize threads to avoid race conditions To synchronize threads using locks To facilitate thread communications using conditions on locks To use blocking queues (ArrayBlockingQueue, LinkedBlockingQueue, PriorityBlockingQueue) to synchronize access to a queue To restrict the number of concurrent tasks that access a shared resource using semaphores To use the resource-ordering technique to avoid deadlocks To describe the life cycle of a thread To create synchronized collections using the static methods in the Collections class To develop parallel programs using the Fork/Join Framework

5,6	6	**Network programming**	Client/Server Computing The InetAddress Class Serving Multiple Clients Sending and Receiving Objects Case Study: Distributed Tic-Tac-Toe Games Programming exercises Programming assignment	
7,8	6	**Database programming (JDBC)**	Relational Database Systems SQL JDBC PreparedStatement CallableStatement, Retrieving Metadata Programming exercises Programming assignment	Understanding relational databases concept and RDBMS systems. Understanding the relational model, relational data structure, restrictions and language. SQL use in working with relational databases. Set up and usage of JDBC. Application of memorized SQL procedures and functions. Work with metadata about a database.
9 10	6	**Java Persistence API**	Entity Relations, Automated generation of JPA entities Programming exercises Programming assignment	Understanding ORM and complete mastery of the application of ORM tools in working with databases.

11 12	6	**Java Hibernate ORM**	Hibernate ORM – Mapping objects in database Example of creation of a persistent class Hibernate Annotations Hibernate Query Language - HQL Criteria of selection of objects in HQL query Using SQL in Hibernate environment Hibernate cashing Hibernate batch processing Hibernate interceptors Programming exercises Programming assignment	To implement Java Hibernate ORM in Java applications.
13	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course
14	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report
15	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.10 Course 10: Java 8: Java Enterprise Edition

Duration: 24 days, 21 online teaching days, 2 day workshop days, 7 ECTS

Number of hours: 3 hours per online/workshop day, Total: 69 hours

Day	Ho-urs	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1 2	6	**Java EE - Servlets**	Java EE Platform Introduction to Servlets Creating and Deploying Servlets Data Flow Servlet and Sessions GlassFish Server Programming exercises Programming assignment	To understand the concept of distributed systems and Java Enterprise Edition platform basics. Ability to create and use servelts in Java enterprise applications.
3 4 5 6	12	**Java Server Pages (JSP)**	JSP Architecture JSP Life Cycle JSP Syntax JSP Directives JSP Actions JSP Imlicit Objects Form Processing JSP Filters Cookies Handling in JSP File Upload in JSP Date Handling in JSP Redirection in JSP JSTL - JavaServer Pages Standard Tag Library JSP - Databases JSP - JavaBean JSP – Expression Language JSP Internationalization Programming exercises Programming assignment	Using JavaServer Pages (JSP), web pages' development technologies supporinng dynamic content application, and enabling Java code insertion into HTML documents. Mastering the advanced concept of application principles of JSP pages in JAVA web applications.

7 8 9 10	12	**Java Server Faces (JSF)**	Introduction to JavaServer Faces Forms in JSF Creating CDI named bean, Implementing the confirmation page, JSF Validation. Facelets templating, Resource library contracts, PrimeFaces Component Library, ICEFaces Component Library, RichFaces Component Library Programming exercises Programming assignment	Using JSF technology for Java web application development. Developing advanced JSF applications, with simplified approach through application of JSF component libraries.
11 12	6	**RESTFul Web Services with JAX – RS**	Generating a RESTful web service from an existing database Testing RESTful web service Generating RESTful Java client code Generating RESTful JavaScript clients for our RESTful web services Programming exercises Programming assignment	Understanding and use of RESTFul Web Services with JAX – RS.
13 14	6	**Context and Dependency Injection**	Introduction to CDI, Qualifiers, Sterotypes, Interceptor Binding Types , Custom CDI Scopes Programming exercises Programming assignment	Understanding and use of CDI concepts and techniques in Java EE applications.

15 16	6	**JMS and Message Driven Beans**	Introduction to JMS, Creating JMS resources, Implementing a JMS message producer, Consuming JMS messages with message-driven beans Programming exercises Programming assignment	Understanding and use of Java Messaging System and message driven beans in Java EE applications.
17 18	6	**Java API for JSON processing**	JSON-P object model API, Generating JSON data with the JSON-P object model API , Parsing JSON data with the JSON-P object model API , JSON-P streaming API, Generating JSON data with the JSON-P streaming API, Parsing JSON data with the JSON-P streaming API Programming exercises Programming assignment	Understanding and use of Java EE mechanisms for JSON processing
19	3	**Java API for WebSocket**	Examining the WebSocket code using samples included with NetBeans, Echo Application, Examining the generated Java code , Building our own WebSocket applications, Java EE, WebSocket, JS i HTML 5 – Case Study Programming exercises Programming assignment	Competence to create individual WebSocket applications.

20 21	6	**Implementing the Business Tier with Session Beans**	Introducing session beans Creating a session bean, Accessing the bean from a client, Session bean transaction management Implementing aspect-oriented programming with interceptors EJB Timer servis Generating session beans from JPA entities Programming exercises Programming assignment	To implement Session beans in Java EE applications.
22	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course
23	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report
24	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.11 Course 11: Software Development Process and Methodologies

Duration: 18 days, 15 online teaching days, 2 day workshop days, 5 ECTS

Number of hours: 3 hours per online/workshop day, Total: 21 hours

Day	Ho-urs	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1	3	**Introduction**	Professional software development Software engineering ethics Case studies Programming exercises Programming assignment	To understand what software engineering is and why it is important; To understand that the development of different types of software systems may require different software engineering techniques; To understand some ethical and professional issues that are important for software engineers; To have been introduced to three systems, of different types, that will be used as examples throughout the book.
2 3	6	**Software Processes**	Software process models Process activities Coping with change The Rational Unified Process Programming exercises Programming assignment	To understand the concepts of software processes and software process models; To have been introduced to three generic software process models and when they might be used; To know about the fundamental process activities of software requirements engineering, software development, testing, and evolution; To understand why processes should be organized to cope with changes in the software requirements and design; To understand how the Rational Unified Process integrates good software engineering practice to create adaptable software processes.

4 5	6	**Agile Software Development**	Agile methods Plan-driven and agile development Extreme programming Agile project management Scaling agile methods Programming exercises Programming assignment	To understand the rationale for agile software development methods, the agile manifesto, and the differences between agile and plan-driven development; To know the key practices in extreme programming and how these relate to the general principles of agile methods; To understand the Scrum approach to agile project management; To be aware of the issues and problems of scaling agile development methods to the development of large software systems.
6 7	6	**Requirement s** **engineering**	Functional and non-functional requirements The software requirements document Requirements specification Requirements engineering processes Requirements elicitation and analysis Requirements validation Requirements management Programming exercises Programming assignment	To understand the concepts of user and system requirements and why these requirements should be written in different ways; To understand the differences between functional and nonfunctional software requirements; To understand how requirements may be organized in a software requirements document; To understand the principal requirements engineering activities of elicitation, analysis and validation, and the relationships between these activities; To understand why requirements management is necessary and how it supports other requirements engineering activities

8 9	6	**System modeling**	Context models Interaction models Structural models Behavioral models Model-driven engineering Programming exercises Programming assignment	To understand how graphical models can be used to represent software systems; To understand why different types of model are required and the fundamental system modeling perspectives of context, interaction, structure, and behavior; To have been introduced to some of the diagram types in the Unified Modeling Language (UML) and how these diagrams may be used in system modeling; To be aware of the ideas underlying model-driven engineering, where a system is automatically generated from structural and behavioral models.
10	3	**Architectural design**	Architectural design decisions Architectural views Architectural patterns Application architectures Programming exercises Programming assignment	To understand why the architectural design of software is important; To understand the decisions that have to be made about the system architecture during the architectural design process; To have been introduced to the idea of architectural patterns, well-tried ways of organizing system architectures, which can be reused in system designs; To know the architectural patterns that are often used in different types of application system, including transaction processing systems and language processing systems.

11 12 13	9	**Design and implementation**	Object-oriented design using the UML Design patterns Implementation issues Open source development Programming exercises Programming assignment	To understand the most important activities in a general, objectoriented design process; To understand some of the different models that may be used to document an object-oriented design; To know about the idea of design patterns and how these are a way of reusing design knowledge and experience; To have been introduced to key issues that have to be considered when implementing software,
14	3	**Software testing**	Development testing Test-driven development Release testing User testing Programming exercises Programming assignment	To understand the stages of testing from testing, during development to acceptance testing by system customers; To have been introduced to techniques that help you choose test cases that are geared to discovering program defects; To understand test-first development, where you design tests before writing code and run these tests automatically; To know the important differences between component, system, and release testing and be aware of user testing processes and techniques.
15	3	**Software evolution**	Evolution processes Program evolution dynamics Software maintenance Legacy system management Programming exercises Programming assignment	To understand that change is inevitable if software systems are to remain useful and that software development and evolution may be integrated in a spiral model; To understand software evolution processes and influences on these processes; To have learned about different types of software maintenance and the factors that affect maintenance costs; and To understand how legacy systems can be assessed to decide whether they should be scrapped, maintained, reengineered, or replaced.

16	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course
17	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report
18	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.12 Course 12: Software Construction

Duration: 21 days, 18 online teaching days, 2 day workshop days, 6 ECTS

Number of hours: 3 hours per online/workshop day, Total: 60 hours

Day	Ho-urs	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1 2	6	**Software Construction Fundamentals**	1.1. Minimizing Complexity 1.2. Anticipating Change 1.3. Constructing for Verification 1.4. Reuse 1.5. Standards in Construction	To understand what is software construction.
3 4	6	**Managing Construction**	2.1. Construction in Life Cycle Models 2.2. Construction Planning 2.3. Construction Measurement	To be able to manage software construction.
5 6 7	9	**Practical Considerations**	3.1. Construction Design 3.2. Construction Languages 3.3. Coding 3.4. Construction Testing 3.5. Construction for Reuse 3.6. Construction with Reuse 3.7. Construction Quality 3.8. Integration	To implement software construction technics in design, coding, testing, software reusing, quality and insoftware integration
8 9	6	**Construction Technologies**	4.1. API Design and Use 4.2. Object-Oriented Runtime Issues 4.3. Parameterization and Generics 4.4. Assertions, Design by Contract, and Defensive Programming	To learn to implement design API To understand OO runtime issues To implement parameterization and generics To implement assertions, design by contract and defensive programming
10 11	6		4.5. Error Handling, Exception Handling, and Fault Tolerance 4.6. Executable Models 4.7. State-Based and Table-Driven Construction Techniques	To implement error handling, exeption handling and fault tolerance To use executable models To implement state-based and table-driven construction techniques

12 13	6		4.8. Runtime Configuration and Internationalization 4.9. Grammar-Based Input Processing 4.10. Concurrency Primitives 4.11. Middleware	To implement runtime configuration and internationalization To implement grammar-based input processing To implement concurrency primitives To implement middleware
14 15	6		4.12. Construction Methods for Distributed Software 4.13. Constructing Heterogeneous Systems 4.14. Performance Analysis and Tuning 4.15. Platform Standards 4.16. Test-First	To implement construction methods for distributed software To implement constructing of heterogeneous systems To use performance analysis and tunng To implement platform standards To implement test/first approach
17 18	6	**Software Construction** **Tools**	5.1. Development Environments 5.2. GUI Builders 5.3. Unit Testing Tools 5.4. Profiling, Performance Analysis, and Slicing Tools Matrix of Topics vs. Reference Material	To be able to use development environments and tools, such as GUI builders, unit testing tools, profiling, performance analysis and slicing tools
19	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Distribution of projects assignments Students work on their project tasks with assistance of instructors	To learn how to specify a project To learn how to organize the project and to break-down tasks To implement acquired knowledge during the course
20	3	**F2F Project Workshop** (in BMU computer rooms, optionally - online)	Students work on their project tasks with assistance of instructors	To develop necessary Java programs To realize all programming tasks of students' project. Presentation of the project report

21	3	**Final examination** (in BMU computer rooms)	Students get examination questions and problems Exam duration - 3 hours	To evaluate knowledge and skills acquired during the course

3.2.13 Course 13: Software Development Project

Duration: 16 days, 5 online teaching days, 10 days workshop days, 4 ECTS

Number of hours: 3 hours per online/workshop day, Total: 45 hours

Day	Ho-urs	Teaching units	Topics	Objectives – knowledge or skills that the student should receive
1	3	**Project Management**	Risk management Managing people Teamwork	To know the principal tasks of software project managers; To have been introduced to the notion of risk management and some of the risks that can arise in software projects; To understand factors that influence personal motivation and what these might mean for software project managers; To understand key issues that influence team working, such as team composition, organization, and communication.
2	3	**Project Planning**	Software pricing Plan-driven development Project scheduling Agile planning Estimation techniques	To understand the fundamentals of software costing and reasons why the price of the software may not be directly related to its development cost; To know what sections should be included in a project plan that is created within a plan-driven development process; To understand what is involved in project scheduling and the use of bar charts to present a project schedule; To have been introduced to the 'planning game', which is used to support project planning in extreme programming; To understand how the COCOMO II model can be used for algorithmic cost estimation.

3	3	**Quality Management**	Software quality Software standards Reviews and inspections Software measurement and metrics	To understand to the quality management process and know why quality planning is important; To understand that software quality is affected by the software development process used; To be aware of the importance of standards in the quality management process and know how standards are used in quality assurance; To understand how reviews and inspections are used as a mechanism for software quality assurance; To understand how measurement may be helpful in assessing some software quality attributes and the current limitations of software measurement.
4	3	**Configuration Management**	Change management Version management System building Release management	To understand the processes and procedures involved in software change management; To know the essential functionality that must be provided by a version management system, and the relationships between version management and system building; To understand the differences between a system version and a system release, and know the stages in the release management process.
5	3	**Service-Oriented Software Engineering**	Service-oriented Architecture Services as reusable components Service engineering Software development with services	To understand the rationale for software process improvement as a means of improving both product quality and the efficiency and effectiveness of software processes; To understand the principles of software process improvement and the cyclic process improvement process; To know how the Goal-Question-Metric approach may be used to guide process measurement; To have been introduced to the ideas of process capability and process maturity, and the general form of the SEI’s CMMI model for process improvement.

6 7 8 9 10 11 12 13 14 15	30	**Software Development Project**	Students spend 3 hours in a computer room and develop their group projects (cc 5 students per project). Their instructor is helping them during the software development. Students may choose to work online instead F2F.	To implement software management knowledge To develop a software using quality management principles, and configuration management
16	1	**Final examination** (in BMU computer rooms)	Presentation of projects	To demonstrate their ability to develop a software, as a team.

4 Pedagogical Approach to SCHE courses

BMU SCHE Java Developer target the following categories of students:

- Bachelor degree holders with or without job, willing to change their profession and job
- Master degree holders interested to learn Java programming, as they need for their jobs
- Individuals that abandoned their bachelor studies and are seeking to get a quick qualification of a Java Developer (in 12 months) and find a job as soon as possible
- Fresh graduates from secondary schools not interested to get bachelor degrees and planning to get a Java Developer job

Some of students may be employed and they cannot be full-time students following F2F (face-to-face) courses. The same is the case with students not living in Belgrade or Niš, towns where BMU has campuses. Therefore, BMU decided to implement SCHE program providing (Figure 3.1):

- Online courses,
- F2F or online two days workshops at the end of each course, allowing students to realize their project assignments, and
- An exam after each course and its workshop.

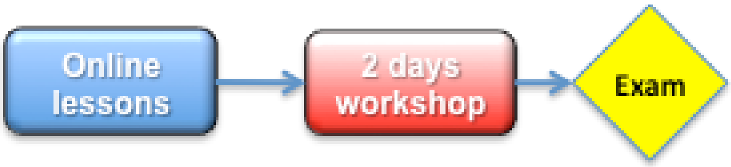


Figure 3.1: Three components of a SCHE Java Developer course

Instead of academic organization of courses (4-5 courses per semester realized in parallel during 15 weeks), it is expected that a SCHE program may be more effective if courses are sequentially thought, as shown in Figure 3.2. Exams should demonstrated students' ability to implement what they learnt. If they fail, they will have one additional exam. If they fall again, they cannot proceed with the SCHE program and must wait a new group of students of the SCHE Job Developer, and continue their program with the course that didn't pass.

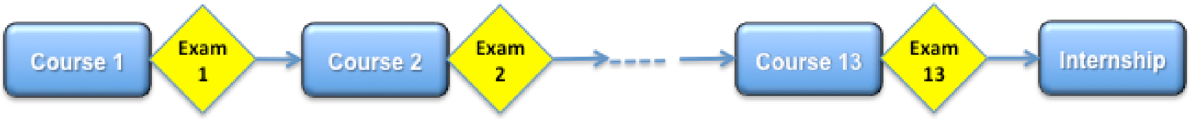


Figure 3.2: Sequential implementation of courses of SCHE Java Developer

Students will be organized in groups of 20, having their own tutor (one per group). Tutor will communicate with online students every days monitoring their work and giving them consultations. Tutors will also check results of given assignments to students and of their testing. Tutors will organize P2D or online workshops (for those not being able to participate in F2F workshops), aiming the course projects. Each student will get his project assignment that he must to complete by the end of workshop and before the exam, planned for the next day.

Figure 3.3 shows the organization of an online lesson. It consists of a number of topics and sub-topics. A topic or sub-topic consist of one or more sections that contain contents in form of multimedia web pages created by mDita Editor developed by BMU.

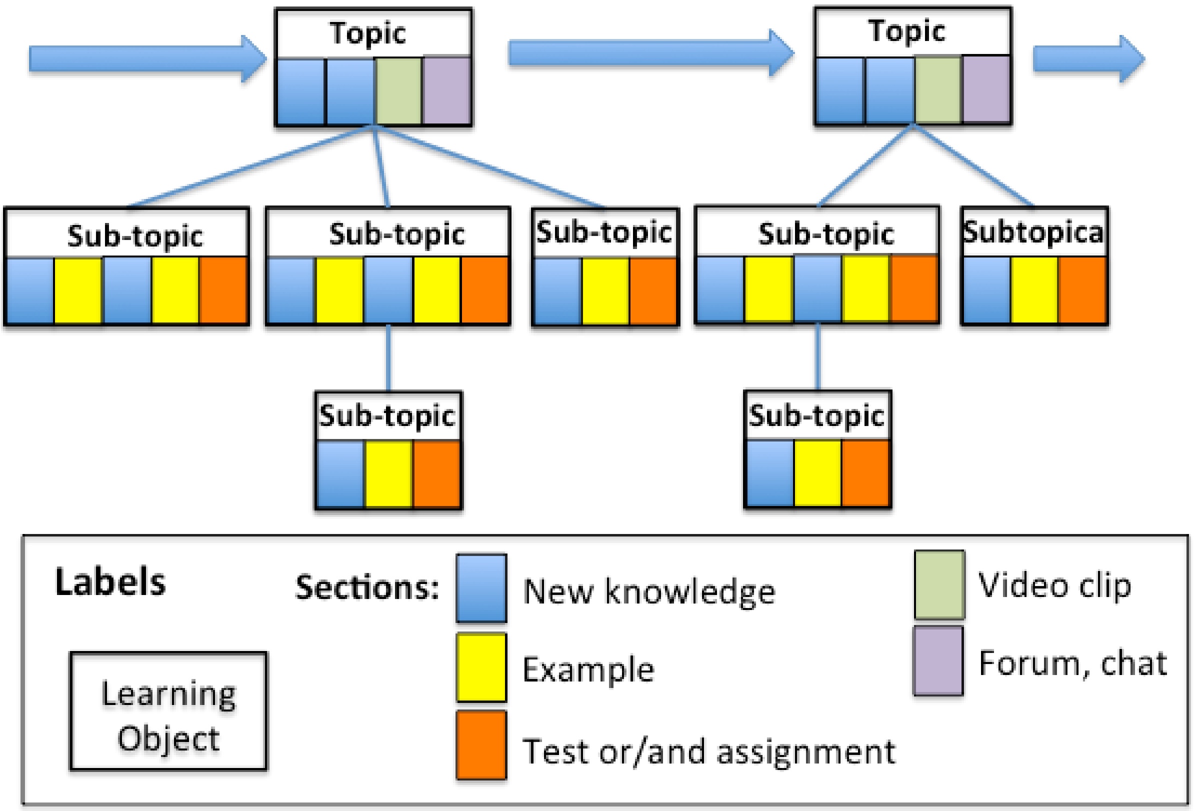
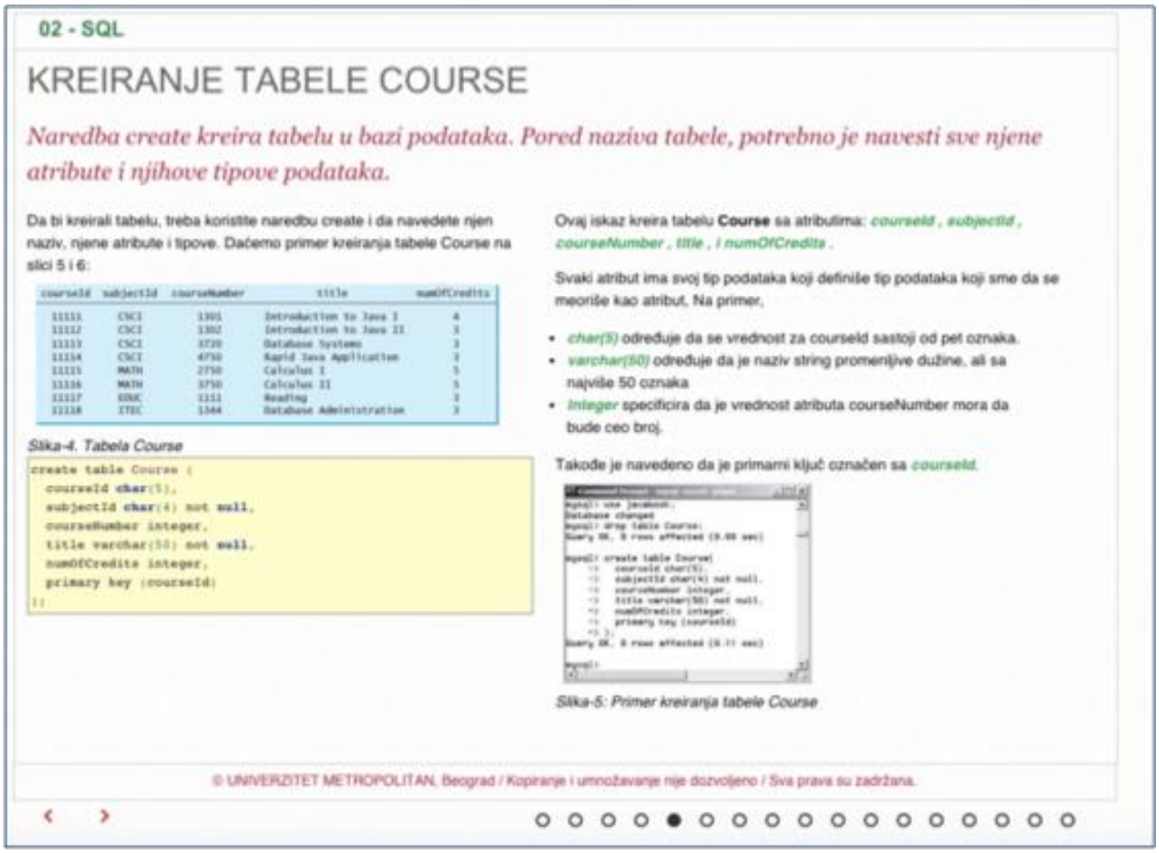


Figure 3.3 : Organization of an online lesson with learning objects, related to topics and sub-topics using sections of different kinds

An online lessons contains a number of learning objects with one or more sections. Sections may provide now knowledge concepts, examples, assignments, tests, video clips, forums or chats. First order learning objects (or LO) contains topic sections or/and sub/topic sections. Each section is multimedia web page that contains textual information, video and audio clips, listings of Java codes and evaluation sections, such as different kind of tests and assignments. Authors of courses organize online lessons as hierarchy of learning objects related to topics and sub-topics. Online lessons, topics and subtopics are specified according to knowledge units and topics defined in BOM (the Body of Knowledge) of the SCHE Java Developer. Hours on online lessons are rough estimation of durations of online lessons, but the focus is on lessons' content, not in their durations.

Delivery of online lessons id managed by LAMS (Learning Activity Management System). It was chosen as it supports the concepts of learning objects and learning activities, organized in processes with branching. It is necessary for achieving a kind of personalization of e-learning, as different learning content may be offered to different students or group of students, based on their ability to learn and their knowledge levels.

Figure 3.4 shows one section (web page) created by mDita editor.



02 - SQL

KREIRANJE TABELE COURSE

Naredba create kreira tabelu u bazi podataka. Pored naziva tabele, potrebno je navesti sve njene atribute i njihove tipove podataka.

Da bi kreirali tabelu, treba koristite naredbu create i da navedete njen naziv, njene atribute i tipove. Daćemo primer kreiranja tabele Course na slici 5 i 6:

courseId	subjectId	courseNumber	title	numOfCredits
11111	CSCI	1301	Introduction to Java I	4
11112	CSCI	1302	Introduction to Java II	3
11113	CSCI	3720	Database Systems	3
11114	CSCI	4750	Rapid Java Application	3
11115	MATH	2750	Calculus I	5
11116	MATH	3750	Calculus II	5
11117	EDUC	1111	Reading	3
11118	ITEC	1344	Database Administration	3

Slika-4. Tabela Course

```
create table Course (
  courseId char(5),
  subjectId char(4) not null,
  courseNumber integer,
  title varchar(50) not null,
  numOfCredits integer,
  primary key (courseId)
);
```

Ovaj iskaz kreira tabelu **Course** sa atributima: *courseId* , *subjectId* , *courseNumber* , *title* , *i numOfCredits* .

Svaki atribut ima svoj tip podataka koji definiše tip podataka koji sme da se meoriše kao atribut. Na primer,

- *char(5)* određuje da se vrednost za courseId sastoji od pet oznaka.
- *varchar(50)* određuje da je naziv string promenljive dužine, ali sa najviše 50 oznaka
- *Integer* specificira da je vrednost atributa courseNumber mora da bude ceo broj.

Takođe je navedeno da je primarni ključ označen sa *courseId*.

```
mysql> use javabook;
Database changed
mysql> drop table Course;
Query OK, 0 rows affected (0.08 sec)

mysql> create table Course(
    ->   courseId char(5),
    ->   subjectId char(4) not null,
    ->   courseNumber integer,
    ->   title varchar(50) not null,
    ->   numOfCredits integer,
    ->   primary key (courseId)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql>
```

Slika-5: Primer kreiranja tabele Course

© UNIVERZITET METROPOLITAN, Beograd / Kopiranje i umnožavanje nije dozvoljeno / Sva prava su zadržana.

Figure 3.4: A section with learning content as shown to students by LAMS

The number of topics (first order LOs) may be different, depending of its content. The same is valid for topics and their sub-topics and sections. So, a course may have different number of lessons, with different number of learning objects for its topics, sub-topics and sections.

When planning the duration of each course, it is assumed that student can use online lessons provided by BMU e-Learning System, six day a week, and at least three learning hours per day (reading or watching video clips and listening the content of a lesson). Besides these three “learning hours”, it is expected that student spend one or more hours for doing tests and assignments related to a topic.