

TM



**EUCIP**

European Certification of  
Informatics Professionals

# EUCIP

## Software Developer

### Elective Level Profile Specification

*Version 2.4, February 2007*

#### **Short Description**

**A EUCIP Software Developer is expected to play a considerable technical role in information systems design and to be very effective in carrying out the creation and maintenance of complex software modules that typically need to be integrated into a wider information system. Different specialisations are possible, either in the field of applications and web services or in system-level software.**

This profile requires a minimum work experience of **18** months in a compatible job role; if this requirement is not fulfilled, the candidate might be certified as an **Associate** Software Developer.

## Tasks Overview

Defines detailed specifications and directly contributes to the efficient creation and/or modification of complex software systems using the proper standards and tools. Ensures that the results meet the requirements both in terms of high quality technical design and in terms of conformity with agreed functional specifications.

Is informed about available standards, methods and tools that are relevant to the specific work environment: identifies advantages and disadvantages of each, and applies them in an intelligent and effective way in order to achieve well-engineered products which ensure the necessary attributes such as fitness for purpose, reliability, efficiency, security, safety, maintainability and cost effectiveness.

Takes care of technical issues in software implementation projects and in the other stages of the life cycle for software development: investigation, analysis, specification, design, construction, testing, implementation and software maintenance. Ensures that adequate documentation is produced and maintained.

Uses database management systems software and appropriate analysis tools to analyse database performance statistics and create reports on demand, including proposals for improvement and resolution of problems.

Understands the main alternatives in configuring databases and can provide support during installation and upgrade of software servers and application tools, ensuring that version control procedures are observed, applying fixes, and maintaining supplier and user documentation.

Addresses complex and non-standard situations, taking charge of technical responsibility for implementation stages of the life cycle for software development, by participating in investigation, analysis and specification, ranging from design, construction, testing, maintenance, upgrade and migration, and achieving a well engineered product.

In case of a senior software developer leading a team, manages a group of technical staff, providing expertise in the selection, provision and use of architectures, software and facilities, taking full responsibility for the quality and timeliness of their work and ensuring effective utilisation of all assigned resources.

Where there is special focus on web systems, selects appropriate tools, templates and standards to create advanced, well designed and engineered web pages with specified content and layout. Tests pages and corrects coding errors. Assists less experienced colleagues with difficult coding problems. Sets coding standards for the employing organisation, fully taking into account bandwidth and browser compatibility issues. Specifies appropriate web server hardware and network connectivity for small and medium sized information systems.

Maintains familiarity with a substantial range of relevant web sites and participates in significant discussions on developments in web tools and technology. Uses the knowledge thus gained to advise clients/users of current/future trends, and to anticipate any necessary changes to standards in the employing organisation.

Converts logical specifications into more detailed designs taking account of the technical and non-technical features and limitations of the target implementation environment.

Interprets object/data models into appropriate DB schema within set constraints (e.g. consistency, security, ownership) and produces object database components as required.

Constructs or modifies, tests and corrects large and/or complex component modules from specifications.

Prepares and coordinates software module testing; identifies defects and causes of failure, and amends programs and system configuration until a fully positive outcome is achieved.

Contributes to defining the software modules that comprise an integration build, ensuring that they meet the defined software test criteria and producing software builds for loading onto target hardware from software source code.

## **Essential Behavioural Skills [ 1 ]<sup>1</sup>**

The Software Developer role requires first of all a rational mental attitude capable of conceptual and analytical thinking, a high regard for detail and a persistent goal-oriented approach, leading to the result through structured solutions formulated in a flexible way.

Another relevant set of skills is the ability to communicate and interact effectively (in both oral and written form) with colleagues and clients: this shall include a general organisational and cross-functional awareness, a good teamworking approach, efficiency in information acquisition, in as much as the ability to plan, organise, make technical decisions, provide directions and follow-up.

Examples of possible courses include techniques for effective meetings, time management, teamworking.

---

<sup>1</sup> numbers in brackets represent EUCIP points

## Detailed Skills Required

### Deep competence level [ 20 ]

#### B3.01 Programming [ 4,5 ]

- Use different programming design methods, such as Object-Oriented (OO) design, “top down” design, structured programming.
- Know how to use abstraction as a technique of problem-solving and design.
- Cope with the specific needs of legacy systems in program design.
- Use different data structures such as records, arrays, and linked lists.
- Decide when to use each one of the data structures above and related algorithms.
- Use some of the main types of programming languages (different generations, functional, procedural, OO-based) to compose new algorithms and functions or to modify existing programs.
- Interpret correctly syntax in programming languages.
- Choose between compiled and interpreted programming languages.

#### B1.07 Object oriented approach to systems development [ 1,5 ]

- Appreciate Object Design approach.
- Appreciate the benefits of “objects” and subsequent re-use of software.
- Understand the use of objects and classes.
- Appreciate Abstraction, Encapsulation, Polymorphism and Messaging Concepts.
- Appreciate the use of Implementing and Testing Models in UML.
- Assist in identifying UML Classes.
- Use standard notation and conventions for Classes.
- Construct Class Diagrams (UML).
- Define Attributes, Associations, Operations and Methods for a Class.
- Make use of inheritance and aggregation hierarchies.
- Appreciate the differences between persistent and non-persistent Classes.
- Appreciate the differences between OO design and other programming design methods.

#### B1.08 Software engineering principles [ 1,5 ]

- Understand roles of the software engineering process (project manager, software developer, maintenance staff, quality assurance and the user).
- Understand software development life cycle models and their applications.
- Understand and apply software development estimation techniques
- Understand and apply principles of software Project Management
- Understand Risk Management
- Understand Quality Assurance

- Understand Configuration Identification, Control and Auditing
- Understand Configuration Status Accounting
- Understand and apply Software Estimating Techniques and Metrics

### **B2.03 Working with databases [ 1,5 ]**

- Use SQL for:
  - o basic 'select' statements
  - o restricting and sorting data
  - o transforming data through single-row functions
  - o displaying data from multiple tables and views
  - o aggregating data using group functions
  - o extracting complex results through subqueries
  - o data manipulation (DML commands)
- Produce readable output through interactive SQL.
- Import and export data: methods include the bulk copy.
- Manage result sets by using cursors and SQL: considerations include locking models and appropriate usage.
- Extract data in XML format: considerations include output format and XML schema structure.
- Manage data manipulation by using stored procedures, transactions, triggers, user-defined functions, and views.
- Control data access by using stored procedures, triggers, user-defined functions, and views.
- Define object-level security including column-level permissions by using GRANT, REVOKE, and DENY.
- Know how to use standard Database interfaces like ODBC, JDBC, etc.

### **B1.09 Computer Aided Software Engineering (CASE) and Integrated Development Environment (IDE) tools [ 1 ]**

- Know when and how to use a CASE tool: top issues related to CASE tools adoption, CASE Tools for different platforms/languages.
- Work with the most used IDEs for Windows and Unix platforms.
- Integrate plug-ins in an IDE. Examples: Oracle Developer 2000, Rational ROSE, Select, Business Objects.
- Customise the build process in an IDE.
- Use the "Configuration Manager" (Debug/Build...).
- Integrate the IDE with a Version Control System. Example: CVS.

### **A6.02 Develop in a collaborative environment [ 1,5 ]**

- Use tools for team work in a collaborative environment.
- Cope with primary issues related to a team work.
- Manage Version Control, Technical Documents, and Distribution tools.
- Manage Build and Test.
- Use messaging tools such as IM, Mailing List, discussion boards.
- Facilitate a collaborative environment.
- Apply procedures for team work.
- Acknowledge the importance of an established set of documentation and coding standard.

- Exploit detailed knowledge and troubleshooting hints available through virtual communities of developers.

### **B3.02 Languages [ 4,5 ]**

- Write effective source code in a specific procedural programming language.
- Example: Basic, Pascal, C, Cobol, etc.
- Use a specific OO programming language.
- Example: C++, Java, Delphi, etc.
- Use a scripting language.
- Example: PERL, Python, Ruby, etc.
- Define the principles of Mark-up Languages.
- Use Extensible Mark-up Language (XML), use provided tools to execute XML-friendly database queries, employ XML technology in programs and applications, know XSLT and how to use it to transform a document.

### **B3.03 Software development process [ 1,5 ]**

- Write documentation: proper formats, tools, internal documentation.
- Develop formal methods, use tools and environments for software engineering, recognise the role of programming paradigm and process maturity.
- Perform Rapid Prototyping.
- Perform testing/acceptance/deployment procedures:
  - o development of major UI components
  - o development of prototypes to explore any other system uncertainties like response time, scalability etc.
- Apply methods and techniques for planning and monitoring progress of projects. Examples: work breakdown structures, critical path analysis, conflict resolution.
- Correct course and control changes, according to the Change Control Process.
- Apply a proper coding process in a development environment aimed at a massively parallel execution, as well as for embedded systems, real time response systems and very high availability systems.
- Conduct acceptance testing.
- Identify milestones.
- Test functionality, system stress and load.
- Use commercial tools packages for various types of testing and bug tracking.
- Build an acceptance test.
- Support deployment and hand-over.
- Provide application and technical support.

**B3.04 Designing and developing distributed and critical applications [ 1,5 ]**

- Choose the right level of transaction support.
- Plan and design for performance, maintainability, extensibility, availability, scalability, and reliability. Considerations include:
  - o number of transactions per time increment
  - o bandwidth
  - o capacity
  - o peak versus average usage requirements
  - o response-time expectations
  - o barriers to performance
  - o processes per server
  - o parallel execution
  - o maintenance expectations
  - o location and knowledge level of maintenance staff
  - o impact of third-party maintenance agreements
  - o hours of operation
  - o level of availability
  - o impact of downtime
  - o growth of the partners
  - o growth of the company
  - o volume of documents
- Design integration with existing applications. Derive the physical design.
- Install remote components: considerations include attended and unattended installations.
- Troubleshoot failed installations.
- Identify situations for applying custom components.
- Monitor and optimise performance: tools include performance counters, Event Viewer, Windows Management Instrumentation (WMI).
- Diagnose and resolve implementation errors.

**B3.07 Build reports [ 1 ]**

- Administer server resources.
- Create high quality web reports.
- Use templates to create mailing labels and letters.
- Create and modify basic tabular reports.
- Build reports using XML.
- Add dynamic data to an HTML page.
- Identify the main components in a report document and how they are related.
- Publish a report on the web.
- Tune reports.
- Create other report styles such as break reports and matrix reports.
- Use report parameters and customise a runtime parameter form.
- Manage report templates.
- Create and embed a graph in a report.
- Identify standard report design styles and run existing reports to various output destinations.

## **Incisive competence level [ 11 ]**

### **B3.06 Secure programming [ 1 ]**

- Understand and apply the principles of secure coding:
  - o Appropriate access control,
  - o Least privilege concept,
  - o Validation and control of input data,
  - o Buffer overflow concept.
  - o Understand secure programming issues:
    - o Socket security,
    - o RPC and DCOM security,
    - o Java applet and ActiveX control security,
    - o EJB and RMI.
- Minimise, isolate and simplify code running with raised privileges.
- Cope with main security issues related to code and data structures.
- Distrust all values external to the program (e.g. arguments, environment variables, etc.).
- Avoid use of any function that copies without checking buffer lengths.
- Avoid link with dynamic libraries, link statically.
- Avoid creation of temporary files in world-writable directories (e.g. /tmp).
- Recognise race conditions.
- Design the security infrastructure. Design the deployment architecture: considerations include security, performance, maintainability, extensibility, availability, scalability, and reliability.
- Monitor performance counters and event logs.
- Understand fundamentals of static analysis of source code.
- Define security code review processes.
- Define secure software deployment procedures.

### **C2.01 Operating Systems [ 2 ]**

- Differentiate between the most widespread operating systems:
  - o Linux/Unix
  - o Windows
  - o MacOS
- Install and upgrade the above OSs.
- Cope with OS conceptual problems:
  - o concurrency management, deadlock and starvation
  - o scheduling
  - o I/O operation and management
  - o file management systems
  - o user and access management
- Analyse network capabilities.
- Configure network interfaces.
- Configure various network protocols and services (including http, SMTP, POP, IMAP, DNS).
- Start and stop various network services.
- Publish resources on the network (e.g. shared printers and folders).
- Measure and monitor system load:

- CPU (both mono- and multi-processor)
- network
- memory and virtual memory
- storage
- processes and threads
- usage of shared resources
- Tune the system to reach required performances.
- Manage user accounts and groups and set up related security policies.
- Apply interoperability tips (file formats, available protocols, etc.).
- Set up systems to reach the required level of interoperability between heterogeneous OSs.
- Use performance boosting techniques as clustering.
- Setup clustering.
- Perform troubleshooting.
- Perform system recovery.

### **A3.03 Solution envisaging [ 1 ]**

- Envisage and develop a solution concept.
- Analyse the feasibility of the solution.
- Analyse and refine the scope of the solution project.
- Identify key project risks.
- Contribute to gathering and analysing business requirements.
- Create a conceptual model of business requirements or data requirements: methods include Object Role Modelling (ORM) and UML.
- Validate the conceptual design
- Create the logical design for the solution.
- Create the logical data model.
- Validate the proposed logical design.
- Write a clear specification document.

### **B1.12 Defining a solution architecture [ 1 ]**

- Gather and analyse:
  - user requirements,
  - operational requirements,
  - system requirements for hardware, software, and network infrastructure.
- Transform requirements into functional specifications: considerations include performance, maintainability, extensibility, scalability, availability, deployability, security, and accessibility.
- Transform functional specifications into technical specifications: considerations include performance, maintainability, extensibility, scalability, availability, deployability, security, and accessibility.
- Select the appropriate technologies for the physical design of the solution.
- Create the physical design for:
  - the solution,
  - deployment,
  - maintenance,
  - the data model.
- Create specifications for auditing and logging.

- Validate the physical design.

#### **A5.03 Project coordination [ 1 ]**

- Coordinate a software development project: planning, control, organisation, configuration management, version control, quality assurance, metrics.
- Establish standards applying to development documentation, coding, code review, UI, and testing.
- Establish processes: processes include reviewing development documentation, reviewing code, creating builds, tracking issues, managing source code, managing change, managing release, and establishing maintenance tasks.
- Contribute to establishing quality and performance metrics to evaluate project control and organisational performance.
- Report actual progress of activities against an agreed plan.

#### **B1.06 Object oriented approach to systems analysis [ 1 ]**

- Act as an effective member of a team involved in analysis using an OO approach.
- Appreciate how the system design approach in the OO paradigm differs from other approaches.
- Use the main OO analysis modelling types and show how they relate to each other.
- Evaluate the benefits of the OO approach to analysis (business and systems).
- Appreciate the use of the OO Model types in UML.
- Use UML Analysis models.
- Perform Business domain modelling (in UML).
- Contribute to Activity Modelling (in UML).
- Create Use Cases in requirements gathering.
- Appreciate UML Dynamic Modelling techniques (e.g. STDs, Sequence and Collaboration diagrams).
- Appreciate UML Design and Architecture Modelling.
- Evaluate OO lifecycles, and development environments from the business view.

#### **B2.04 Designing and implementing business solutions with transactional support [ 1 ]**

- Analyse architecture requirements. Analyse security requirements.
- Analyse integration requirements. Analyse functional requirements.
- Develop an application made of Business Components.
- Implement Business Rules.
- Handle Exceptions and Errors.
- Create and manage Database Objects.
- Present business data.
- Create custom and dynamic queries.
- Leverage middle-tier validation.
- Handle Business Component Transactions.
- Package Business Services and Data Models.
- Extend and Substitute Business Components.
- Deploy Business Component Applications.
- Create JSP Clients for a Business Components Application.

- Refer to Best Practices.
- Perform system tuning for performance.

#### **B2.02 Designing and implementing databases [ 1 ]**

- Choose among different types of database architecture (e.g. relational, hierarchical, matrix, object-oriented) suitable to application requirements.
- Exploit data abstraction: physical level, conceptual level and view level, object-based logical model, record-based logical model and physical data model.
- Apply the principles of object-based logical models: i.e. the entityrelationship model, the object-oriented model.
- Define entities: considerations include entity composition and normalisation.
- Design entity keys: considerations include FOREIGN KEY constraints, PRIMARY KEY constraints, and UNIQUE constraints.
- Design attribute domain integrity: considerations include CHECK constraints, data types, and nullability.
- Know how to use database design tools. Examples: Oracle Designer 2000, ERWin.
- Implement a physical database.
- Create and alter databases: considerations include file groups, file placement, growth strategy, and space requirements.
- Create and alter database objects: objects include constraints, indexes, stored procedures, tables, triggers, user-defined functions, and views.
- Define object-level security including column-level permission by using GRANT, REVOKE, and DENY
- Alter database objects to support replication and partitioned views.
- Troubleshoot failed object creation.

#### **B4.02 Designing and developing web applications [ 1 ]**

- Choose platforms that support each programming language and environment.
- EITHER:
  - o Master servlets and JSPs, which are the most popular components of the J2EE standard and critical elements used by companies building e-commerce sites.
  - o Build web-based applications using Java servlets and Java Server Pages (JSP). Know the concepts and use of the servlet API, plus the productive development of applications through Java Server Pages.
- OR:
  - o Master COM/COM+/.NET and ASP.
  - o Build web-based applications using ASP or VBA in a .NET environment. Know the concepts and use of web services.

#### **B4.03 Build internet applications [ 1 ]**

- Create form modules, including components for database interaction and GUI controls.
- Reuse objects and code.
- Choose appropriate data sources for data blocks.

- Ensure application security.
- Create and manage multiple-form Internet applications.
- Handle the notion of stateless connection and use of sessions.

## Annexes

<b>Sample Learning Modules</b>	<b>EUCIP Points</b>
EUCIP CORE PLAN	X
EUCIP CORE BUILD	X
EUCIP CORE OPERATE	X
EUCIP IT ADMINISTRATOR	
<i>2. Operating Systems</i>	<b>A 4/4</b>
Univ. SW Engineering	<b>E 7/7</b>
Univ. Operating Systems	<b>A 4/4</b>
Univ. IT Security	<b>G 2/2</b>
IBM Test 141: XML and Related Technologies	<b>D 1/3</b>
IBM Test 285: Developing with IBM WebSphere Studio v5	<b>E 1/2</b>
IBM Test 286: Application Development with WebSphere Studio v5	<b>D 2/3</b>
IBM Test 287: Enterprise Application Development with WebSphere Studio v5	<b>D 1/3</b>
IBM Test 486: Object-Oriented Analysis and Design with UMLTest	<b>C 6/6</b> <b>E 1/7</b>
IBM Test 630: ClearCase for Windows	<b>E 1/7</b>
IBM AIX Admin	<b>A 3/4</b>
IBM LPI L1	<b>A 2/4</b>
Microsoft Certified Solution Developer (MCSD)	
<i>MS 70-300: Analyzing Requirements and Defining Solution Architectures for Microsoft .NET</i>	<b>E 7/7</b>
Microsoft Certified Application Developer (MCAD*)	
<i>MS 70-305: Developing and Implementing Web Applications with Microsoft Visual Basic .NET and Microsoft Visual Studio® .NET</i> <i>OR 70-315: Developing and Implementing Web Applications with Microsoft Visual C# .NET and Microsoft Visual Studio .NET</i>	<b>B 6/6</b>
<i>MS 70-306: Developing and Implementing Windows-based Applications with MS Visual Basic .NET and MS Visual Studio .NET</i> <i>OR 70-316: Developing and Implementing Windows-based Applications with MS Visual C# .NET and MS Visual Studio .NET</i>	<b>C 6/6</b>
<i>MS 70-310: Developing XML Web Services and Server Components with Microsoft Visual Basic .NET and the Microsoft .NET Framework</i> <i>OR 70-320: Developing XML Web Services and Server Components with Microsoft Visual C# .NET and the Microsoft .NET Framework</i>	<b>D 3/3</b>
<i>MS 70-229: Designing and Implementing Databases with MS SQL Server 2000 Enterprise Edition (or equivalent*)</i>	<b>F 3/3</b>
<i>MS 70-330: Implementing Security for Applications with Microsoft Visual Basic .NET</i>	<b>G 2/2</b>
Microsoft Certified Systems Administrator (MCSA*)	
<i>MS 70-290: Managing and Maintaining a Microsoft Windows Server 2003 Environment</i>	<b>A 4/4</b>

<b>Sample Learning Modules</b>	<b>EUCIP Points</b>
Oracle Certif. DBA	
<i>1Z0-007: Introduction to Oracle9i SQL</i>	<b>F 1/3</b>
<i>1Z0-031: Oracle9i Database Administration Fundamentals I</i>	<b>F 1/3</b>
<i>1Z0-042: Oracle Database 10g Administration I</i>	<b>F 2/3</b>
Oracle Certif. Developer	
<i>1Z0-141: Oracle Forms Developer - Build Internet Applications</i>	<b>F 2/3</b>
Sun Certif. System Administrator for the Solaris OS	<b>A 4/4</b>
Sun Certif. Developer for the Java2 Platform	<b>C 2/6</b>
	<b>D 1/3</b>
Sun Certif. Enterprise Architect for J2EE	<b>C 2/6</b>
	<b>E 3/7</b>
Sun Certif. Programmer for the Java2 Platform	<b>B 3/6</b>
	<b>C 2/6</b>
Sun Certif. Web Component Developer for J2EE	<b>D 3/3</b>
Sun Certif. Business Component Developer for J2EE	<b>F 1/3</b>
	<b>G 1/2</b>

## **External references to SFIA<sup>®</sup> version 3 by the SFIA Foundation**

### **Skill 22: Programming/software development**

*“The design, creation, testing and documenting of new and amended programs from supplied specifications in accordance with agreed standards.”*

Levels 4 and 5

### **Skill 19: Systems design**

*“The specification and design of information systems, their components and architecture to meet defined business needs.”*

Levels 4 and 5

## **External references to AITTS by the German Government – *Arbeitsprozessorientierten Weiterbildung in der IT-Branche***

### ***Profil 1.3: Software Developer (Softwareentwickler/in)***

*“Software Developer konzipieren und implementieren einzelne Software-Bausteine [Komponenten und Module].”*

## **External references to *Nomenclature 2005* by CIGREF (club informatique des grandes entreprises françaises)**

### ***Métier 4.2: Développeur***

*“À la demande de la maîtrise d’œuvre, et sur la base des spécifications fonctionnelles émises par celle-ci, le développeur analyse, paramètre et code les composants logiciels applicatifs dans le respect des normes et procédures, ainsi que les évolutions souhaitées.”*