Specialized Platform Development (SPD)

Preamble

The Specialized Platform Development (SPD) Knowledge Area (KA) refers to at- tributes involving creating a software platform to address specific needs and requirements for particular areas. Specialized platforms, such as healthcare providers, financial institutions, or transportation companies, are tailored to meet specific needs. Developing a specialized platform typically involves several key stages, i.e., requirements, design, development, deployment, and maintenance.

Changes since CS 2013: *The SPD Beta Versions considered the following factors:*

- ***Emerging Computing Areas*** such as data science/analytics – that use multi- platforms to retrieve sensing data. Cybersecurity – involves protecting certain data extraction, recognizing protocols to protect network transfer ability, and manipulating it. Artificial intelligence and machine learning – use artifacts that retrieve information for robotics and drones to perform specific tasks. This continuous emergence of computing technology areas has increased the appetite to develop platforms communicating software with specialized environments. This need has also increased the need to develop specialized programming languages for these platforms, such as web and mobile development. The Interactive Computing Platform addresses the advent of Large Language Models (LLMs), such as OpenAI's ChatGPT, OpenAI's Codex, and GitHub's Copilot, in addition to other platforms that perform data analysis, and visualizations.
- **Industry needs and competencies** have created a high demand for developers on specialized platforms, such as mobile, web, robotics, embedded, and interactive. Some of the unique professional competencies obtained by current job descriptions relevant to this KA are:
    - *Create a mobile app that provides a consistent user experience across various devices, screen sizes, and operating systems.*
    - *Analyze people's experience using a novel peripheral for an immersive system facilitated using a head-mounted display and mixed reality, with attention to usability and accessibility specifications.*
    - *Build and optimize a secure web page for evolving business needs using a variety of appropriate programming languages.*
    - *Develop application programming interfaces (APIs) to support mobile functionality and remain current with the terminology, concepts, and best practices for coding mobile apps.*
    - *Availability of devices and artifacts, such as raspberry PIs, Arduinos, and mobile devices. The low cost of microcontrollers and devices, such as robots using ROS that can perform specialized actions, has*

The consideration of these factors resulted in the following significant changes from the CS2013 version:

- **Renamed of the Knowledge Area name**: From Platform-Based Development (PBD) to Specialized Platform Development due to the specific needs of the already mentioned tasks. This particular KA is often called *software platform development* since the specialized development takes part in the software stages for multi-platform development.
- **Increase of Computer Science Core Hours**: Based on the already mentioned needs, the SPD beta version has increased the number of computer science course hours from 0 to 9. The KA subsets the web and mobile knowledge units (often the most closely related units in CS Core) into foundations and specialized platforms core hours to provide flexibility and adaptability. This division allows programs at different institutions to offer different interests, concentrations, or degrees that focus on different application areas, where many of these concepts intersect the CS core. Therefore, the *common aspects, web* and mobile foundations have concepts in many CS programs' core. Finally, the rest of the knowledge units permit the curriculum to have an extended and flexible number of KA core hours.
- **Renamed old knowledge units and incorporated new ones**: in the spirit of capturing the future technology and societal responsibilities, the Robotics, Embedded Systems, and Society, Ethics, and Professionalism (SEP) knowledge units were introduced in this version. These KU work harmoniously with other KAs, consistent with the topics and concepts covered in specific KAs' knowledge units.

Specialized platform development provides a deep understanding of a particular user group's needs and requirements and considers other knowledge areas that help design and build a platform that meets other KA's needs. Considering other KAs' needs, SPD helps to streamline workflows, improve efficiency, and drive innovation across the recommended curriculum discussed in CS2023.

Core Hours

Knowledge Units	CS Core	KA Core
SPD/Web Foundations	2	
SPD/Mobile Foundations	1	
SPD/Common Aspects		3
SPD/Web Platforms		*
SPD/Mobile Platforms		*
SPD/Robot Platforms		*
SPD/Embedded Platforms		*

SPD/Game Platforms		*
SPD/Interactive Computing Platforms		*
Total	**3**	

Knowledge Units

SPD/Introduction → Common Aspects/Shared Concerns

This unit aims to develop core concepts related to specialized platform development. Students shall recognize the need to develop for various specialized platforms and their corresponding applications, the programming languages used for these applications, and how to effectively use such languages.

- Topics
    a. Overview of platforms (e.g., Web, Mobile, Game, Industrial)
        i. Input/Sensors/Control Devices/Haptic devices
        ii. Resource constraints
            - Computational
            - Data storage
            - Communication
            - Societal, Compliance, Security, Uptime availability, fault tolerance
        iii. Output/Actuators/Haptic devices
    b. Programming via platform-specific Application Programming Interface (API) vs traditional application construction
    c. Overview of Platform Languages (e.g., Kotlin, Swift, C#, C++, Java, JavaScript, HTML5)
    d. Programming under platform constraints (e.g., available development tools, development)
    e. Techniques for learning and mastering a platform-specific programming language.
- Illustrative Learning Outcomes
    a. List the constraints of mobile programming
    b. Describe the three-tier model of web programming
    c. Describe how the state is maintained in web programming
    d. List the characteristics of scripting languages

SPD/Web Platforms

This unit aims to develop concepts relating to web platforms. Concepts include programing language features, web platforms, frameworks, security and privacy considerations, architecture, and storage solutions.

- Topics
    a. Web programming languages (e.g., HTML5, JavaScript, PHP, CSS)

b. Web platforms, frameworks, or meta-frameworks
c. Software as a Service (SaaS)
d. Web standards such as document object model, accessibility
e. Security and Privacy considerations
f. Analyzing requirements for web applications
g. Computing services (e.g., Amazon AWS, Microsoft Azure)
    i. Cloud Hosting
    ii. Scalability (e.g., Autoscaling, Clusters)
    iii. How to estimate costs for these services (based on requirements)
h. Data management
    i. Data residency (where the data is located and what paths can be taken to access it)
    ii. Data integrity: guaranteeing data is accessible and guaranteeing that data is deleted when required
i. Architecture
    i. Monoliths vs. Microservices
    ii. Micro-frontends
    iii. Event-Driven vs. RESTful architectures: advantages and disadvantages
    iv. Serverless, cloud computing on demand
j. Storage Solutions
    i. Relational Databases
    ii. NoSQL databases

- Illustrative Learning Outcomes
    a. Design and Implement a web application using microservice architecture design.
    b. Describe the web platform's constraints and opportunities, such as hosting, services, and scalability, that developers should consider.
    c. Compare and contrast web programming with general-purpose programming.
    d. Describe the differences between Software-as-a-Service and traditional software products.
    e. Discuss how web standards impact software development.
    f. Review an existing web application against a current web standard.

SPD/Mobile Platforms

This unit aims to develop concepts relating to web platform technologies and considerations.

The mobile platform also offers local, on-device computing. Typical on-device security and machine-language-specific chips make possible applications with different impacts from other traditional platforms.

- Topics
    a. Development
        i. Mobile programming languages
        ii. Mobile programming environments

iii. Native versus cross-platform development
iv. Software architecture patterns used in mobile development

b. Mobile platform constraints
i. User interface design
ii. Understanding differences in user experience between mobile and web-based applications
iii. Security
iv. Power/performance tradeoff

c. Access
i. Accessing data through APIs
ii. Designing API endpoints for mobile apps: pitfalls and design considerations
iii. Network and the Web interfaces

d. Mobile computing affordances
i. Location-aware applications
ii. Sensor-driven computing (e.g., gyroscope, accelerometer, health data from a watch)
iii. Telephony, Instant messaging
iv. Augmented Reality.

e. Specification and Testing

f. Asynchronous computing
i. How it differs from traditional synchronous programming
ii. Handling success via callbacks
iii. Handling errors asynchronously
iv. Testing asynchronous code and typical problems in testing

- Illustrative Learning Outcomes
  a. Implement a location-aware mobile application that uses data APIs.
  b. Implement a sensor-driven mobile application that logs data on a server.
  c. Implement a communication app that uses telephony and instant messaging.
  d. Compare and contrast mobile programming with general-purpose programming.
  e. Describe the pros and cons of native and cross-platform mobile app development.

SPD/Robot Platforms

The robot platforms knowledge unit considers topics related to the deployment of software on existing robot platforms and the application of these robots. Concepts include robotic platforms, specialized programming languages, tools for robotic development, and the interconnection between physical and simulated systems.

- Topics
  a. Types of robotic platforms and devices
  b. Sensors, embedded computation, and effectors (actuators)
  c. Robot-specific languages and libraries
  d. Robotic platform constraints and design considerations
  e. Interconnections with physical or simulated systems

f. Robotics
    i. Robotic software architecture (e.g., using the Robot Operating System)
    ii. Forward kinematics
    iii. Inverse kinematics
    iv. Dynamics
    v. Navigation and robotic path planning
    vi. Manipulation and grasping
    vii. Safety considerations

- Illustrative Learning Outcomes
    a. Design and implement an application on a given robotic platform (e.g., using Lego Mindstorms, MATLAB, or the Robot Operating System connected to a simulator or physical robot)
    b. Assemble an Arduino-based robot kit and program it to navigate a maze
    c. Compare robot-specific languages and techniques with those used for general-purpose software development
    d. Explain the rationale behind the design of the robotic platform and its interconnections with physical or simulated systems,
    e. Given a high-level application, design a robot software architecture using ROS specifying all components and interconnections (ROS topics) to accomplish that application
    f. Discuss the constraints a given robotic platform imposes on developers

SPD/Embedded Platforms

This Knowledge unit considers embedded computing platforms and their applications. Embedded platforms cover knowledge ranging from sensor technology to ubiquitous computing applications.

Reference PDC and OS for topics related to concurrency, timing, scheduling, and timeouts.

- Topics
    a. Introduction to the Unique Characteristics of Embedded Systems
        i. real-time vs. soft real-time and non-real-time systems
        ii. Resource constraints (e.g., memory profiles, deadlines, etc.)
    b. Safety considerations and safety analysis
    c. Sensors and Actuators
    d. Embedded programming
    e. Real-time resource management
    f. Analysis and Verification
    g. Application Design
- Illustrative Learning Outcomes
    a. Design and implement a small embedded system for a given platform (e.g., a smart alarm clock or a drone)
    b. Describe unique characteristics of embedded systems versus other systems
    c. Interface with sensors/actuators
    d. Debug a problem with an existing embedded platform
    e. Identify different types of embedded architectures
    f. Evaluate which architecture is best for a given set of requirements

SPD/Game Platforms
The Game Platforms knowledge unit draws attention to concepts related to the engineering of performant real-time interactive software on constrained computing platforms. Material on requirements, design thinking, quality assurance, and compliance enhances problem-solving skills and creativity.

- Topics
    a. Historical and Contemporary Platforms for Games
        i. *Evolution of Game Platforms*: Brown Box to Metaverse and beyond; Improvement in Computing Architectures (CPU and GPU); Platform Convergence and Mobility
        ii. *Typical Game Platforms*: Personal Computer; Home Console; Handheld Console; Arcade Machine; Interactive Television; Mobile Phone; Tablet; Integrated Head-Mounted Display; Immersive Installations and Simulators; Internet of Things enabled Devices; CAVE Systems; Web Browsers; Cloud-based Streaming Systems
        iii. *Characteristics and Constraints of Different Game Platforms*: Features (local storage, internetworking, peripherals); Run-time performance (GPU/CPU frequency, number of cores); Chipsets (physics processing units, vector co-processors); Expansion Bandwidth (PCIe); Network throughput (Ethernet); Memory types and capacities (DDR/GDDR); Maximum stack depth; Power consumption; Thermal design; Endian; etc.
        iv. *Typical Sensors, Controllers, and Actuators:* typical control system designs—peripherals (mouse, keypad, joystick), game controllers, wearables, interactive surfaces; electronics and bespoke hardware; computer vision, inside-out tracking, and outside-in tracking; IoT-enabled electronics and i/o; etc.
    b. Social, Legal, and Ethical Considerations for Game Platforms
        i. *Usability*: user requirements; affordances; ergonomic design; user research; heuristic evaluation methods for games
        ii. *Accessibility:* equality and access; universal design; legislated requirements for game platforms; compliance evaluation.
        iii. *Sustainability:* materials; power usage; supply-chain; recycling; planned obsolescence; etc.
    c. Real-time Simulation and Rendering Systems
        i. *CPU and GPU architectures*: Flynn's taxonomy; parallelization; instruction sets; common components—graphics compute array, graphics memory controller, video graphics array basic input/output system; bus interface; power management unit; video processing unit; display interface, etc.
        ii. *Pipelines for physical simulations and graphical rendering:* tile-based, immediate-mode, etc.

iii. *Common Contexts for Algorithms, Data Structures, and Mathematical Functions*: game loops; spatial partitioning, viewport culling, and level of detail; collision detection and resolution; physical simulation; behavior for intelligent agents; procedural content generation; etc.
iv. *Media representations, i/o, and computation techniques for virtual worlds:* audio; music; sprites; models and textures; text; dialogue; multimedia (e.g., olfaction, tactile); etc.

d. Game Development Tools and Techniques
   i. *Programming Languages*: C++; C#; Lua; Python; JavaScript; etc.
   ii. *Shader Languages*: HLSL; GLSL; ShaderGraph; etc.
   iii. *Graphics Libraries and APIs*: DirectX; SDL; OpenGL; Metal; Vulkan; WebGL
   iv. *Common Development Tools and Environments*: IDEs; Debuggers; Profilers; Version Control Systems (including those handling binary assets); Development Kits and Production/Consumer Kits; Emulators; Engines—Open Game Engine; Unreal; Unity; Godot; CryEngine; Phyre; Source 2; Phaser; Twine; etc.
   v. *Techniques*: Ideation; Prototyping; Iterative Design and Implementation; Compiling Executable Builds; Development Operations and Quality Assurance—Play Testing and Technical Testing; Profiling; Optimization; Porting; Internationalization and Localization; Networking; etc.

e. Game Design
   i. *Vocabulary:* game definitions; mechanics-dynamics-aesthetics model; industry terminology; models of experience and emotion; etc.
   ii. *Design Thinking and User-Centered Experience Design:* methods of designing games; iteration, incrementing, and the double-diamond; phases of pre- and post-production; stakeholder and customer involvement; community management.
   iii. *Genres:* Adventure; walking simulator; first-person shooter; real-time strategy; multiplayer online battle arena (MOBA); role-playing game (rpg); etc.
   iv. *Audiences and Player Taxonomies*: people who play games; diversity and broadening participation; pleasures, player types, and preferences; Bartle, yee, etc.
   v. *Proliferation of digital game technologies to domains beyond entertainment*: Education and Training; Serious Games; Virtual Production; Esports; Gamification; Immersive Experience Design; Creative Industry Practice; Artistic Practice; Procedural Rhetoric.

- Illustrative Learning Outcomes
  a. Recall the characteristics of common general-purpose graphics processing architectures
  b. Identify the key stages of the immediate-mode rendering pipeline
  c. Describe the key constraints a given game platform will likely impose on developers

d. Translate complex mathematical functions into performant source code
e. Use an industry-standard graphics API to render a 3D model in a virtual scene
f. Modify a shader to change a visual effect according to stated requirements
g. Implement a game for a particular platform according to a specification
h. Optimize a function for processing collision detection in a simulated environment
i. Assess a game's run-time and memory performance using an industry-standard tool and development environment
j. Compare the interfaces of different game platforms, highlighting their respective implications for human-computer interaction
k. Recommend an appropriate set of development tools and techniques for implementing a game of a particular genre for a given platform
l. Discuss the key challenges in making a digital game that is cross-platform compatible
m. Suggest how game developers can enhance the accessibility of a game interface
n. Create novel forms of gameplay using frontier game platforms

SPD/Interactive Computing Platforms
This knowledge unit concerns interactive computing platforms and the use of Large Language Models (LLM) to interact with computer users based on queries and other interactivity actions. Most of these topics span applications for Data Science, Quantum Computing, and various creative disciplines. Additionally, it concentrates on LLM to interact with.

- Topics
    a. Data Analysis Platforms
        i. Jupyter notebooks; Google Colab; R; SPSS; Observable, etc.
        ii. Cloud SQL/data analysis platforms (e.g., BigQuery)
            - Apache Spark
    b. Data Visualizations
        i. Interactive presentations backed by data
        ii. Design tools requiring low-latency feedback loops
            - rendering tools
            - graphic design tools
    c. Creative coding
        i. Creative interactive frameworks (can crossover with web, embedded/IoT/other low-fidelity hardware)
            - Live Music
            - Generative Art
            - Exhibition/demonstrative works
        ii. Machine-assisted interactivity (e.g., AI/ML pairing)
    d. Large Language Models (LLMs)
        i. Use of applications such as OpenAI's ChatGPT, OpenAI's Codex, and GitHub's Copilot
- Supporting math studies:
    a. Signal analysis / Fourier analysis / Signal processing (for music composition, audio/RF analysis)
    b. Statistics (for Data Analysis)

- Supporting humanities studies
    a. Visual art
    b. Journalism and other interactive storytelling. Exploratory, data-intensive applications intended to be consumed by a wide audience.
    c. Music theory, composition

- **Illustrative Learning Outcomes**
    a. Interactively analyze large datasets
    b. Create a backing track for musical performance (e.g., with live coding)
    c. Create compelling computational notebooks that construct a narrative for a given journalistic goal/story.
    d. Implement interactive code that uses a dataset and generates exploratory graphics
    e. Create a program that performs a task using LLM systems
    f. Contrast a program developed by an AI platform and by a human
    g. Implement a system that interacts with a human without using a screen
    h. Contextualize the attributes of different data analysis styles, such as interactive vs. engineered pipeline
    i. Write a program using a notebook computing platform (e.g., searching, sorting, or graph manipulation)

SPD/Society, Ethics, and Professionalism
This knowledge unit captures the society, ethics, and professionalism aspects from the specialized platform development viewpoint. Every stage from the software development perspective impacts the SEP knowledge unit.

Topics
- Augmented technology and societal impact
- Robotic design
- Graphical User Interfaces considerations for DEI
- Recognizing data privacy and implications

Professional Dispositions

- Learning to learn (new platforms, languages)
- Inventiveness (in designing software architecture within non-traditional constraints)
- Adaptability (to new constraints)
- Learning to debug and test code

Math Requirements

Desired:
- Calculus
- Linear Algebra

- Probability/Statistics (e.g., dynamic systems, visualization e.g., algorithmically generated Tuftian-style displays)
- Discrete Math/Structures (e.g., graphs for process control and path search)

Shared Concepts and Crosscutting Themes

Shared Concepts:

- Artificial Intelligence
- Graphics
- Human-Computer Interaction
- Modeling
- Programming Languages
- Software Engineering
- Requirements Engineering

Competency Specifications

- **Task 1:** Determine whether to develop an app natively or using cross-platform tools.
- **Competency Statement:** Have technical and app design knowledge, understand performance and scalability issues, and evaluate different approaches and tools by carefully considering factors such as app requirements, target audience, time-to-market, and costs.
- **Competency area:** Application
- **Competency unit:** Evaluation
- **Required knowledge areas and knowledge units:**
    - SE / Tools and Environments
    - SPD / Common Aspects
    - SPD / Mobile Platform
- **Required skill level:** Explain
- **Core level:** CS core

- **Task 2:** Create a mobile app that provides a consistent user experience across various devices, screen sizes, and operating systems.
- **Competency Statement:** Have the technical knowledge and design skills for mobile app development, optimize the app's usability and performance, and conduct extensive testing to ensure its functionality on various devices and platforms.
- **Competency area:** Application
- **Competency unit:** Design / Development / Testing
- **Required knowledge areas and knowledge units:**
    - HCI / Understanding the User
    - PL / Object-Oriented Programming

    - SDP / Development Methods
    - SE / Tools and Environments
    - SE / Software Design
    - SE / Software Construction
    - SE / Software Verification and Validation
    - SPD / Mobile Platform
    - SEP / TBD
- **Required skill level:** Develop
- **Core level:** CS core

- **Task 3:** Using an engine, translate prototype gameplay implemented by a designer using blueprints into high-performance and maintainable code with attention to maintaining cross-platform compatibility.
- **Competency Statement:** Demonstrate an ability to implement, profile, and optimize software for a game platform based on set requirements and an ability to verify the functional coherence, performance, and portability of the solution
- **Competency area:** Software, Application
- **Competency unit:** Development, Testing, Evaluation
- **Required knowledge areas and knowledge units:**
    - SE / Software Design
    - SE / Software Construction
    - SE / Software Verification and Validation
    - AL / Algorithmic Strategies
    - SF / System Performance
    - SF / Performance Evaluation
    - PL / Object-Oriented Programming
    - SPD / Common Aspects
    - SPD / Game Platforms
- **Required skill level:** Develop
- **Core level:**

- **Task 4:** Analyze people's experience using a novel peripheral for an immersive system facilitated using a head-mounted display and mixed reality, with attention to usability and accessibility specifications.
- **Competency Statement:** Demonstrate sufficient capacity to assess the characteristics of a game platform interface for quality, legislative requirements, and end-user acceptance
- **Competency area:** Systems
- **Competency unit:** Requirements, Testing, Evaluation, Consumer Acceptance, Adaptation to Social Issues
- **Required knowledge areas and knowledge units:**

    - HCI / Understanding the User
    - HCI / Accessibility and Inclusive Design
    - HCI / Evaluating the Design
    - GIT / Immersion
    - GIT / Physical Computing
    - SE / Software Verification and Validation
    - SEP / Equity, Diversity, and Inclusion
    - SPD / Game Platforms
- **Required skill level:** Apply
- **Core level:**

- **Task 5:** Build and optimize a secure web page for evolving business needs using a variety of *appropriate* programming languages.
- **Competency Statement:** Have security and application design knowledge, understand potential security hazards and room for optimization.
- **Competency area:** Application
- **Competency unit:** Evaluation
- **Required knowledge areas and knowledge units:**
- **Required skill level:** Develop
    - AR / Performance and Energy Efficiency
    - CYB /
    - NC / Network Security
    - OS / Protection and Safety
    - SF / System Security
    - SE / Software Design
    - SE / Tools and Environments
    - SPD / Common Aspects
    - SPD / Mobile Platform
    - SEP / Privacy
- **Core level:** CS core

- **Task 6:** Develop application programming interfaces (APIs) to support mobile functionality and remain current with the terminology, concepts, and best practices for coding mobile apps.
- **Competency Statement:** Manifest proficiency as mobile app developer by designing, developing, and implementing mobile apps. Developer employs application programming interfaces for code reduction and development.
- **Competency area:** Application
- **Competency unit:** Development/Evaluation
- **Required knowledge areas and knowledge units:**
    - FPL / Hardware Interface

    - OS / File Systems API and Implementation
    - SE / Software Design
    - SE / Tools and Environments
    - SPD / Common Aspects
    - SPD / Mobile Platform
- **Required skill level:** Develop
- **Core level:** CS core

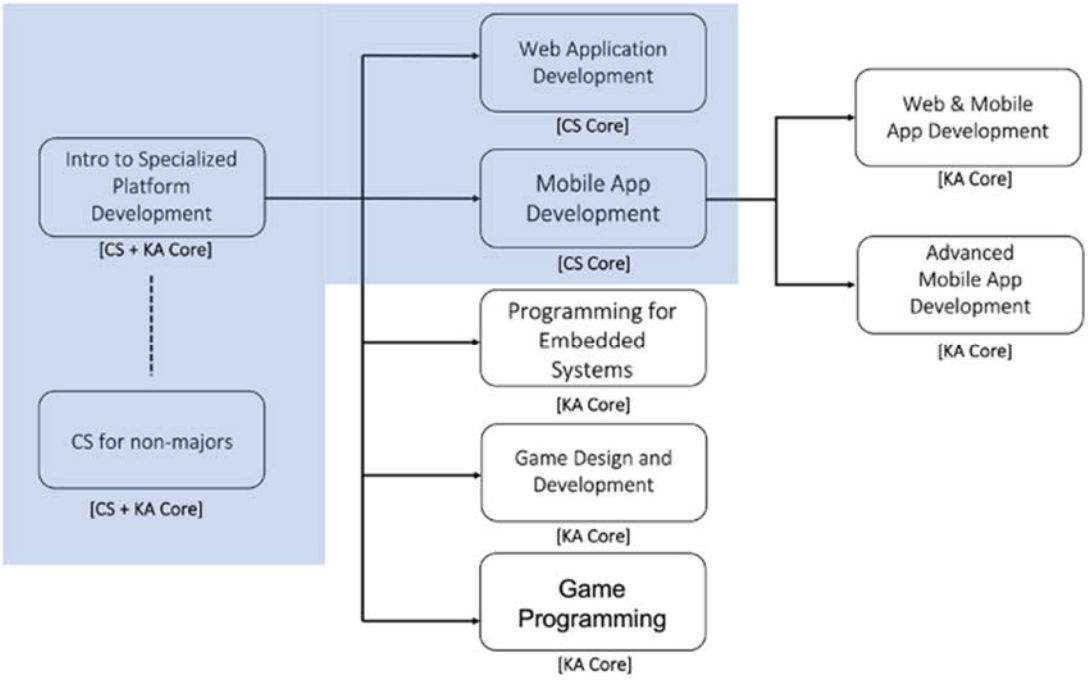
- **Task 7:** Identify robotic applications' or machines' purposes and goals and complete all design stages for systems that accomplish stated goals.****
- **Competency Statement:** Own understanding on interactivity between software and hardware interfacing to perform robotic-based applications.
- **Competency area:** Application
- **Competency unit:** Development/Evaluation
- **Required knowledge areas and knowledge units:**
    - AL
    - CYB/
    - SE / Tools and Environments
    - SE / Software Design
    - SPD / Common Aspects
    - SPD / Mobile Platform
- **Required skill level:** Explain
- **Core level:**

- **Task 8:** Design program architecture based on project requirements and hardware specifications by writing software code, embedded programs, and system protocols.
- **Competency Statement:** Indicate expertise recognizing embedded programming to various devices and interactively between libraries, platforms, and software.
- **Competency area:** Application
- **Competency unit:** Development/Deployment/Integration
- **Required knowledge areas and knowledge units:**
    - FPL/Hardware Interface
    - SE / Software Design
    - SE / Tools and Environments
    - SF / System Performance
    - SPD / Common Aspects
    - SPD / Embedded Systems
- **Required skill level:** Develop
- **Core level:**

- **Task 9:** Provide continued support for one or more web properties.
- **Competency Statement:** Indicate expertise recognizing embedded programming to various devices and interactively between libraries, platforms, and software.
- **Competency area:** Application
- **Competency unit:** Development/Deployment/Integration
- **Required knowledge areas and knowledge units:**
    - DM / NoSQL System
    - SE / Tools and Environments
    - SE / Software Design
    - SPD / Common Aspects
    - SPD / Web Development
    - NC / Single Hop Communication
    - OS / File Systems API and Implementation
- **Required skill level:** Apply
- **Core level:** CS core

- **Task 10:** Cooperating with back-end developers, designers, and the rest of the team to deliver well-architected and high-quality solutions.
- **Competency Statement:** Aptitude to discuss, synthesize, and integrate ideas from various departments related to product management.
- **Competency area:** Application
- **Competency unit:** Deployment/Integration
- **Required knowledge areas and knowledge units:**
    - SE / Project Management
    - SE / Software Design
    - SE / Teamwork
    - SPD / Common Aspects
    - SPD / Web Development
    - SPD / Mobile Development
- **Required skill level:** Explain
- **Core level:**

Course Packaging Suggestions



Committee

Chair: Christian Servin (El Paso Community College, El Paso, TX, USA)

Members:

- Sherif G. Aly, The American University in Cairo, Egypt
- Yoonsik Cheon, The University of Texas at El Paso, El Paso, Texas, USA
- Eric Eaton, University of Pennsylvania, Philadelphia, PA, USA
- Claudia L. Guevara, Jochen Schweizer mydays Holding GmbH, Munich, Germany
- Larry Heimann, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
- Amruth N. Kumar, Ramapo College of New Jersey, Mahwah, NJ, USA
- R. Tyler Pirtle, Google
- Michael Scott, Falmouth University, UK

Contributors:

- Orlando Gordillo, NASA, USA
- Sean R. Piotrowski, Rider University, USA
- Mark 0'Neil, Blackboard Inc., USA
- John DiGennaro, Qwickly
- Rory K. Summerley, Falmouth University, Penryn, Cornwall, UK.

Appendix: Core Topics and Skill Levels

KA	KU	Topic	Skill	Core	Hours
	Common Aspects	● Overview of platforms (e.g., Web, Mobile, Game, Robot, Embedded, and Interactive) ● Programming via platform-specific Application Programming Interface (API) vs traditional application construction ● Overview of Platform Languages (e.g., Kotlin, Swift, C#, C++, Java, JavaScript, HTML5) ● Programming under platform constraints (e.g., available development tools, development) ● Techniques for learning and mastering a platform-specific programming language.	Apply	CS	2
	Web Platforms	● Web programming languages (e.g., HTML5, JavaScript, PHP, CSS) ● Web platforms, frameworks, or meta-frameworks ● Software as a Service (SaaS) ● Web standards such as document object model, accessibility ● Security and Privacy considerations ● Analyzing requirements for web applications ● Computing services (e.g., Amazon AWS, Microsoft Azure) ● Data management ● Architecture ● Storage Solutions	Apply	CS	3
	Mobile Platforms	● Development ● Mobile platform constraints ● Access ● Mobile computing affordances ● Specification and Testing ● Asynchronous computing	Apply	CS	3
	Robot Platforms	● Types of robotic platforms and devices ● Sensors, embedded computation, and effectors (actuators) ● Robot-specific languages and libraries ● Robotic platform constraints and design considerations ● Interconnections with physical or simulated systems ● Robotics	Apply	KA	3
	Embedded Platforms	● Introduction to the Unique Characteristics of Embedded Systems. ● Safety considerations and safety analysis ● Sensors and Actuators ● Embedded programming ● Real-time resource management	Apply	KA	3

		● Analysis and Verification ● Application Design			
	Game Platforms	● Historic and Contemporary Platforms for Games ● Social, Legal, and Ethical Considerations for Game Platforms ● Real-time Simulation and Rendering Systems ● Game Development Tools and Techniques ● Game Design	Apply	KA	4
	Interactive	● Data Analysis Platforms ● Data Visualizations ● Creative coding ● Quantum Computing Platforms ● Language Models (LLMs) ● Supporting math studies ● Supporting humanities studies	Apply	KA	2
	SEP	● TBD		KA	3