Specialized Platform Development
(Alpha Version)

Preamble

The Specialized Platform Development (SPD) knowledge area is concerned with the design and development of software applications that reside on and interoperate with specific software platforms. SPD may be applied over a wide breadth of computing ecosystems, and is complementary to the other Knowledge Areas, providing a mechanism for students to take their programming and software development skills and apply them to a broader platform widely used in practice. The SPD considerations intersect the fundamental areas of the core of the computing curriculum, and therefore advisable to incorporate fundamental concepts in this knowledge area. There is a need for a specialized platform development curriculum mainly due to the high demand on computing needs, for both software and hardware interactively, and the appearance of application areas such as drone platforms, robotics, IoT, and quantum computing (to mention a few) in addition to the traditional ones such as web programming, multimedia development, mobile computing, app development. Computer Science curricula have incorporated and adapted best practices in teaching computer science over the past decade. Teaching core and elective courses in programming implemented in different platforms and operating systems depend on institutions’ requirements and workforce recommendations. For example, traditionally speaking, software development has evolved from platforms using text editors and command lines to integrated development environments, interactive computing platforms, and graphical user interfaces that assist computing programming development between software and hardware. Furthermore, specialized platforms demonstrate software development principles using a variety of low-cost and accessible environments, such as Raspberry PIs, and Arduinos (often used in Robotics and Drone programming). Additionally, Mobile and Web Platforms have played an essential role in industry and software engineering for the past decade, especially with emerging computing areas such as intelligence computing (i.e., Data Science, AI/ML), Cloud, and Quantum computing, platforms considerations, and constraints development are considered in this knowledge area.

With the intention to provide flexibility and adaptability to the number of topics offered into the Computer Science Core and the Specialized Knowledge Area Core this area subsets the web and mobile knowledge units (currently most influenced units in CS Core) into foundations and
specialized platforms core hours. Currently, different institution’s programs offers a variation of interests/concentrations or degrees that focus on specialized developments. However, it is recognized that many of these concepts intersect the computer science core. Therefore the Web and Mobile Foundations have concepts that can be found in the core of many computer science programs. Common Aspects Knowledge Unit concentrates exposes concepts that various specialized platforms share in common. Finally, the rest of the knowledge units permits to have an extended number of KA Core hours.

**Allocation of Core Hours**

**SPD. Specialized Platform Development**

<table>
<thead>
<tr>
<th></th>
<th>CS Core</th>
<th>KA Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPD/Web Foundations</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>SPD/Mobile Foundations</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SPD/Common Aspects</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>SPD/Web Platforms</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>SPD/Mobile Platforms</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>SPD/Robot Platforms</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>SPD/Embedded Platforms</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>SPD/Game Platforms</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>SPD/Interactive Computing Platforms</td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

* 6 hours, distributed over at most two topics (preferably all into one)
Description of Knowledge Units

SPD/Introduction → Common Aspects/Shared Concerns

This unit aims to develop core concepts relating to specialized platform development. Students shall recognize various specialized platform development and their corresponding applications, programming languages used for these particular and modern applications, and how to use them effectively.

- **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 APIs
  - **c.** Overview of Platform Languages (e.g., Kotlin, Swift, Objective C, Python, C#, C++, Java, JavaScript, HTML5)
  - **d.** Programming under platform constraints (e.g. available development tools, development)

- **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, considerations through web platforms, privacy considerations, architecture, and storage solutions.

- **Topics**
  - **a.** Web programming languages (e.g., HTML5, JavaScript, PHP, CSS)
  - **b.** Web platform constraints
  - **c.** Software as a Service (SaaS)
  - **d.** Web standards
    - i. Security
      - Certificates (TLS)
  - **e.** Computing services (e.g., Amazon AWS, Microsoft Azure)
    - i. Cloud Hosting
    - ii. Scalability (e.g. Autoscaling, Clusters)
  - **f.** Data management
    - i. Privacy etc
  - **g.** Architecture
    - i. Monoliths vs. Microservices
ii. Micro-frontends  
iii. Event-Driven vs. RESTful  

h. Storage Solutions  
i. Relational Databases  
ii. NoSQL databases  

**Illustrative Learning Outcomes**  
a. Design and Implement, using the microservice architecture design, a simple web application.  
b. Describe the web constraints and opportunities such as hosting, services, and scalability that developers shall be considering.  
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. [Assessment]  

**SPD/Mobile Platforms**  
This unit aims to develop concepts relating to web platform technologies and considerations.  

**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  
iii. Network and the Web  
d. Mobile computing affordances  
i. Location-aware applications  
ii. Sensor-driven computing (e.g., gyroscope, accelerometer, health data from the watch)  
iii. Telephony, Instant messaging  
iv. Augmented Reality  
e. Emerging technologies
f. Testing

g. Asynchronous computing
   i. How it differs from traditional synchronous programming
   ii. Handling success via callbacks
   iii. Handling errors asynchronously
   iv. Testing asynchronous code

• Illustrative Learning Outcomes
  a. Design and implement a location-aware mobile application that uses data APIs.
  b. Design and implement a sensor-driven mobile application that logs data on a server.
  c. Design and 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 aims to consider topics related to deployment of software on existing robot platforms and the application of these robots to specific problems. Concepts include robotic platforms, specialized programming languages and 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 (i.e., the Robot Operating System (ROS)) and its architecture
     ii. Forward kinematics
     iii. Inverse kinematics
     iv. Dynamics
     v. Navigation and robotic path planning
     vi. Manipulation and grasping

• 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 general-purpose programming languages and 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 that a given robotic platform imposes on developers
**SPD/Embedded Platforms**

Knowledge unit recognizes the impact on embedded platforms and their applications. Embedded platforms are an extensive specialized area that spans from sensor technology to applications of ubiquitous computing.

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

- Topics
     i. RT vs non-RT
     ii. Resource constraints (eg memory profiles etc)
  b. Sensors and Actuators.
  c. Embedded processors and architectures.
  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, a drone, etc)
  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

**SPD/Game Platforms**

The Knowledge unit aims to recognize concepts related to game platforms and computing technology that make this unit unique. Game design can enhance problem-solving in computer science by using principles of game design to engage the transferability of concepts in problem solving.

- Topics
  a. Types of Game Platforms, such as Personal Computer (PC), Home Console, VR, Mobile, AR, Mixed Reality, Web
  b. Game and visual programming platform Languages, such as Lua, Unreal/Unity, Custom Game Engine Development
  c. Game Platform Constraints
     i. Code Optimization
     ii. Network/Multiplayer requirements
     iii. GPU optimization
     iv. Port/platform optimization
  d. Coding for alternate game controllers/users with distinct needs
     i. Accessibility hardware (such as XAC)
     ii. Accessibility software/options
iii. Game-specific controller - custom-built/3D-printed projects such as skateboard, lightgun, fishing rod
e. Games User Research (UX/UI/UR and user testing for games)
f. Simulations
   i. Mapping to a system based on data or projections
   ii. Modeling specific systems vs abstract systems
g. Serious Games
   i. Comply with well-known standards and industry compliance.

- Illustrative Learning Outcomes
  a. Implement a simple application on a game platform
  b. Describe the constraints that game platforms impose on developers
  c. Compare game programming with general-purpose programming
  d. Consider real world attributes (such as Physics, behaviours) to develop a model for a simulation
  e. Recognize the limitations of simulations based on model constraints
  f. Recognize ethical responsibilities that comply with relevant industry standards
  g. Understand how user research apply to games

**SPD/Interactive Computing Platforms**

Knowledge unit recognizing the impact on interactive computing platforms. Emerging computing areas such as Data Science, Quantum Computing and various creative disciplines explore this knowledge unit. This unit aims to explore concepts that impact interdisciplinary areas in computing often known as CS+X.

- Topics
  a. Data Analysis Platforms
     i. Jupyter notebooks; Google Colab; R
     ii. Cloud data analytics (BigQuery), Apache Spark, generic “large data”
     iii. Prologs/Datalogs, other SQL-type exploratory analysis
  b. Data Visualizations
  c. Creative coding
     i. Creative interactive frameworks (can x-over with web, embedded/IoT/other lo-fi hardware)
        - Live Music
        - Generative Art
        - Exhibition/demonstrative works
     ii. Machine-assisted interactivity
        - AI/ML “pairing”
     iii. Reactive/FRP paradigms
  d. Quantum Computing Platforms
     i. Simulating real time quantum circuits
     ii. Qiskit; Quantum Development Kit; Cirq

- Supporting math studies
  a. Signal analysis / Fourier analysis / Signal processing
  b. Process control (PID)
c. Vector calculus
d. Linear algebra

- Supporting humanities studies
  a. Visual art
  b. Journalism
  c. Music theory, composition

- Illustrative Learning Outcomes
  a. Interactively analyze large datasets using a read, evaluate, and print loop (repl)
  b. Use a ML algorithm that can play a backing track (e.g. with live coding)
  c. Demonstrate simple logical gates for quantum primitives
  d. Create compelling computational notebooks that construct a narrative for a given journalistic
  e. Implement interactive, exploratory graphics for a dataset
  f. Implement an algorithm that interacts with a human without using a screen
  g. Contextualize the attributes of different styles of data analysis such as interactive vs engineered pipeline
  h. Write an algorithm using a notebook computing platform (e.g., quantum, graph, sorting algorithms)

List of Professional Dispositions Appropriate for this KA
- Learning to learn (new platforms, languages)
- Inventiveness (in designing software architecture within non-traditional constraints)
- Adaptability (to new constraints)
- Learning to Debug & Test your code

Math needed and wanted
- Calculus
- Linear Algebra
- Probability/Statistics (e.g., dynamic systems, visualization)
- Discrete Math/Structures (e.g., graphs for process control and path search)

Crosscutting and Overlapping topics
- Artificial Intelligence
- Graphics
- Human Computer Interaction
- Modeling
- Programming Languages
- Software Engineering
Subcommittee
Chair: Christian Servin (El Paso Community College, El Paso, TX, USA)

Subcommittee members:

- Amruth N. Kumar, Ramapo College of New Jersey, Mahwah, NJ, USA
- Sherif G. Aly, The American University in Cairo
- Yoonsik Cheon, The University of Texas at El Paso, El Paso, Texas, U.S.A.
- Eric Eaton, University of Pennsylvania
- Orlando Gordillo, NASA, USA
- Claudia L. Guevara, Jochen Schweizer mydays Holding GmbH, Munich, Germany
- Rory K. Summerley, Falmouth University, Penryn, Cornwall, UK.
- R. Tyler Pirtle, Google