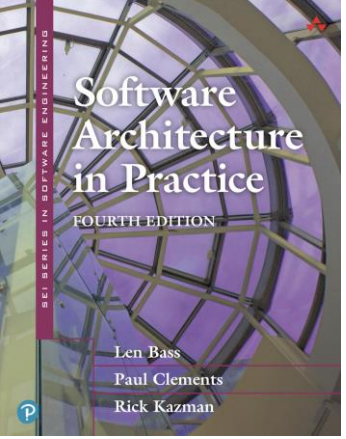
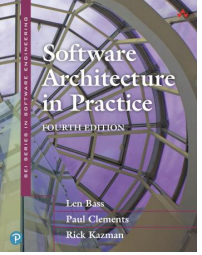


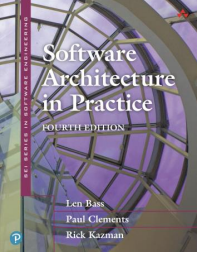
Chapter 18: Mobile Systems

The telephone will be used to inform people that a telegram has been sent.
—Alexander Graham Bell



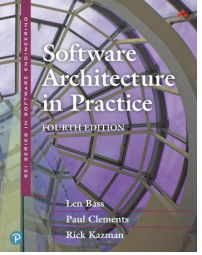
Chapter Outline

- Energy
- Network Connectivity
- Sensors and Actuators
- Resources
- Life Cycle
- Summary



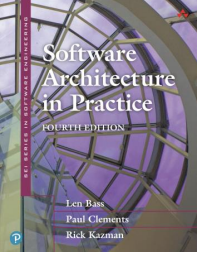
Mobile System Characteristics

- A mobile system has the ability to be in movement while continuing to deliver some or all of its functionality.
- Its driving characteristics are different than those of fixed systems. The most important ones are: energy, network connectivity, sensors and actuators, resources, and life cycle.



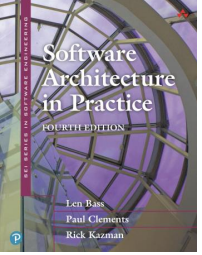
Energy

- Battery-powered mobile systems include a component, that interacts with the BMS (Battery Management System)
- Two characteristics of batteries change as they age: the maximum battery capacity and the maximum sustained current.
- An architect must allow for managing consumption given available power so that the device performs at an acceptable level.
- Of course, the battery manager itself utilizes resources—memory and CPU time. The amount of CPU time consumed by the battery manager can be managed by adjusting the query interval.



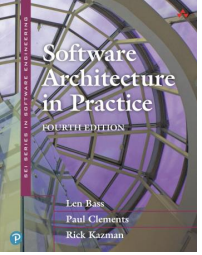
Throttling Energy Usage

- Energy usage can be reduced by terminating or degrading portions of the system (the throttle usage tactic described in Chapter 6).
- This can be accomplished by:
  - reducing the brightness or the refresh rate of the display on a smartphone,
  - reducing the number of active cores of the processor,
  - reducing the clock rate of the cores,
  - reducing the frequency of sensor readings.



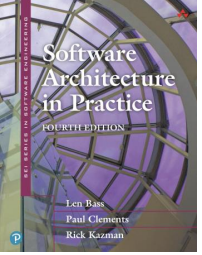
Throttling Energy Usage

- Mobile systems should gracefully tolerate power failures and restarts.
- Example hardware requirements:
  - The system's computer does not suffer permanent damage if power is cut at any time.
  - The system's computer (re)starts the OS whenever sufficient power is provided.
  - The system's OS has the software scheduled to launch as soon as it is ready.
- Example software requirements:
  - The runtime environment can be killed at any moment without affecting the integrity of the binaries, configurations, and operational data in permanent storage, and while keeping state consistent after a restart.
  - Applications need a strategy to deal with data that arrives while the application is inoperative.
  - The runtime can start after a failure so that the startup time is less than a specified period.



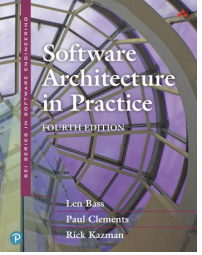
Network Connectivity

- Wireless networks are categorized based on the distance over which they operate.
  - *Within 4 centimeters*. Near Field Communication (NFC) is used for keycards and contactless payment systems.
  - *Within 10 meters*. The IEEE 802.15 family of standards covers this distance. Bluetooth and Zigbee are common protocols.
  - *Within 100 meters*. The IEEE 802.11 family of standards (Wi-Fi) is used within this distance.
  - *Within several kilometers*. The IEEE 802.16 standards cover this distance. WiMAX is the commercial name for these standards.
  - *More than several kilometers*. This is achieved by cellular or satellite communication.



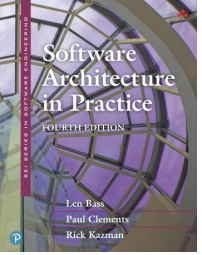
Network Connectivity

- Designing for communication and network connectivity requires the architect to balance a large number of concerns, including:
  - *Number of communication interfaces to support*. Only the strictly required interfaces should be included to optimize power consumption, heat generation, and space allocation.
  - *Movement from one protocol to another*. The architect must account for the possibility that the mobile system may move from an environment that supports one protocol to an environment that supports another protocol. Such transitions should be seamless to the user.
  - *Choosing the appropriate protocol dynamically*. If multiple protocols are simultaneously available, the system should choose a protocol based on factors such as cost, bandwidth, and power consumption.
  - *Modifiability*. Given the large number of protocols and their rapid evolution, the system should be designed to support changes or replacements in the elements of the system involved in communication.
  - *Bandwidth*. The information to be communicated to other systems should be analyzed for distance, volume, and latency requirements so that appropriate architectural choices can be made.
  - *Intermittent/limited/no connectivity*. Communication may be lost while the device is in motion. The system should be designed so that data integrity is maintained, and computation can be resumed when connectivity returns. The system should be designed to deal gracefully with limited connectivity or even no connectivity.
  - *Security*. Mobile devices are particularly vulnerable to spoofing, eavesdropping, and man- in-the-middle attacks, so responding to such attacks should be part of the architect's concerns.



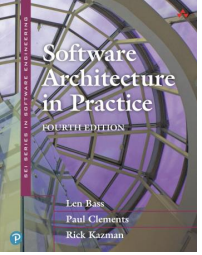
Sensors and Actuators

- A *sensor* is a device that detects the physical characteristics of its environment and translates those characteristics into an electronic representation.
- A *sensor hub* is a coprocessor that helps integrate data from different sensors and process it. A sensor hub can help offload these jobs from a product's main CPU, thereby saving battery consumption and improving performance.
- An *actuator* is the reverse of a sensor: It takes a digital representation as input and causes some action in the environment.



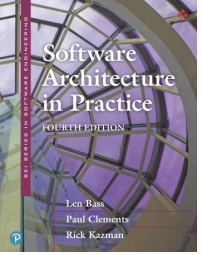
Sensors and Actuators

- An architect has several concerns with respect to sensors:
  - How to create an accurate representation of the environment based on the sensor inputs.
  - How the system should respond to that representation of the environment.
  - Security and privacy of the sensor data and actuator commands.
  - Degraded operation. If sensors fail or become unreadable, the system should enter a degraded mode.



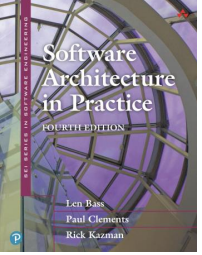
Resources

- The tradeoff in the choice of resources is between the contribution of the resource and its volume, weight, and cost.
- Costs include both the manufacturing costs and nonrecurring engineering costs.
- Volume, weight, and cost constraints may be given by the marketing department and by physical considerations of device use.
- The physical considerations for the device's use depend on both human and usage factors.



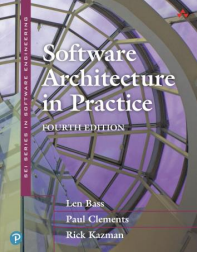
Resources

- Other constraints on mobile resources include:
  - *Safety considerations*. Physical resources that have safety consequences must not fail or must have backups. Backup processors, networks, or sensors add cost and weight, as well as consume space.
  - *Thermal limits*. Heat can be generated by the system itself which can have a detrimental effect on the system's performance, even to the point of inducing failure.
  - *Other environmental concerns*. Other concerns include exposure to adverse conditions such as moisture or dust, or being dropped.



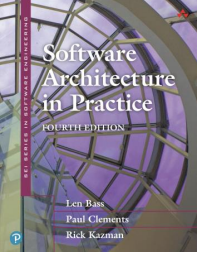
Resources

- An architect must make many decisions on resources and their usage:
  - *Assigning tasks to electronic control units (ECUs).* This decision can be based on a number of factors including: criticality, fit of the ECU to the function, locality of communication, and connectivity.
  - *Offloading functionality to the cloud.* Is there sufficient power for specific functions, or adequate connectivity to offload functions to the cloud?
  - *Shutting down functions depending on the mode of operations.* Subsystems that are not being used can scale down their footprint, allowing other subsystems to access more resources.
  - *Strategy for displaying information.* Different display options are possible, based on the screen size/resolution.



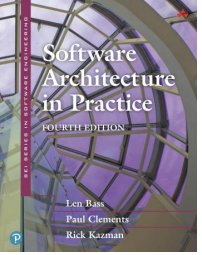
Lifecycle

- The life cycle of mobile systems tends to feature some idiosyncrasies that an architect needs to take into account:
  - hardware first
  - testing
  - deploying updates
  - logging



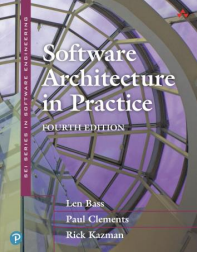
Hardware First

- For many mobile systems, the hardware is chosen before the software is designed.
- The software architecture must live with the constraints imposed by the chosen hardware.
- The best approach for a software architect is to actively drive early discussions, emphasizing the tradeoffs.



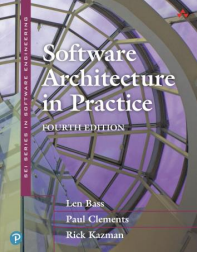
Testing

- Mobile devices present some unique considerations for testing:
  - *Test display layouts*. Smartphones and tablets come in a wide variety of shapes, sizes, and aspect ratios. Verifying the correctness of the layout on all of these devices is complicated.
  - *Test operational edge cases*. An application should survive battery exhaustion and system shutdown, preserving state. Also, the user interface typically operates asynchronously. When the user interface does not react correctly, re-creating the sequence of events that caused the problem is difficult.
  - *Test resource usage*. Some vendors will make simulators of their devices available to software architects. But testing battery usage with a simulator is problematic.
  - *Test for network transitions*. Ensuring that the system makes the best choice when multiple communication networks are available is also difficult.



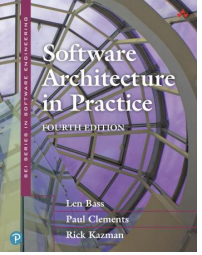
Testing Example

- Suppose we are testing a car's lane keep assist function. Testing of this system may address the following levels:
  - *Software component*. A lane detection component will be tested via unit and end-to-end testing, with the aim of validating the software's stability and correctness.
  - *Function*. The next step is to run the software component together with other components of the lane keep assist function, such as a mapping component to identify highway exits, in a simulated environment.
  - *Device*. The bundled lane keep assist function, needs to be deployed on its target ECU and tested there for performance and stability.
  - *System*. Finally, all devices with all functions and components are tested in a lab and then in a prototype. This is to confirm that the integrated subsystems work together and deliver the desired functionality and system quality attributes.



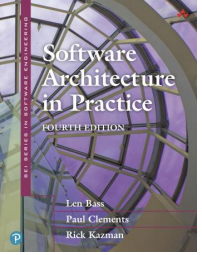
Deploying Updates

- Updates to the system may target the software, the data, or (less often) the hardware. The following issues relate to deploying updates:
  - *Maintaining data consistency*. For consumer devices, upgrades tend to be automatic and one-way (there's no way to roll back to an earlier version). This suggests keeping data on the cloud is a good idea.
  - *Safety*. The architect needs to determine which states of the system can safely support an update. This implies that the system needs to be aware of safety-relevant states with respect to updates.
  - *Partial system deployment*. Re-deploying an application or major subsystem consumes both bandwidth and time. The application should be architected so that the portions that change frequently can be easily updated. This calls for a specific type of modifiability (see Chapter 8) and an attention to deployability (see Chapter 5).
  - *Extendability*. Mobile vehicle systems tend to have long lifetimes. Retrofitting cars, trains, airplanes, satellites, and so forth will likely become necessary at some point. Retrofitting means adding new technology to old systems, either by replacement or addition.



Logging

- Logs are critical when investigating and resolving incidents that have occurred or may occur.
- In mobile systems, the logs should be offloaded to a location where they are accessible. This is useful not only for incident handling, but also for performing analyses on the usage of the system.
- Many applications do something similar when they encounter a problem and ask for permission to send the details to the vendor.
- For mobile systems, this logging capability is particularly important.



Summary

- Mobile systems span a broad range of forms and applications, from smartphones and tablets to vehicles such as automobiles and aircraft.
- We have categorized the differences between mobile systems and fixed systems as being based on five key characteristics: energy, connectivity, sensors, resources, and life cycle.