Carnegie Mellon

Software Architectures & Design

Course Introduction and Overview

Instructor: Nguyễn Đức Mận
0904 235 945
mannd@duytan.edu.vn

Nội dung được xây dựng dựa trên tài liệu của Prof. Matthew Bass 2018 - ISR

Introductions - 1

Matthew Bass

- **Currently**
  - Teaching Professor for Carnegie Mellon in SE Department
  - Consultant
- **Previously**
  - Siemens: Member of the Software Architecture Group at Siemens Corporate Research
  - SEI: Resident affiliate at Software Engineering Institute
  - SEI: Member of the technical staff
  - 15+ years experience as an architect and software engineer
  - Worked in with a range of systems in Medical, Building Automation, Automotive domains among others

Introductions - 2

Duc-Man NGUYEN

- **Currently**
  - Dean, International School
  - Instructor
- **Previously**
  - Dean, IT Department
  - Vice Dean, IT Department
  - Head of SE, MIS Division
  - Team lead HSD, Duy Tan Soft (CSE)
  - 20+ years experience as an instructor and software engineering reseacher
  - Research fields: DB, project management, Software Engineering, Software testing

Course Objectives

- **The Software Architecture course is designed to:**
  - familiarize participants with software architecture concepts and principles
  - familiarize participants with the software architectures in an organizational context
  - introduce participants to the factors that drive architectural design
  - teach attendees approaches for eliciting and specifying architecturally significant requirements
  - introduce participants to the activities involved in designing and validating a software architecture

Course Outcomes

- **Participants will have a better understanding of**
  - the relationships between system qualities and software architectures
  - what drives software architecture
  - approaches for eliciting architectural drivers
  - how to plan architectural activities
  - how design and validate software architectures

Contact Information

Matthew Bass

mbass@cmu.edu

Masters of Software Engineering Program

www.mse.cs.cmu.edu

Duc-Man Nguyen

mannd@duytan.edu.vn

International School, DTU

http://is.duytan.edu.vn

Agenda

- **Motivating examples**
- **Architectural alignment**
- **What is software architecture**
- **Course objectives**

Agenda

- **Motivating examples**
- **Architectural alignment**
- **What is software architecture**
- **Course objectives**

Motivation

Let's start with a story …

The Cell Phone Industry

US & European Market

- **In the "old" days**
  - relatively few people had cell phones
  - phones were very expensive
  - phones primarily only made phone calls
- **Now, however,**
  - Market essentially saturated
    - 75% of the population has a cell phone
  - People upgrade cell phones based on innovative features or for prestige
  - Market leaders need to produce many product variants in order to maintain market share

Business Model

- **When an innovative phone is introduced it is expensive**
  - Anyone remember what a Motorola Razr first cost?
- **As soon as a competitor is released to the market the price drops drastically**
  - What can you get a Razr for now?
- **Much of the profit comes from the initial "expensive" window**
- **In order to maintain market share and profitability companies must release many products with innovative features/designs to the market**

So What?

What does this say about the cell phone software?

What properties used to be important to be able to achieve objectives?

What properties are now important?

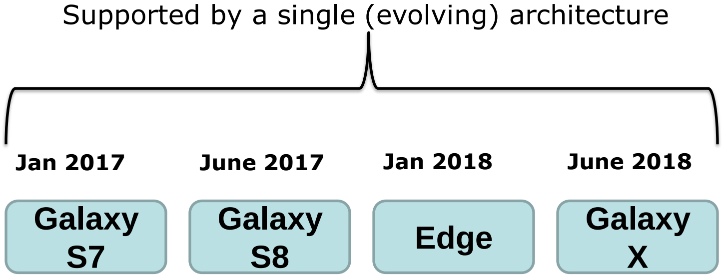
If Not?

What happens if the software doesn't have these properties?

- There once was a cell phone manufacturer that had the 3rd largest market share
- They were known for quality products
- … but there software didn't have certain properties
  - It took them a long time to develop a new product
  - Couldn't easily adapt the software or add new features)
- As they tried to adapt, the quality of the products suffered
  - They released a phone that caused hearing damage
- They lost market share
- They determined that their software wasn't suitable for the current realities of the market and they sold the cell phone division

Sample Product Roadmap

Supported by a single (evolving) architecture



Each release has:

- Target market, Needed capabilities, Expected features, Anticipated battery life
- Expected BOM, Anticipated price, Expected # of units, Desired profit margin
- …

Product Roadmap & Architecture

- **What does it take to go from product 1 to product 2?**
- **We might need to:**
  - Change the underlying hardware
  - Add features
  - Remove features
  - Change UI
  - Change things like battery life, response time, ect
- **We have a limited time and budget to successfully accomplish this**
- **Otherwise**
  - We will not achieve the product roadmap
  - We will not make release dates
  - R & D costs will increase (and thus margins will decrease)
  - …

There Are Many Other Examples

- **Amazon**
  - For every 100 ms decrease in latency Amazon's sales increase by 1%
- **Google**
  - Google's loading time decreasing from .9 seconds to .4 seconds increases ad revenue by 20%
  - Search results in a particular shade of blue yields a 16 times greater click through rate than the next best color at Google
- **Netflix**
  - Video stream loading time and buffering impacts users rate of abandonment
- **From airplanes to medical devices the structure of the systems (including software) have an impact on the business**

What's the Message Here?

There is a relationship between the business goals of the organization and *some* aspect of the system

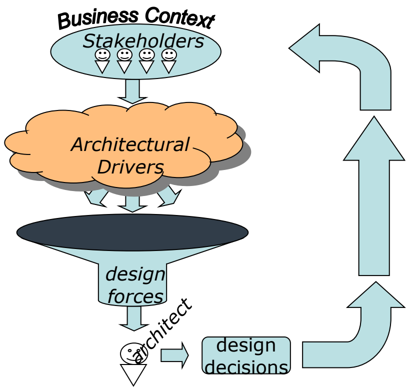
- In the case of the cell phone manufacturer the effort required to create a product variant was important
- In the case of Google response time drastically effects their gross revenue
- Netflix ability to serve and retain customers is dependent on their throughput
- With Amazon the systemic properties (e.g. performance/scalability) impacts their value as an organization

While functionality is important these properties were also vital for achieving the objectives

Agenda

- **Motivating examples**
- **Architectural alignment**
- **What is software architecture**
- **Course objectives**

Architectural Alignment

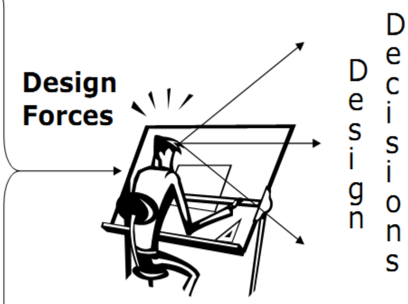


What Inspires Design?

Architectures are the result of design choices…

Architectural design influences include:

- long term business goals
- immediate business needs
- cost schedule
- organizational structure, talent, and resources
- marketing strategies
- affordable/available technology
- functional/behavioral requirements from stakeholders
- the architects experiences



A better question is, "what are design choices based on," or… "what influences architects?"

Architectural Alignment

- **When the properties needed to support the business context are promoted we have "*architectural alignment*"**
- **The "structure" of the system support these properties**
- **How does this alignment occur?**
  - First we need to recognize the architectural drivers
  - We need to make decisions that support the architectural drivers (when possible)
  - We need to understand how the business is impacted when technical tradeoffs are made
  - We (may) need to refine the business case accordingly

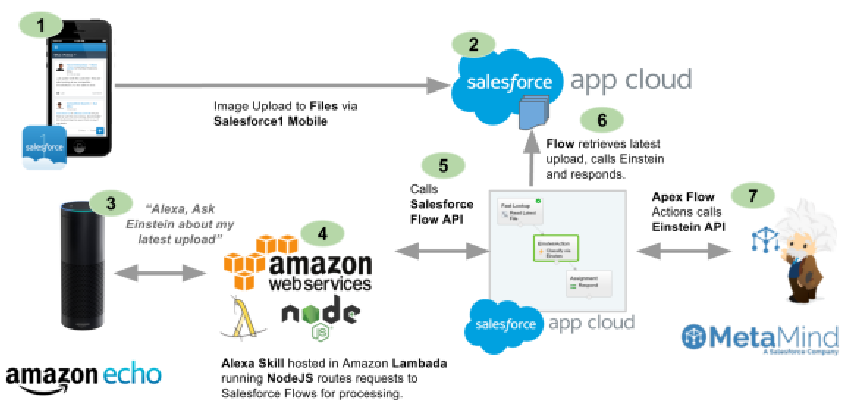
Recognizing Architectural Drivers

- **Given the business context we need to articulate the** *architecturally significant requirements*
- This requires three things
  - First, that we know how to adequately understand the business context
  - Second, that we recognize the architecturally significant requirements implied by the business context
  - Third, that we can articulate these requirements in a way that is understandable and actionable

Business Context

- **We typically look at the business processes that will be automated**
  - Think about what the business analyst gives you as input to the design activity
- **These are part of the business context, but not all of it**
  - We often ignore (or consider as an afterthought) the quality attributes implied by the organizational objectives
  - We aren't sure how to account for the strategic objectives of the organization
  - Other factors such as support for legacy systems, changing the organization structure, and so forth have an impact

Business Context



Recognizing the Architectural Drivers

- **Typical requirements elicitation doesn't explicitly identify architectural drivers**
  - We usually focus on functionality
  - If we do focus on other properties we often articulate them ambiguously
- **Why is this?**
  - We don't understand what drives software architecture
  - The primary purpose of the system is to automate portions of the business
  - We don't speak the same language (even if we think we do)

Articulating Architectural Drivers

- **Once the drivers are identified we often articulate them ambiguously**
  - E.g. "the system must be modifiable" or "the system must be reliable"
- **It is left to the engineer to determine exactly what that means**
  - The engineer has to determine what aspect of the system is meant to change
  - The engineer also needs to determine "how modifiable" the system needs to be
- **The decisions the engineer makes will impact the business**

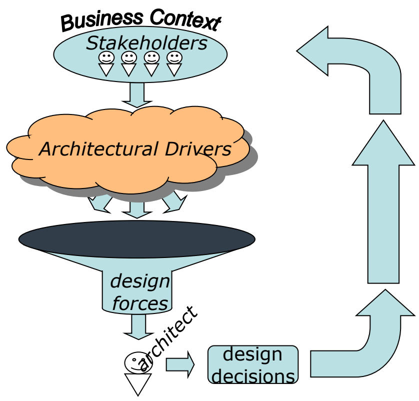
Making Design Decisions

- **Once the architectural drivers are identified the architect must make decisions that support them**
- **What makes this difficult?**
  - Historically design approaches are driven by functionality
    - Structured design
    - Object Oriented Design
  - Lack of structured architectural design approaches
    - Structured methods are a recent development
    - Formal training has only recently become available
  - Architects typically architect by instinct

Architecting By Instinct

- **Architects are typically experienced developers**
- **They often work in the same domain for along time**
- **Over time their experience results in a set of intuition**
  - They intuitively know the requirements
  - They intuitively know what solutions work and which ones don’t
- **They often don’t explicitly understand how this intuition developed**
  - But they rely on it

Tradeoffs are Inevitable



Scoping

- **In the absence of engineering constraints the system could do anything**
  - And marketing usually starts by asking for everything …
- **A typical scoping process involves some kind of effort (cost) estimation**
  - This is typically driven by the desired features
- **Estimation models such as Function Point or Use Case estimates are driven (mostly) by function**

Architectural Tradeoffs

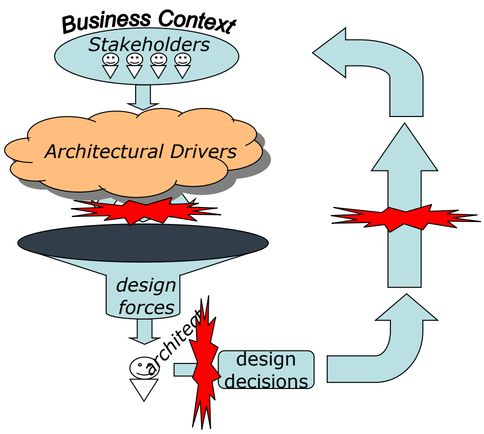
- **In addition to the trading off cost vs. schedule vs. quality there are architectural tradeoffs e.g.**
  - In order to increase modifiability you decouple elements, this typically increases response time
  - One way to increase performance is to cache data, this makes synchronization more difficult for redundant elements
- **Making these architectural tradeoffs impact the business context**

“We can’t give you 1 second page loading time in a web based environment if you want panning and zooming capability …”

Refining the Business Context

- **These tradeoffs are typically discovered as part of the architecting process**
- **Maintaining alignment often requires refining the business context**
- **This means that you:**
  - Understand the relationship between architectural decisions and the business context
  - Know what technical options you have
  - Understand the impact of each of these options
  - Can communicate this impact in a way that the people responsible for business decisions can understand
  - The set of business people needed can refine the business context accordingly

Areas of Breakdown



Agenda

- Motivating examples
- Architectural alignment
- What is software architecture
- Course objectives

Software Architecture Defined

"*The software architecture of a program or computing system is the structure or structures of the system, which comprise the software elements, the externally visible properties of those elements, and the relationships among them."**

*Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003.

Let's Parse the Definition...

"*The software architecture of a program or computing system is the structure or structures of the system, which comprise the software elements, the externally visible properties of those elements, and the relationships among them."**

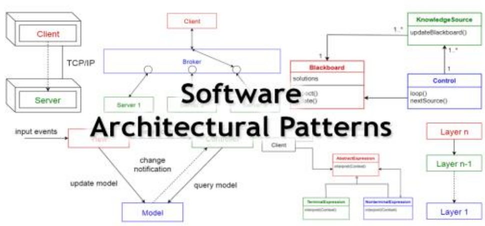
*Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice, Second Edition.* Boston, MA: Addison-Wesley, 2003.

Structures

- **We mentioned that all software systems are multiple structures**
- **“Systems” are made of many different structures e.g.**
  - Buildings have HVAC, wiring, plumbing, supporting structures and so forth
  - Humans have cardiovascular, skeletal, muscular structures, and so forth
- **The nature of these structures impact the behavior or properties of the overall system**
- **The behavior of the system can be analyzed by looking at the relevant structure(s)**

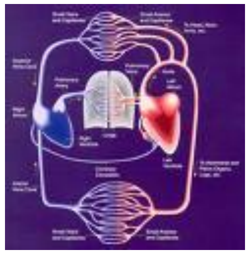
Structures and Views

- **We will use the terms *structure* and *view* when discussing architecture:**
  - ***structure*** – an actual set of architectural elements as they exist in software or hardware
  - ***view*** – a representation of a coherent set of architectural elements, as written and read by system stakeholders. A view represents a set of elements and the relationships among them.



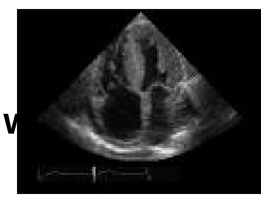
Human “System”

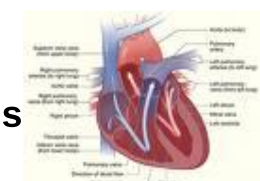
- **The human body has many structures**



Views of the Human System

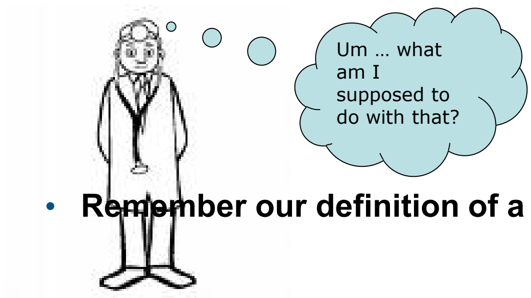
- **Each “structure” may have multiple views**
  - Each “view” highlights different information





Views Highlight Different Aspects of System

- **If you are a cardiologist you are only interested in certain information**



- **Remember our definition of a "good**

Example Views

- **An EKG will show you a *dynamic* representation of the electrical impulses of the heart**
  - This may give you some indication of the sinus node on the heart (responsible for emitting electrical impulses)
  - It won't tell you anything, however, about how the blood is flowing through the heart
- **An Ultrasound will show you a *dynamic* view of the blood flowing through the heart**
  - This can be used to identify chamber defects or pumping issues

Software Structures

- **The same holds true for software**
- **In software there are three classes of structures**
  - Static structures – we call these “Modular Structures”
  - Dynamic structures – we call these “Component and Connectors”
  - A mapping of software elements (dynamic or static) to non-software elements – we call these “Allocation Structures”

Static Structures

- **A software system exists in many forms prior to executing on a processor**
- **These are code elements of one kind or another that are related in different ways e.g.**
  - Decomposition structure
  - Uses structure
  - Generalization
- **Static views can be used for things like:**
  - Analyzing modifiability
  - Estimation
  - Resource planning
  - …

Dynamic Structures

- **Once code is compiled and executing, however, it no longer exists in the same form**
- **We call the structure of runtime elements Dynamic Structures e.g.**
  - Process structures
  - Concurrency structures
  - Shared data structures
- **Dynamic views can be used for**
  - Performance analysis
  - Fault tolerance
  - Security
  - …

Allocation Structures

- **Software does not exist independent of it's environment**
- **Allocation views show the mapping of software elements to the environment in which it lives**
  - Deployment – processes to processors
  - Work break down – code modules to people or teams
  - Implementation – code modules to file structure
- **These are used for things like:**
  - Work assignment
  - Performance
  - Fault-tolerance
  - …

Why Do We Care?

- **Managers need architectural information to do certain things e.g.**
  - Estimation
  - Work break down
  - …
- **It is helpful to know where and how to get and understand this information**
- **We will look at this in more detail as we progress through the course**

Note: we will only be looking at relevant aspects

Back To The Definition...

"The software architecture of a program or computing system is the structure or structures of the system, which comprise the software elements, the externally visible properties of those elements, and the relationships among them."

What Are Software Elements?

- **Elements are the course grained "parts" of the system.**
- **The type of element under consideration depends upon the perspective taken.**
  - static perspectives: elements = classes, files,...
  - runtime perspectives: elements = threads, data flow, control...
  - physical perspectives: elements = computers, actuators, networks...
- **Views are often more complex than this – *we will discuss them in detail later…***

Back To The Definition...

"The software architecture of a program or computing system is the structure or structures of the system, which comprise the software elements, the externally visible properties of those elements, and the relationships among them."

What Are Externally Visible Properties?

- **Externally visible properties are those assumptions that one element can make about another element.**
  - Examples include:
    - services and data that an element requires and provides
    - performance characteristics
    - fault handling
    - resources elements share, and so forth..

What Are Externally Visible Properties?

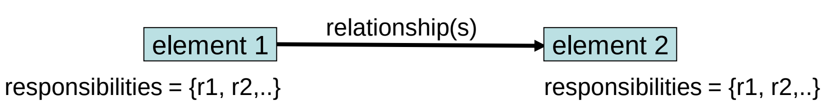
- **Assigning responsibilities to elements is an essential part of architecture design.**
  - Allows architects to reason about the system's overall properties.
- **The kinds of properties that are visible depend upon the view of the system.**
  - performance → runtime view
  - modifiability → static view

Back To The Definition...

"The software architecture of a program or computing system is the structure or structures of the system, which comprise the software elements, the externally visible properties of those elements, and the relationships among them."

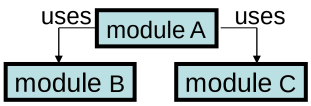
What Are The Relationships?

- **Elements interact with each other via interfaces that partition details into public and private parts.**
  - These interactions form relationships between architectural elements.
  - Architecture is concerned with the public side of this partitioning.

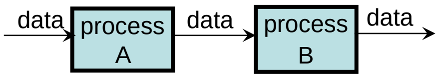


What Are The Relationships?

- **Since the elements and their visible properties are dependent upon the structures of the system, the relationships among elements will depend upon these structures as well.**



Static View
- elements: modules
- relations: uses



Runtime View
- elements: processes
- relations: dataflow

What Kind of Systems Have Software Architectures?

- **Every *software intensive* system *has* an architectural design whether deliberately designed or not! Here is why:**
  - All systems have elements and relationships – even if a system has a single element, related to itself.
  - Just having an architecture is different from having one that is codified and understood.
  - If you don't design the architecture, you will get one anyway –*you might not like what you get!*

Agenda

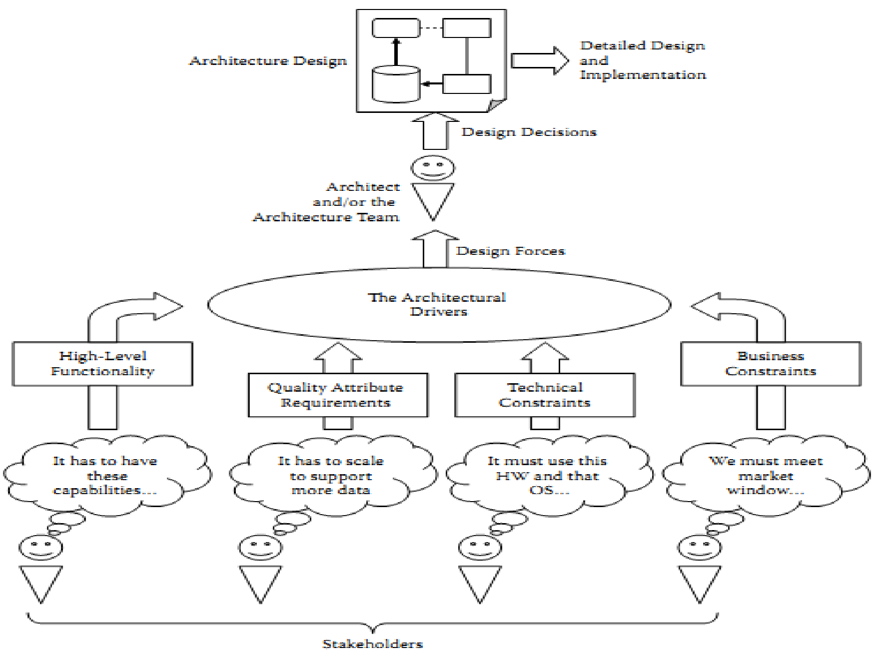
- **Motivating examples**
- **Architectural alignment**
- **What is software architecture**
- **Course objectives**

This Course

- **We're going to introduce an approach for maintaining alignment**
- **This means we need to ensure traceability between:**
  - Business context,
  - Architecturally significant requirements, and
  - Design decisions

Maintaining Alignment

- **Additionally refinements need to be made**
- **As we discover issues based on the engineering limitations we need to**
  - Determine what the alternatives are
  - Identify the impact on the business context of the alternatives
  - Determine which makes most sense from a business perspective
  - Refine the business context accordingly



Course Objective

In short that is what this course is all about

Course Topics

- **First we are going to synchronize on terms**
  - We are going to talk about what *we mean* by software architecture
  - Definitions abound (and are constantly changing) so we are just going to worry about what we mean for the purposes of this course
  - We will define what we mean by business context

Course Topics II

- **We will then look at what *drives* software architecture**
- **We will do this by looking at:**
  - The kind of requirements that have an impact on the architecture
  - The portion of the business context that is relevant to these requirements

Course Topics III

- **We will then look at the architectural activities including:**
  - Documentation
  - Design activities
  - Evaluation
- **Once we understand all of this we can talk more concretely about planning architectural activities**
- **Time permitting we will talk about additional topics such as Software Product Lines, and Global Software Development**

Grades

Assessment Type	Grade Percentile
Quizzes (2 Quizzes)	10%
Homework (2 Hws)	5%
Labs (2 labs)	15%
Midterm Exam	15%
Group Project	20%
Final Exam	35%
Total:	100%

Course Learning Outcomes

- CLO.1. *Identify* the technical, organizational, and business role of software architecture
- CLO.2. *Describe* the architectural drivers upon guidance for eliciting and analyzing the business requirements, system and software requirements.
- CLO.3. *Apply* the guidance for designing and documenting architectural designs
- CLO.4. *Evaluate* the architecture of the system by ATAM and/or ACDM
- CLO.5. *Practice* how to transition architecture into software organizations
  - CLO.6. *Identify* and acquire needed information.
  - CLO.7. *State* clearly all appropriate items in report.
  - CLO.8. *Use* electronic/multimedia communication effectively.
  - CLO.9. *Demonstrate* effectively as a team member.

Course Group Project

- Apply the Architectural Process to Discover RE, Design the SA, Evaluation the Architecture and Documentation.
- Use ArchitectureStudio / Archictural tools/ Draw tool to draw the architecture
- Topics
- Team of 4-5 students
- 8 weeks

Summary

- **Today we**
  - Talked about why managers need to know something about software architecture
    - Many management activities are interdependent with technical activities
    - Unless there is a common means for communication these activities may diverge
    - Without alignment of these activities the organizations goals are not likely going to be met
  - Scoped the focus of the course
    - We are talking primarily about product focused organizations
    - The roles we will be focusing on are product and project managers

Summary II

- **We Discussed the responsibilities of product and project managers**
  - Product managers are the “product owners” they are responsible for the strategic direction of the product throughout the product lifecycle
  - Project managers are responsible for the project development lifecycle
    - They worry about cost, quality, and schedule

QUESTIONS?

References

1. **Windley, P.,** ***The Discipline of Product Management*****, 2002** [**www.utah.gov**](http://www.utah.gov)
2. **Carole Hedden,** ***Marketing Tools***
3. **Bass et al. 2013,** ***Software Architecture in Practice 3rd edition***
4. ***Video for review term*** ***https://youtu.be/qailvg7Z3jQ***