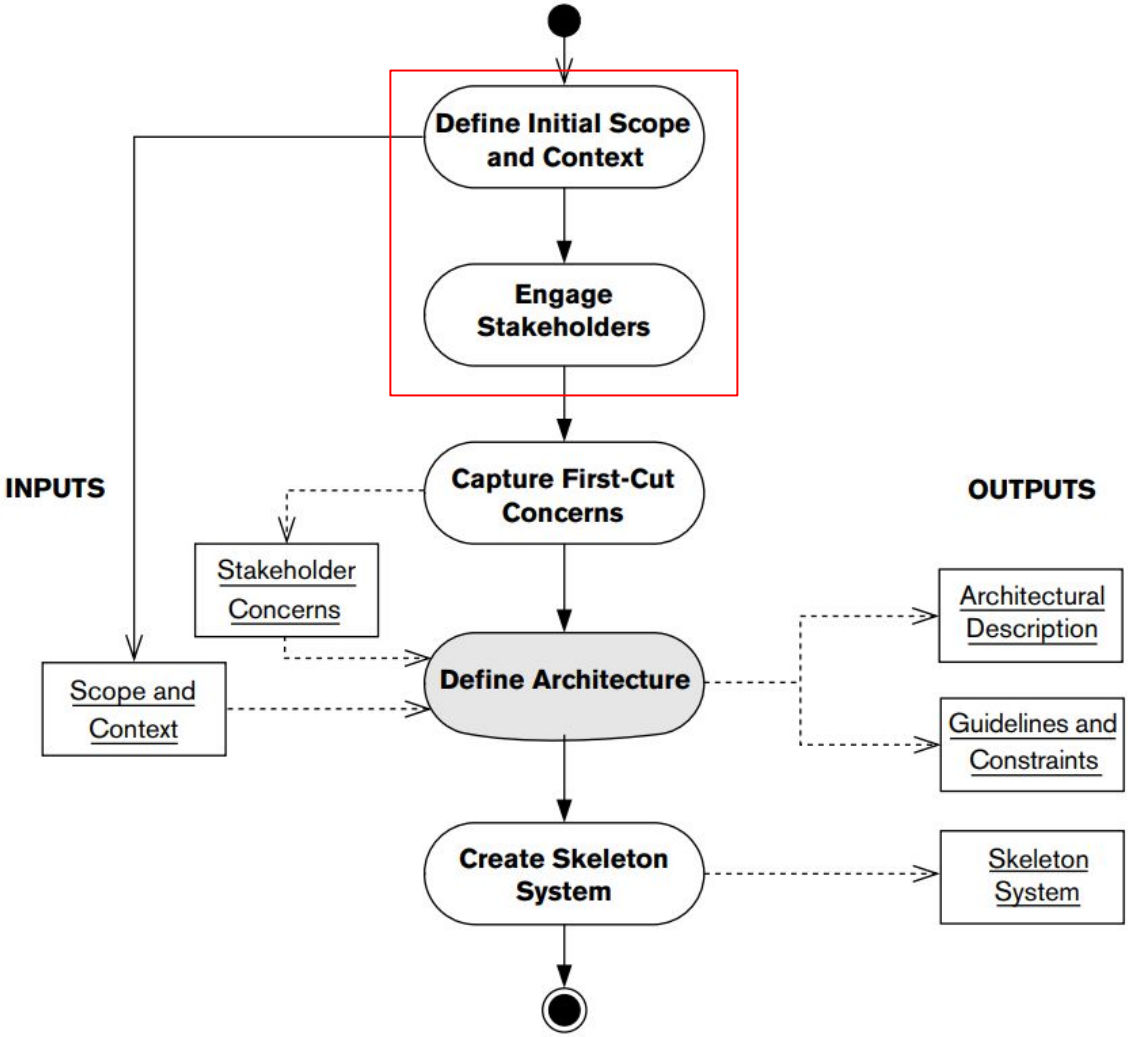
Setting Context and Identifying Stakeholders

CSCE 742 - Lecture 5 - 09/11/2018

The Architecture Definition Process



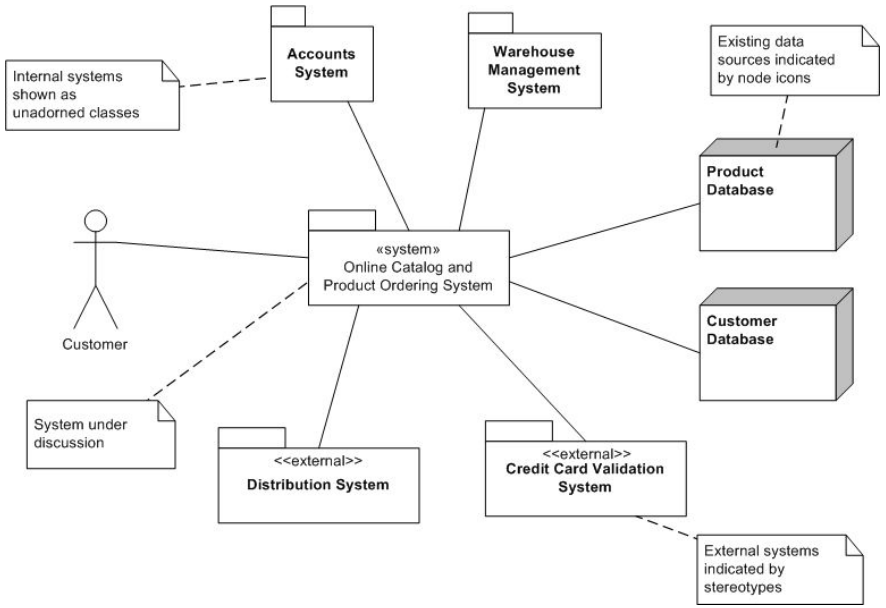
Today's Class

- Defining the system context.
  - The "Context View"
  - Establishing scope, understanding the external entities interacting with your system, and establishing boundaries.
- Identifying and engaging stakeholders.
  - "Working with Difficult People"
  - What classes of people are interested in your system?

Defining System Context

The Context View

- System context should be a view in the architectural description.
  - Context is often implicit rather than being defined.
  - Context may be loosely defined during requirements analysis, but at too low of a level of detail.
  - You need to refer to elements of the system context elsewhere in your architectural description.



The Context View

- The **context view** defines the relationships, dependencies, and interactions between the system and its environment.
  - Defines what it does/does not do.
  - Defines boundaries between it and the world.
  - Defines how it interacts with other systems and people across these boundaries.
- Focus is on the outside world, not the internal architecture of the system.
  - System is a “black box”.

System Scope

- What are the main responsibilities of the system? What is it required to do?
  - Can also define certain tasks it does not have to do.
- Requirements should already be defined. Scope definition summarizes them for stakeholders.
- Defined as a high-level list of system's key capabilities or requirements.

Example: Clothing Retailer

- The online clothing store must:
  - Present the catalog to the user, including pictures and product details.
  - Provide a flexible search facility.
  - Accept orders for goods.
  - Accept payment by credit card .
  - Provide automated interfaces into back-end systems for fulfillment.
- The system will not be required to:
  - Amend or cancel orders.
  - Allow payments by means other than credit card.
  - Display live stock levels.

External Entities

- An **external entity** is any system, organization, or person with which this system interacts in some way.
  - A user or class of user.
  - Other systems (internal or external).
  - An interface to another ecosystem.
  - A data store external to the system.
  - A physical device external to the system.
- Each external entity offers services to the system or uses services of the system.

External Entity Quality

- Quality properties of external entities affect the architecture of your system.
- Ex: travel booking system
  - Interacts with other systems around the world.
  - Systems in some locations may have low availability.
  - Travel system interface to external systems will need to account for this.
    - Retry interactions N times, log attempts.
- Not just software.
  - Hardware in the loop may fail or have a delayed response.

External Entity Identification

- Data provider or consumer
  - Supplies data or receives data.
  - Identify the content, scope, and meaning of data.
- Service provider or consumer
  - Performs some action or requests an action.
  - Identify what is being requested and its parameters, actions to be taken, data to be returned, error information, exception actions.
- Event provider or consumer
  - Subscribes and watches for events of interest.
  - Identify events of interest, meaning and content, timing of occurence.

External Interface Characteristics

- Quality properties of external **interfaces** may differ from the systems they interface.
- The interface is the constraining factor on your architecture.
- Characterize interfaces as part of planning.
  - Expected number of requests or transfers, size of data, expected growth over time.
  - Are interactions scheduled, occur in response to events, or are ad-hoc.
  - Are interactions automated, manual, or a combination?

External Interface Characteristics

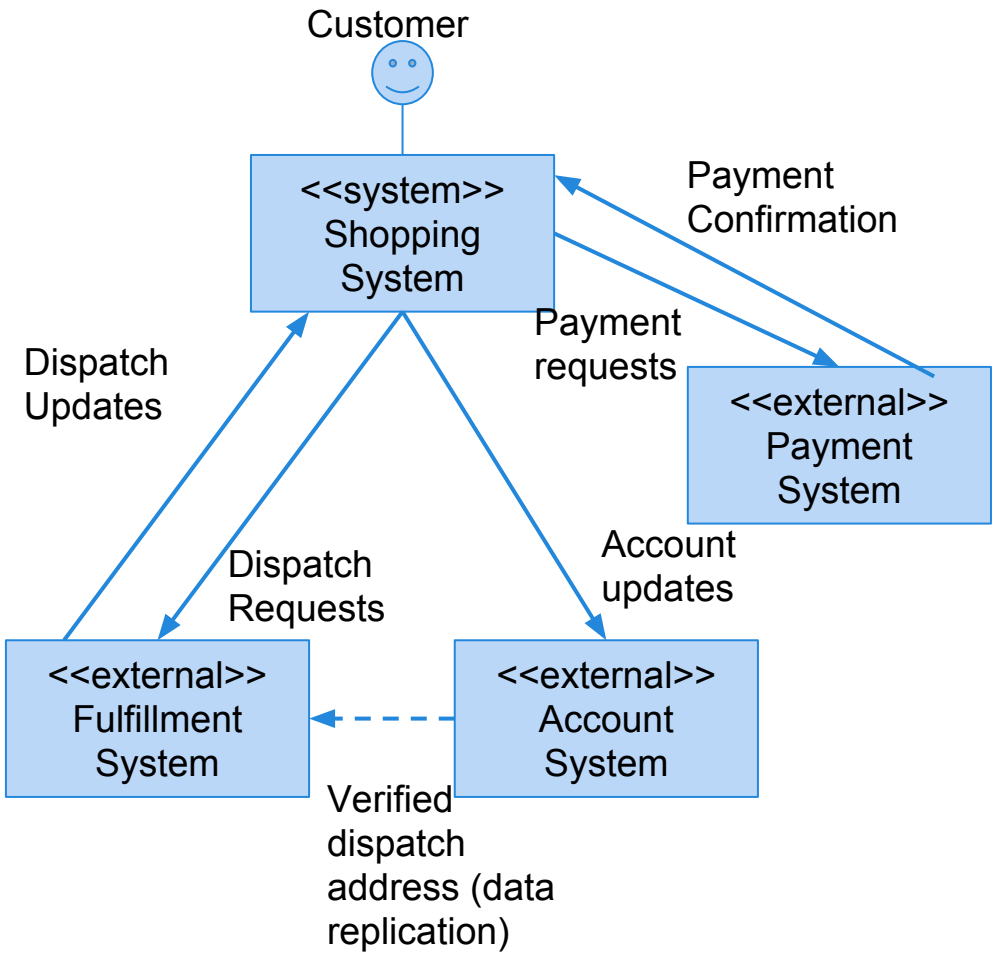
- Characterize interfaces as part of planning.
  - Do interactions need to complete fully or partially?
  - What is the criticality and timeliness of interactions?
  - Are interactions performed using individual calls or in batch?
  - What level of security is required?
  - What is the service level of the interface? (response time, scalability, availability)
  - What protocols are used by the interface?
  - What data and file formats are used?

External Interdependencies

- There may be interdependencies between your system and external entities.
- Online shopping system
  - Must interact with a payment system, a customer account system, and a fulfillment system.
  - Normally independent, but data dependency:
    - Fulfillment system contains a list of addresses for each customer. Maintained by replicating data from account system.
  - Updates to addresses within the shopping system must take this dependency into account.

External Interdependencies

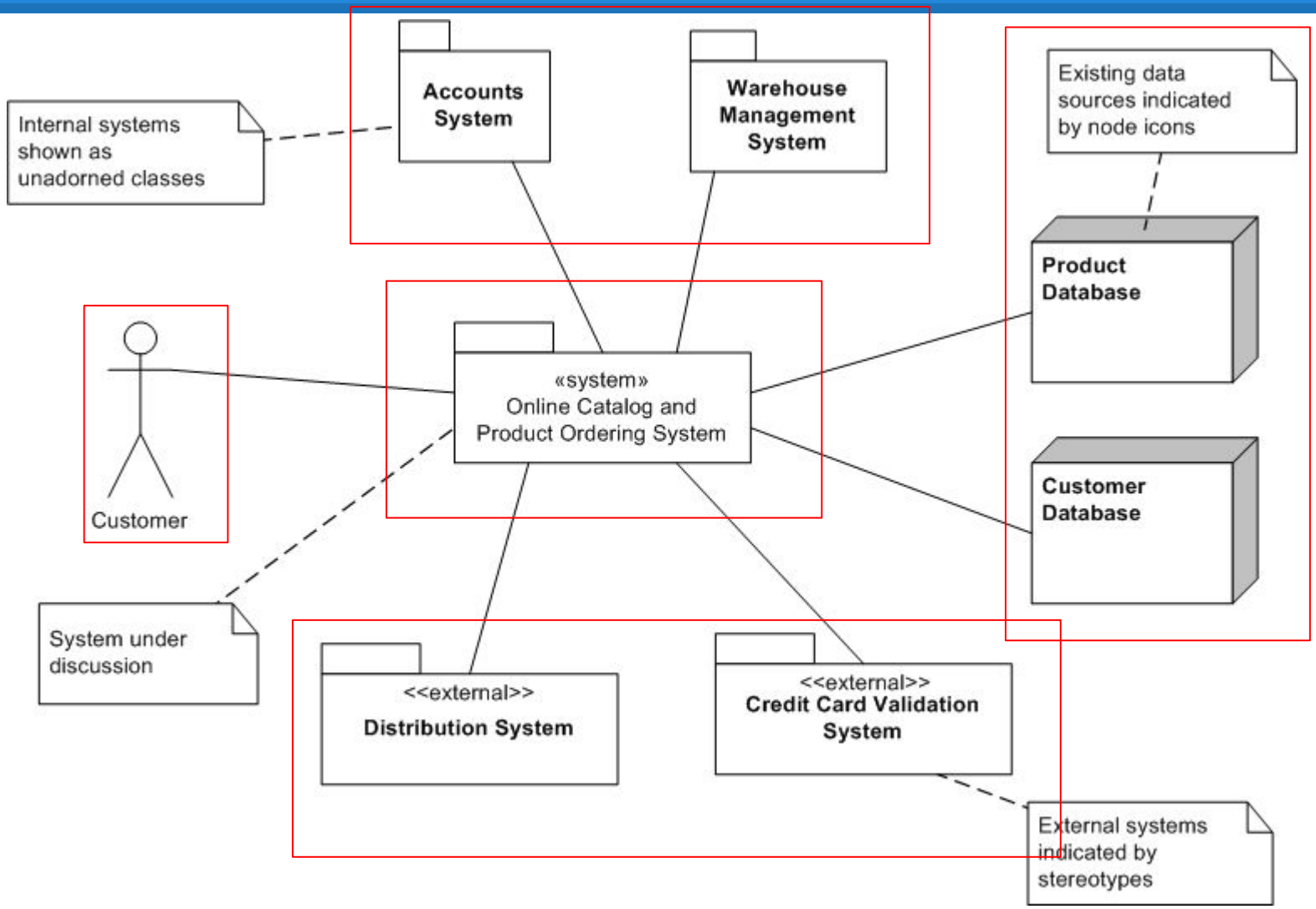
- Online shopping system
  - Architecture must handle this external system dependency.
  - Allow resubmission to fulfillment system after a delay if a request is rejected.
  - Delay orders with address updates.
  - Resubmitting failed orders will be easier if the fulfillment system interface returns reason for failure.



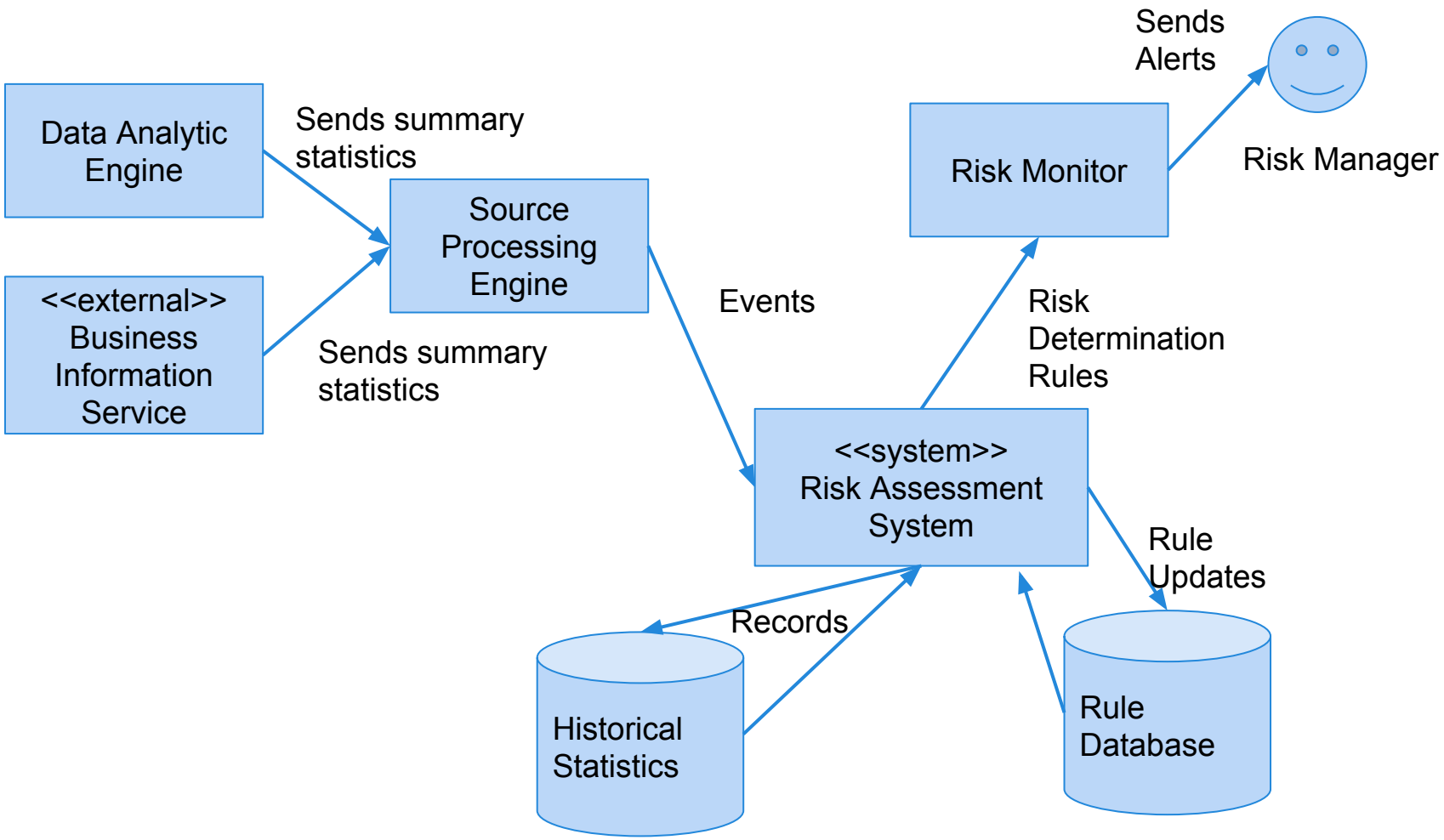
Context Model

- Places the system in its environment and details relationships with external entities.
  - The system itself, with internal structure hidden.
  - External entities, with name, nature (system, data store, person), owner, and responsibilities (services, functions, data).
  - Interfaces between system and external entities, with interactions, data exchanged, exception processing details, and quality properties..
- Should be kept relatively simple, with detail presented alongside.

Context Model



Example - Risk Assessment



Pitfalls

- Missing or incorrect external entities
  - To avoid, work with stakeholders:
    - Is all functionality required part of system scope, provided by external entity, or excluded?
  - Involve domain experts.
  - Ensure context model is under version control.
- Missing implicit dependencies
  - I.e., falsely assuming data is available in two external systems when there is actually data transfer.
  - Assume nothing, work with external organizations to uncover implicit dependencies.

Pitfalls

- Inaccurate interface descriptions
  - Important to capture architectural implications.
  - Can you confidently use an interface? Characterize the effect on the architecture?
  - Do not gloss over complicated issues.
- Inappropriate level of detail
  - Look for vague scope and requirements.
  - If context diagram too cluttered, move detail to appendices or secondary views.
  - Group external entities by type.

Pitfalls

- Scope creep
  - Increase in expectations on system responsibilities without consideration of what is achievable.
  - Challenge additions to scope.
  - Work with stakeholders to understand effect of added requirements.
  - Ensure scope changes are version-managed.
- Assumed context or scope
  - State the obvious to avoid misunderstanding.
  - Do not assume stakeholders know something.

Pitfalls

- Overcomplicated interactions
  - Interacting with legacy systems can be more complicated than expected.
  - Take time to understand interfaces early.
  - Prototype interactions and test thoroughly.
- Overuse of jargon
  - Be careful not to use terms unfamiliar to stakeholders.
  - If you need to use jargon, provide a glossary.

Activity - Airport Parking

- You will develop a new automated parking system at the CAE airport.
- In this new system:
  - A user can insert their card into the card reader at the ramp entrance.
  - This will record the time they entered airport parking.
  - They then can use the same credit or debit card to pay at an exit lane.
  - The system should be fully automated.
  - The system should also support ticketed parking
    - User receives a ticket and pays either by credit card or cash on exiting.

Activity - Airport Parking

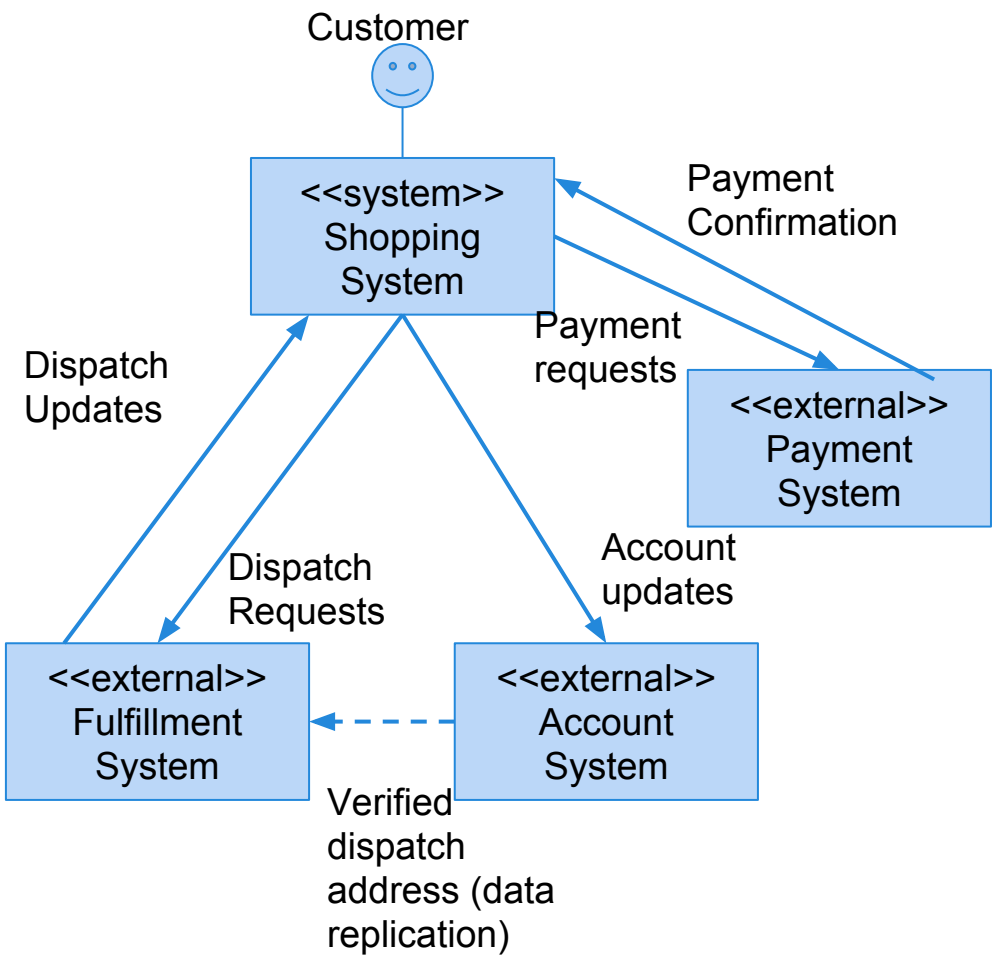
The system needs to interact with a number of entities and systems, including:

- Customers parking in the ramp
- Airport police and emergency responders
- Ramp managers
- External systems for validating credit card details and submitting payments
- The airport's accounting system
- External physical gate systems with basic controllers (raise / lower)
- External physical systems for signage
- An existing personnel system for staffing exit kiosks

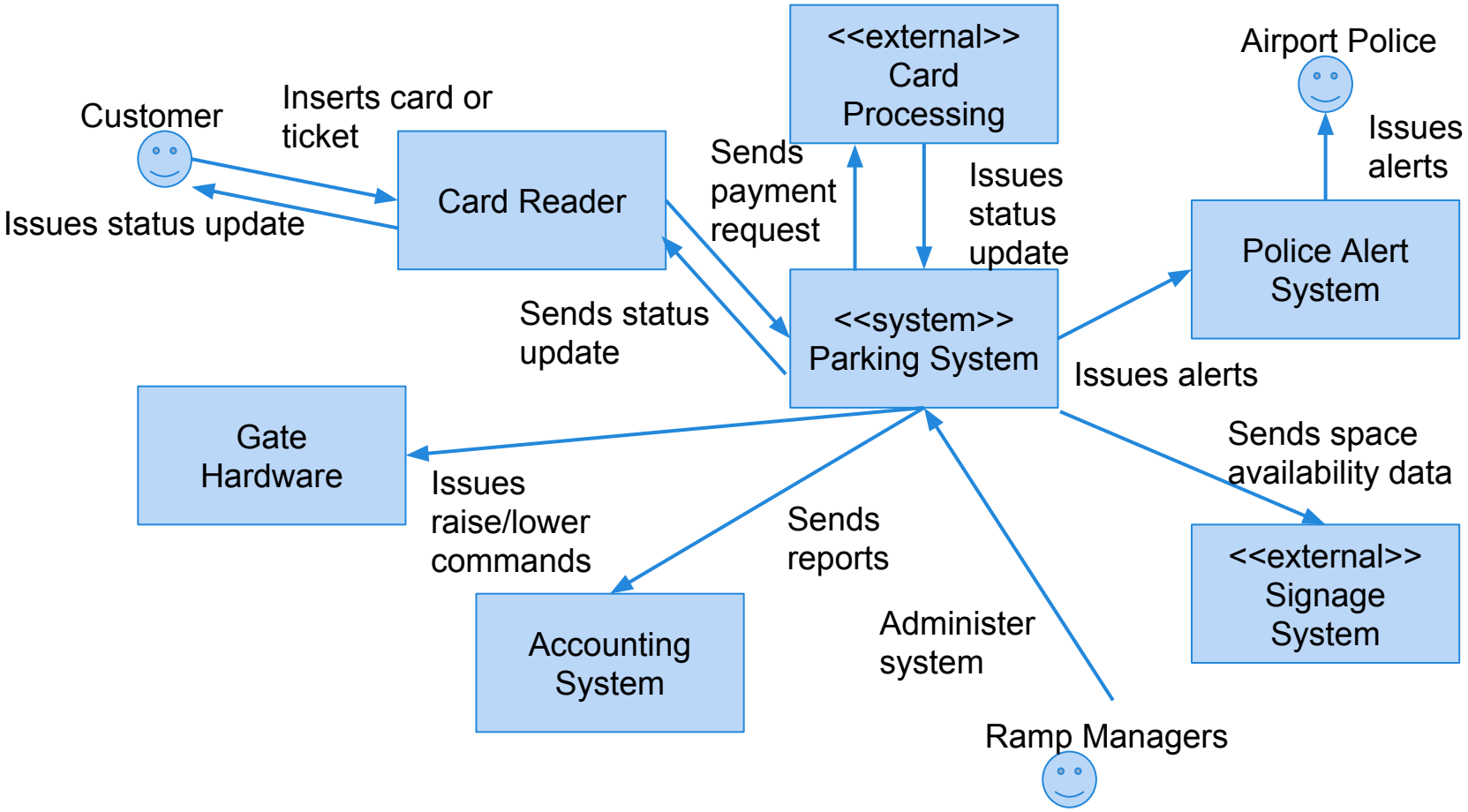
Activity - Airport Parking

Develop a context view diagram for the airport parking system.

Think about the physical devices, stakeholders, and internal and external software systems.



Activity - Airport Parking



Identifying and Engaging Stakeholders

The Stakeholders

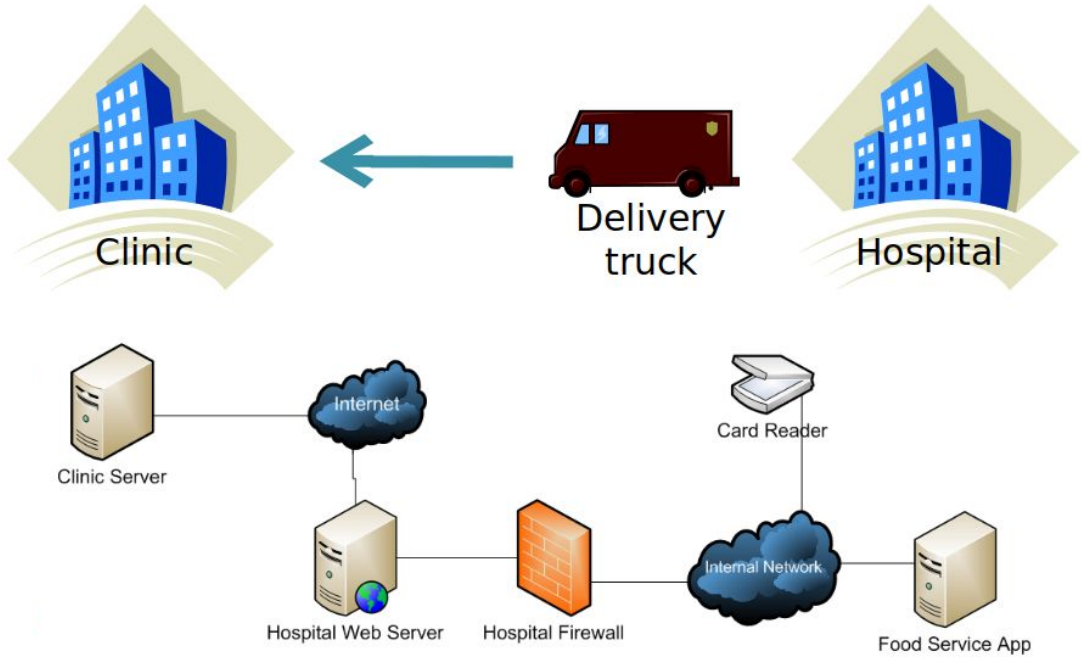
The Stakeholders

- Architecture definition requires identifying and engaging the **stakeholders**.
  - People, groups, or entities with an interest in or concerns about the realization of the architecture.
- Identifying stakeholders and gaining commitment is key to project success.
  - Need to cast net widely **early** in the project.
- Need to draw up and maintain a list of potential stakeholders.
  - May need *proxy stakeholders* who can speak for future stakeholders.

Who are the Stakeholders?

Hospital Food Ordering

- Who are the stakeholders of this system?



Who are the Stakeholders?

- A subjective choice.
- There are no purely objective criteria for whether you chose “correct” stakeholders.
- Selection depends on system goals, organization considerations, politics, resources, cost, timescale.
- Cast your net widely early in the project.
  - Reconcile differences while you still can.

Effective Stakeholders Are...

- Informed
  - Have information, background, understanding needed to offer feedback.
- Committed
  - Willing to be available and to make decisions.
- Authorized
  - Allowed to make binding decisions.
- Representative
  - Do you have someone who can represent a group of stakeholders with authority?

Stakeholder Responsibilities

- Ensure that all concerns are communicated.
  - Representative stakeholders must convey concerns of the people they represent.
- Make decisions in a timely and authoritative manner.
  - And stick with them!
  - If they lack authority, they must escalate to those who do have that power.
- Review the architecture definition to ensure system meets their concerns.

Common Classes of Stakeholders

- **Acquirers** oversee system procurement.
  - Typically senior management, legal, purchasing.
  - Make monetary decisions.
  - *Concerns* center around alignment with corporate objectives, return on investment, cost/timescale of project, resources needed to build and run system.
- **Assessors** oversee system conformance to legal regulations and standards.
  - *Concerns* center around formal, demonstrable compliance to any relevant regulations.

Common Classes of Stakeholders

- **Communicators** explain the system to others through documentation and training.
  - *Concerns* lie in understanding the architecture and explaining it to audiences with varying backgrounds.
- **Developers** write the code, using specifications and architecture.
  - *Concerns* around understanding architecture, build standards, choice of platform, language, tool support, maintainability, flexibility, preservation of knowledge over time.

Common Classes of Stakeholders

- **Maintainers** will manage system evolution after release.
  - *Concerns* focus on maintainability documentation, system monitoring, debugging environment, source control, knowledge preservation.
- **Suppliers** build the hardware, software, or infrastructure on which the system will run.
  - Not usually part of building, running, or using system
  - Still impose constraints due to requirements of products they supply.
  - *Concerns* around whether your system can work with their product ecosystem.

Common Classes of Stakeholders

- **Support staff** provide support to users for the product or system when it is running.
  - *Concerns* around having information required to solve problems with users.
- **System administrators** ensure the operation of the system once deployed.
  - *Concerns* around system monitoring and management, business continuity, disaster recovery, availability, resilience, scalability.

Common Classes of Stakeholders

- **Testers** verify system correctness before and after release.
  - Often independent from developers. Can perform a more thorough job of evaluating the system than other stakeholders.
  - *Concerns* around refinement of requirements, ability to prove requirements are met, building testing infrastructure
- **Users** interact with system functionality.
  - *Concerns* around scope, functionality, performance, security. Ultimate judges of your success.

Example: Traditional Development

- An educational software supplier has partnered with a college to develop a course content management system.
- Who are our stakeholders?
  - Think about the groups discussed earlier, and who fits each group.
  - Stakeholders can belong to either organization.

Example: Traditional Development

- Acquirers:

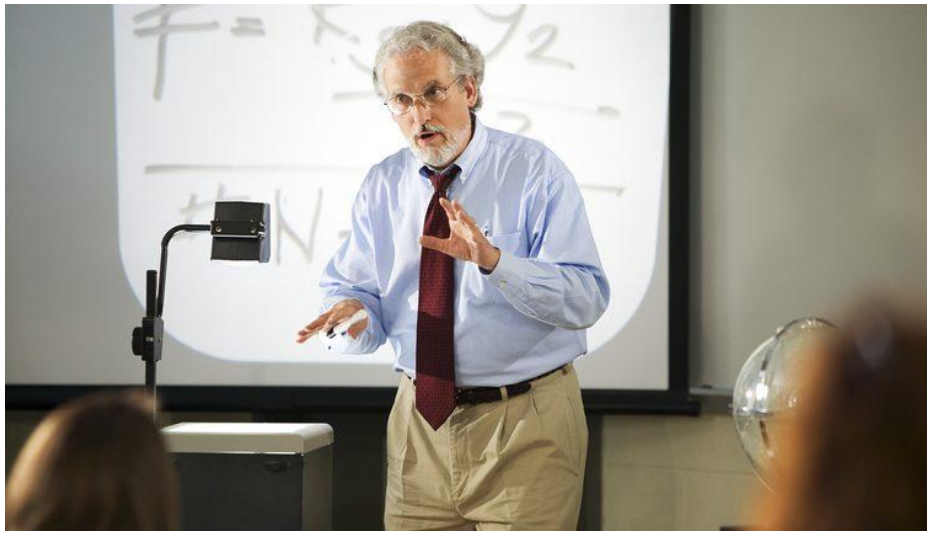
Senior management at software company. Purchasing department at college.

- Users:

Lecturers and admin staff at college.

- Developers, maintainers from software company.
- System admins from college.
- Communicators:

From company. Provide training and user manuals.



Example: Off-The-Shelf Deployment

- Selecting, tailoring, and implementing an existing software package within your overall system.
- A hardware manufacturer wants a resource planning system to manage its supply chain from ordering to delivery.
- Who are our stakeholders?
  - Think about the groups discussed earlier, and who fits each group.

Example: Off-The-Shelf Deployment

- Acquirers:

Senior management, purchasing department, IT management

- Users:

Internal staff (working in order entry, purchasing, finance, manufacturing, distribution)

- Developers, system admins, maintainers:

Internal staff

- Communicators:

Internal trainers, internal help desk, system supplier

Proxy Stakeholders

- May not be possible to identify all stakeholders until system is developed.
- Must identify proxy stakeholders - individuals or groups that predict the concerns of the real stakeholders.
- Offer feedback and concerns on behalf of potential stakeholders who can’t be interviewed.
  - I.e., you can’t interview customers that don’t exist.

Checklist (Food for Thought)

- Have you identified stakeholders from each class? If not, is the omission justified?
- Have you informed stakeholders of their responsibilities, have they agreed to these?
- Does each stakeholder understand the level of commitment involved?
- Is each stakeholder aware of the particular role to fulfill?

Checklist (Food for Thought)

- For each group of stakeholders, have you identified a suitable representative?
  - Does this proxy have the knowledge and authority to speak on behalf of the group?
- For each stakeholder group that does not yet exist, have you identified and engaged a suitable proxy?
- If suppliers are included as stakeholders, are their responsibilities and contractual obligations understood by both sides?

Key Points

- The **context view** defines the relationships, dependencies, and interactions between the system and its environment.
  - Defines what it does/does not do.
  - Defines boundaries between it and the world.
  - Defines how it interacts with other systems and people across these boundaries.
- Focus is on the outside world, not the internal architecture of the system.

Key Points

- Architecture definition requires identifying and engaging the **stakeholders**.
  - People, groups, or entities with an interest in or concerns about the realization of the architecture.
- Identifying stakeholders and gaining commitment is key to project success.
- **The stakeholders help set the context.**

Next Time

- Concerns, Principles, and Decisions
  - Source: Rozanski & Woods: ch. 8

- Homework:
  - Project 1, due 9/18
  - Assignment 1, due 9/25
    - Questions?