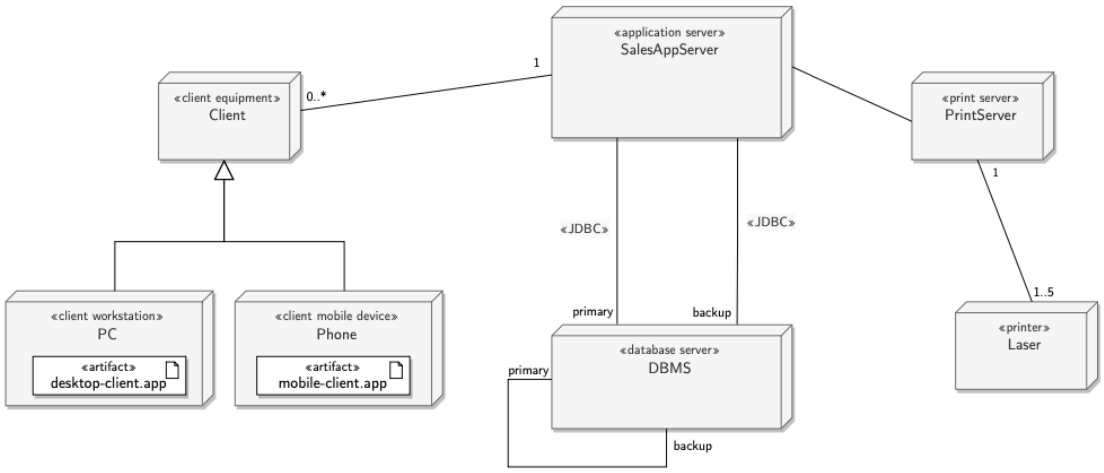
Midterm

1. The following UML deployment diagram was produced for a sales application software system that supports both desktop and mobile clients. The system contains an application server and two database servers. Both database servers are identical. The first acts as a main server, while the second acts as a redundant backup in case the first one fails. The application server is connected to a print server and a collection of laser printers to support printing of orders and invoices.



   a. What kind(s) of software structure(s) does this diagram represent?

   This UML deployment diagram represents an Allocation structure. More specifically, this is a Deployment structure mapping software elements to system elements.

   b. Considering the 4+1 view model, which view does this diagram represent?

   This UML deployment diagram represents the Physical view from the 4+1 view model. It shows the system hardware and how software components are distributed across the processor and communication nodes in the system.

   c. Briefly describe the purpose and importance of this structure and view in the broader context of software architecture and design.

   The purpose of allocation (deployment) structures and the Physical view from the 4+1 view model is to show how elements from other structures such as module structures and component & connector structures relate to non-software structures (such as CPUs, file systems, and networks). They provide the system engineer's perspective.

   This structure and view are important because they can be used to

reason about performance, data integrity, security, and availability. It is of particular interest in distributed systems and is the key structure involved in the achievement of the deployability architecture characteristic.

2. Consider the description, requirements, and domain concerns provided below for a system.

   Description:
   - A company wants to build a software system supporting chat nurses (advice nurses) answering questions from customers about potential health problems.

   Users:
   - 250+ nurses worldwide, hundreds to thousands of customers

   Requirements**:**
   - (1) The system shall have access to patient medical histories.
   - (2) The system shall provide a service-level agreement on turnaround time for interactions.
   - (3) The system shall assist nurses in providing a medical diagnosis.
   - (4) Support nurses may be geographically divergent from clients (i.e., not in the same place).
   - (5) The system shall enable client customers to reach local medical staff (if necessary), contacting the local medical staff directly ahead of time (if necessary).

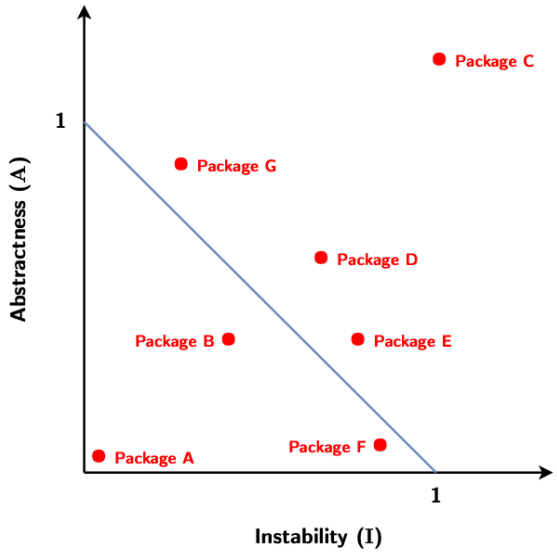
   Additional Context**:**
   - The company is heavily involved in building software solutions in niche spaces like this one.
   - The company is in a period of aggressive growth with a 2nd round venture capitalist funding.
   - Fast time to market is an overall company goal to capture the market.
   - A legal team has determined that these conversations between nurses and customers is not considered a medical record.

   Identify a list of relevant architecture characteristics for the system. Provide a brief rationale for the relevance of each identified architecture characteristic.

   - **Scalability**: The system must be able to handle the large number of nurses expected as well as the predicted in the numbers of customers
   - **Security/Privacy**: Because the system has access to patient medical histories which contain sensitive personal information, this information needs to be protected

- **Performance/Legal**: A service-level agreement describing the expected turnaround time for interactions indicates the need for performance considerations to fulfill the agreement; the agreement may also be legally binding
- **Reliability/Safety**: Because the system is supporting nurses in providing a medical diagnosis, it is imperative that the error rates be low (i.e., high reliability) to ensure patient safety (which can be impacted by a wrong diagnosis)
- **Deployability**: The geographically distributed nature of the system raises concerns about the deployability of the system
- **Availability**: The need to be able to reach/contact local medical staff suggests the need for high availability because customers should not have to wait for access to the system
- **Agility, Testability, Deployability**: The domain concerns expressed as "time to market" translate to these characteristics
- **Legal**: Implicit domain knowledge may suggest that there is a legal obligation to protect sensitive patient medical histories

3. You are part of a software architecture and design team and several of your colleagues have been measuring the distance from the main sequence for the packages in the initial design of a system you are developing. They have plotted their results in the figure shown below. Your job is to interpret these results. Given the results shown in the plot below, what observations and recommendations do you have?

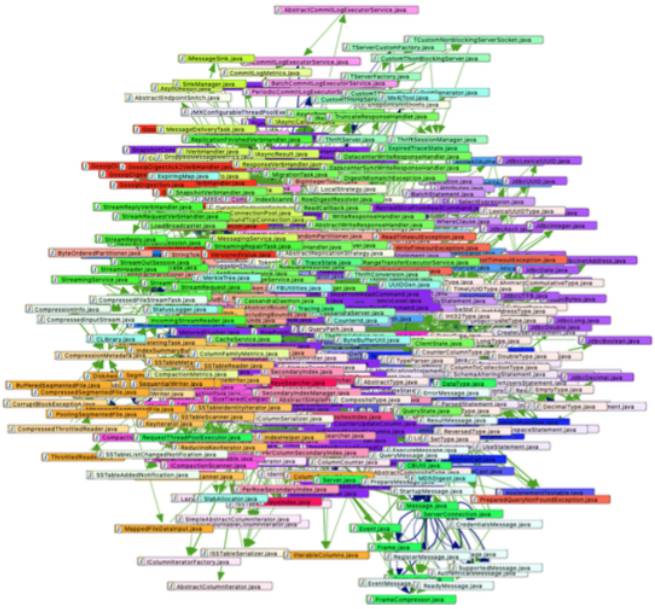


Package A is in the *Zone of Pain*. This means that this package is likely to contain code with too much implementation and not enough abstraction and that could become brittle and hard to maintain. Therefore, we should consider introducing more abstractions.

Package C is in the *Zone of Uselessness*. This means that this package is likely to contain code that is too abstract and that may become difficult to use. Therefore, we should consider refactoring to remove these useless elements.

All other packages are close to the main sequence. This means that these packages each exhibit a healthy mixture of abstractness and instability (which are competing concerns). Therefore, these packages are (currently) fine as they are.

4. Below is a visualization of the dependency graph among the components of Apache-Cassandra, a very popular and successful open-source project that implements a NoSQL database.



   a. Based on this visualization, what observations do you have about the architecture of Apache-Cassandra?

      This is an example of the Big Ball of Mud. The antipattern refers to the absence of any discernible architecture structure. This can clearly be observed in the visualization.

   b. What are the primary challenges for software developers that want to contribute to this open-source project?

      The lack of structure makes such systems increasingly difficult to

change and maintain. Maintenance contains issues related to *testability, interchangeability, extensibility, deployability, scalability,* and *comprehensibility*.

A software developer that wishes to contribute to such a project may face significant challenges in understanding where to add new functions/features and/or change existing functions/features. In turn, they will face challenges in assessing how their changes may impact other existing functions/features.

5. A Canadian insurance company wants to create a software system to help file and process insurance claims (for cars, homes, etc.). Insurance claims processing is a very complicated process. Despite that there is a fairly standard process for filing and processing a claim, regardless of jurisdiction (e.g., province/territory), each jurisdiction has different claims rules and regulations for what is and what is not allowed in an insurance claim. For example, some jurisdictions allow free windshield replacement if your windshield is damaged by a rock, whereas other jurisdictions do not. This creates a very large set of conditions for a standard claims process. The insurance company wants to ensure that new rules within a jurisdiction can be added, removed, or changed without impacting other parts of the system. Similarly, it is desired that jurisdictions can also be added or removed (in the case of provincial differences or similarities) without impacting other parts of the system.
    a. Which (monolithic) architectural style would you propose to handle the complexity of the insurance claims processing system? Provide a brief rationale for your suggestion. Be sure to describe how your suggestion will help to achieve the design goals related to the modifiability and extensibility of the insurance claims processing system.

This is an example of the *Microkernel* architecture. The claims rules for each jurisdiction can be contained in separate standalone *plug-in components* (implemented as source code or a specific rules engine instance accessed by the plug-in component). This way, rules can be added, removed, or changed for a particular jurisdiction without impacting any other part of the system. Furthermore, new jurisdictions can be added and removed without impacting other parts of the system. The core system would be the standard process for filing and processing a claim, something that doesn't change often and that is rather common to all jurisdictions.

b. Provide a simple sketch (i.e., box and line diagram) of your proposed architecture design by instantiating the elements from your selected architecture style.