Designing with Patterns

Synopsis:

In this exercise, your task is to propose a design for the Messaging Infrastructure layer of the new Brokerage Information System (BIS) for the BizCo company. To accomplish this, you must select a pattern and instantiate it, modifying the architecture to meet the quality attribute requirements for the company.

BizCo Business Goals

To establish BizCo as the industry leader in brokering appraisal for commercial and residential properties. We must:

- Increase market share of brokerage services
- Dramatically decrease response time to changing market conditions
- Have direct, secure access to brokerage information 24/7
- Have reporting that aggregates, collates, and communicates timely information clearly and as needed

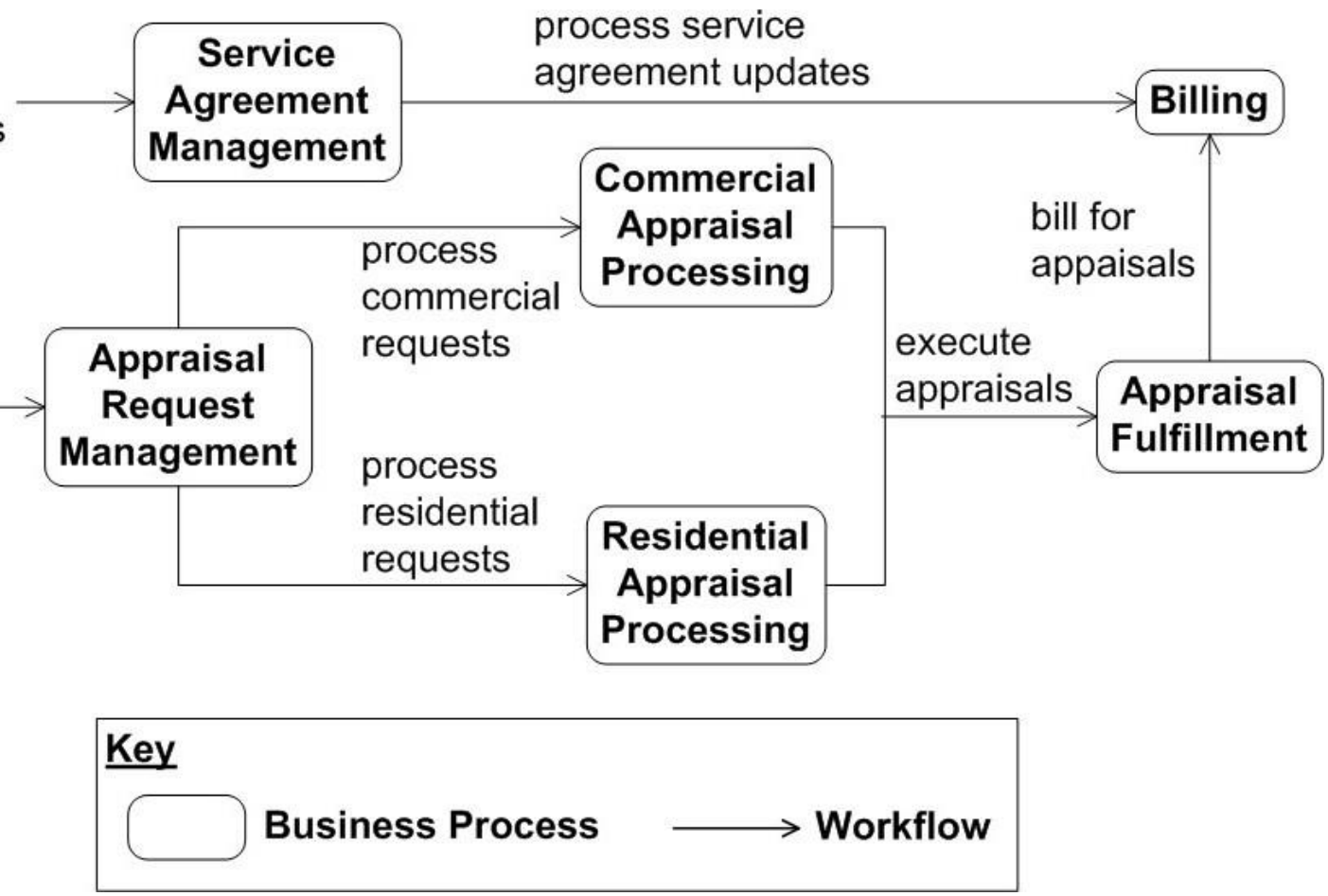
BizCo Business Processes

A real estate appraisal is needed to secure a mortgage loan when purchasing a property. A real estate appraisal is performed by licensed appraisers who work as independent contractors. As a real estate appraisal broker, BizCo connects the lender with an independent appraiser.

The typical workflow at a BizCo branch office begins with a Service Agreement between BizCo and lenders, and between Bizco and appraisers. The Service Agreement Management business process establishes and records the service agreements.

The Appraisal Request Management business process records appraisal requests and routes residential and commercial requests differently pairing commercial and residential appraisals with commercial and residential appraisers and routes the requests accordingly.

Once the property has been appraised, the Appraisal Fulfilment process records the appraisal results, and notifies the Billing process to bill the lender and pay the appraiser according to their service agreement.



BizCo Information Systems

Three standalone systems operate at each BizCo branch. Please note that the systems are not integrated within the branch and do not integrate externally.

Table 2 - BizCo Existing Information Systems per Branch

BizCo Branch Systems	Description
Commercial Property System (CPS)	Services requests for data about appraisal requests submitted, assigned, and fulfilled for commercial properties including request details, lender/appraiser assignments, and status.
Residential Property System (RPS)	Services requests for data about appraisal requests submitted, assigned, and fulfilled for residential properties including request details, lender/appraiser assignments, and status.
Brokerage Billing System (BBS)	Services requests for data about branch office, lender, and appraiser business addresses and contacts, contract terms and conditions, and accounts receivable.

Table 3. X indicates system support for a business process

Business Process	CPS	RPS	BBS
Appraisal Request Management	**X**	**X**	
Commercial Appraisal Processing	**X**		
Residential Appraisal Processing		**X**	
Appraisal Fulfillment	**X**	**X**	
Billing			**X**
Service Agreement Management			**X**

BizCo Information Reporting

CPS, RPS, and BBS at each branch generate reports independently

- Summarize brokerage activities
- Biweekly
- Hard-copy
- Sent to main office in Pittsburgh via courier

BizCo Brokerage Information System (BIS) – the Proposed System

BizCo needs to gather more extensive information from the branch offices' existing systems in a timely manner. To this end, your organization plans to develop the new **Brokerage Information System (BIS).**

The BIS must allow the main office to display information on:

- Branch office, lender, and appraiser business address and contacts
- Contract terms and conditions for lenders and appraisers
- Appraisal requests submitted, assigned, and fulfilled
- Brokerage activities across and for individual branch offices
- Brokerage activities across and for individual lenders, appraisers, and brokers
- Account receivables aggregated among all offices

BizCo BIS Software Architecture

The BizCo systems architect has designed a Layered Architecture, consisting of four layers, as a module structure for the new system, encapsulating the three existing systems, CPS, RPS, and BBS.

Figure 2 BIS Architecture

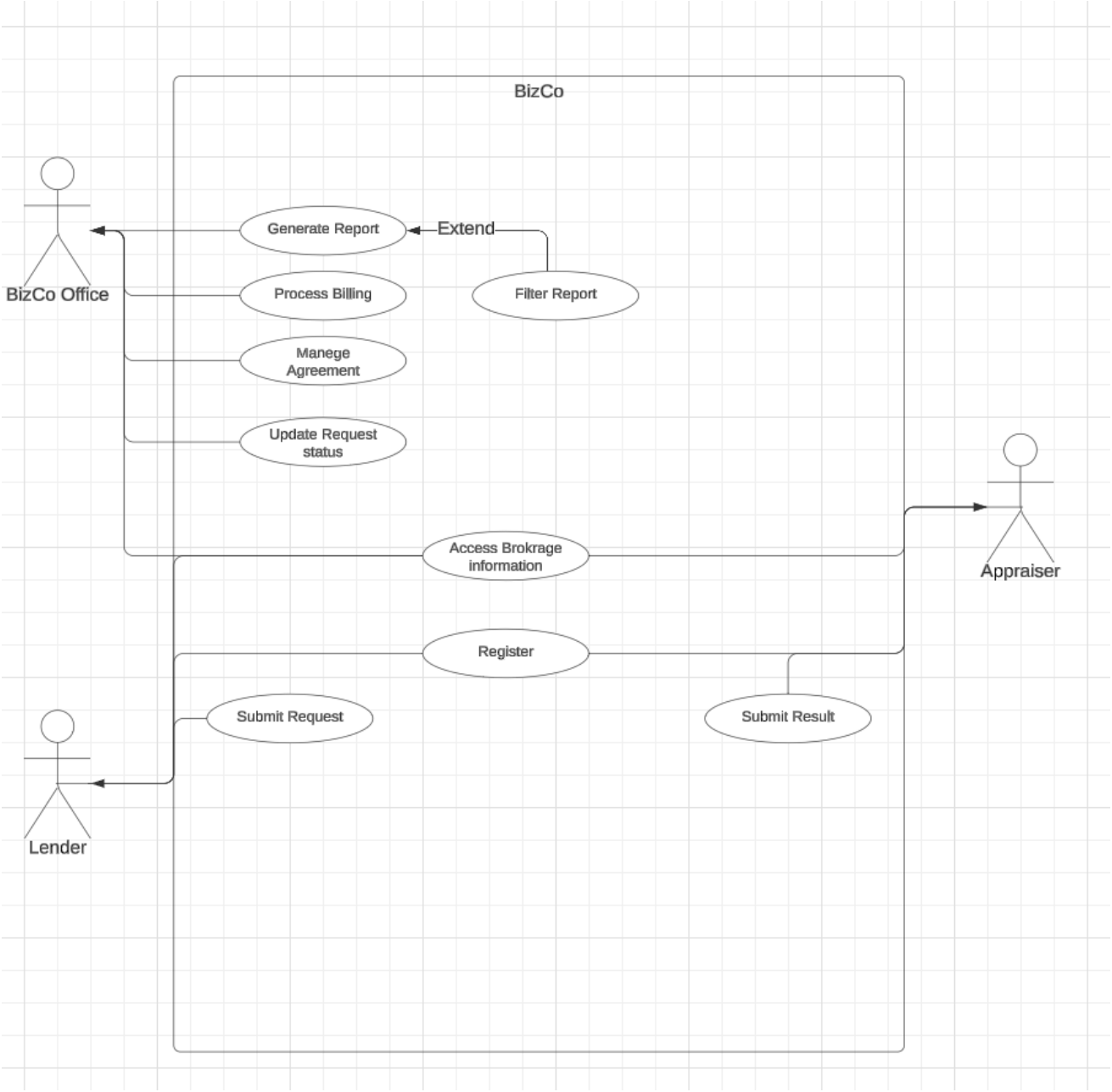
Table 4 BIS Architecture Layers

Layer	Description
User Interface	Allows users to select, customize, and display brokerage data and activity reports.
Business Logic	Interacts with remote services to collect data and collates the data into brokerage reports.
Message Infrastructure	Provides access to remote services.
Services	A collection of services that provide brokerage data. CPS,

Do the Following:

- **Use case Diagram**

Based on your understanding of the system and assumptions, draw a use case diagram and construct a table explaining each use case.



Generate Report

Description: BizCo Office generates detailed reports for appraisal activities and branch performance. These reports are used for analysis and decision-making.

Actors: BizCo

Office Filter Report

Description: Extends the functionality of report generation by allowing users to filter reports based on specific needs, such as appraiser, lender, or branch.

Actors: BizCo Office

Process Billing

Description: Manages invoices for lenders and payments to appraisers based on completed appraisals and service agreements.

Actors: BizCo Office

Manage Agreement

Description: Records and updates agreements between BizCo and lenders or appraisers, including contract terms and conditions.

Actors: BizCo Office

Update Request Status

Description: Tracks and updates the status of appraisal requests, ensuring that both lenders and appraisers are informed about progress.

Actors: BizCo Office

Submit Request

Description: Lenders submit appraisal requests, which are routed based on property type (commercial or residential) to appropriate appraisers.

Actors: Lender

Submit Result

Description: Appraisers submit completed appraisal results into the system for fulfillment and billing processes.

Actors: Appraiser

Access Brokerage Information

Description: Provides authorized users with access to brokerage information, including branch details, lender/appraiser contracts, and account summaries.

Actors: BizCo Office,

Lender Register

Description: Registers new users, such as lenders, appraisers, and BizCo staff, into the system to enable proper role-based functionality.

Actors: BizCo Office

List of Actors

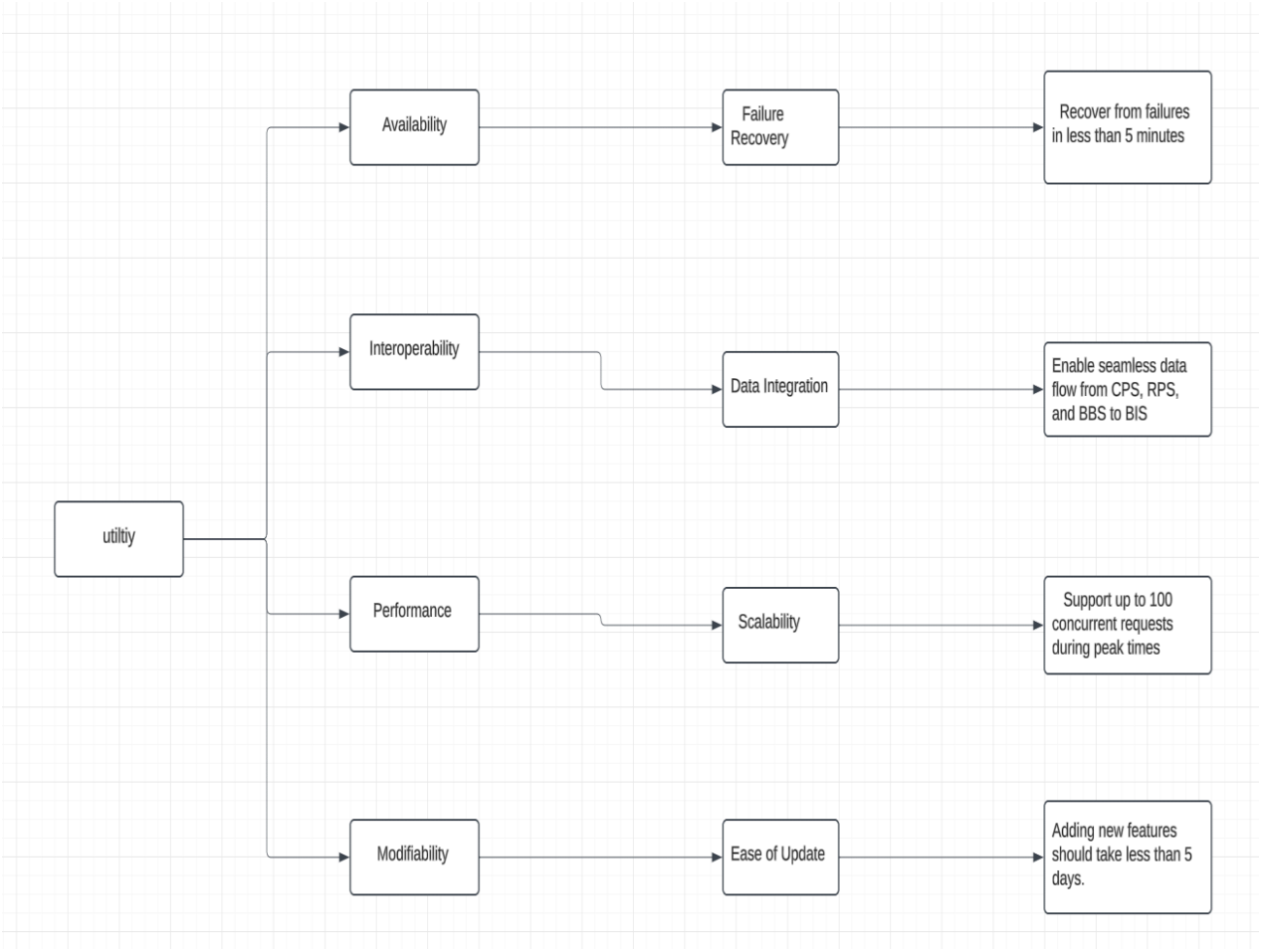
BizCo Office: Responsible for report generation, billing, agreement management, and request status updates.

Lender: Initiates appraisal requests and accesses brokerage-related information.

Appraiser: Completes appraisals and submits results to the system for further processing.

Identify the quality attribute requirements.

- Create a utility tree for the current system. Consider a minimum of **four** different quality attributes. Ensure that **the goals and scenarios(functionalities)** that you elucidated at the leaf nodes have explicit responses and **response measures**.



- Based on the utility tree you created on the previous question, Use the following table to write down what you think are the most critical quality attributes for the Messaging Infrastructure layer and prioritize their relative importance on a scale from 1 to n, where 1 is most important quality attribute requirement for the messaging infrastructure, and n is the least important which should correspond to utility tree source.

*Hint: Review the **business goals**!*

Relative Importance	Quality Attribute	Utility Tree Score
1	Availability	**High**
2	Interoperability	**High**

3	Performance	**Medium**
4	Modifiability	**Medium**

Design the Messaging Infrastructure Layer •

Now, you can begin designing the Messaging Infrastructure layer using one of three patterns: **Messaging, Publisher-Subscriber, or SOA**. In this step, you must first consider the patterns, weighing the pros and cons of each and the tradeoffs from a *quality attribute perspective*.

Read about and consider the Messaging, Publisher-Subscriber, and SOA patterns.

The Messaging Pattern

The Messaging Pattern	
Context	Some distributed systems are composed of services that were developed independently. To form a coherent system, however, these services must interact reliably, but without incurring overly tight dependencies on one another.
Problem	Integrating independently developed services, each having its own business logic and value, into a coherent application requires reliable collaboration between services. However, since services are developed independently, they are generally unaware of each other's specific functional interfaces. Furthermore, each service may participate in multiple integration contexts, so using them in a specific context should not preclude their use in other contexts.
Solution	Connect the services via a message bus that allows them to transfer data messages asynchronously. Encode the messages (request data and data types) so that senders and receivers can communicate reliably without having to know all the data type information statically.

Messaging Pattern

Figure 3 Messaging Pattern, Sequence Diagram of Services Interactions

Messaging Pattern Benefits	**Messaging Pattern Liabilities**
Services can interact without having to deal with networking and service location concerns.	Lack of statically typed interfaces makes it hard to validate system behavior prior to runtime.
Asynchronous messaging allows services to handle multiple requests simultaneously without blocking.	Service requests are encapsulated within self-describing messages that require extra time and space for message processing.

Allows services to participate in multiple application integration and usage contexts.	

Publisher-Subscriber Pattern	
Context	Components in some distributed applications are loosely coupled and operate largely independently. If such applications need to propagate information to some or all of their components, a notification mechanism is needed to inform the components about state changes or events that affect or coordinate their own computation.
Problem	The notification mechanism should not couple application components too tightly, or they will lose their independence. Components want to know only that another component is in a specific state, not which specific component is involved. Components that disseminate events often do not care which other components want to receive the information. Components should not depend on how other components can be reached or on their specific location in the system.
Solution	Define a change propagation infrastructure that allows publishers in a distributed application to disseminate events that may be of interest to others. Notify subscribers interested in those events whenever such information is published.

Publisher-Subscriber Pattern

Figure 4 Publisher-Subscriber Pattern, sequence diagram of services interactions

Publisher-Subscriber Pattern Benefits	**Publisher-Subscriber Pattern Liabilities**
Publishers can asynchronously transmit events to Subscribers without blocking.	Publishing can cause unnecessary overhead if Subscribers are interested in only a specific type of event.
Asynchronous communication decouples Publishers from Subscribers, allowing them to be active and available at different times.	Filtering events to decrease event publishing and notification overhead can result in other costs (e.g., decrease in throughput, unnecessary notifications, breakdown of anonymous communication model).
Publishers and Subscribers are unaware of each other's location and identity.	

Dynamic Routing Pattern (SOA)

	Dynamic Routing Pattern (SOA)
Context	It is often necessary to build complex business processes by wiring together a set of relatively simple services in a dynamic way.
Problem	Routing messages through a distributed system based on filtering rules is inefficient because messages are sent to every destination's filter and router for inspection and rules resolution, whether or not the message could be processed.
Solution	Define a message router that includes both filtering rules and knowledge about the processing destination paths so that messages are delivered only to the processing endpoints that can act on them. Unlike filters, message routers do not modify the message content and are concerned only with message destination.

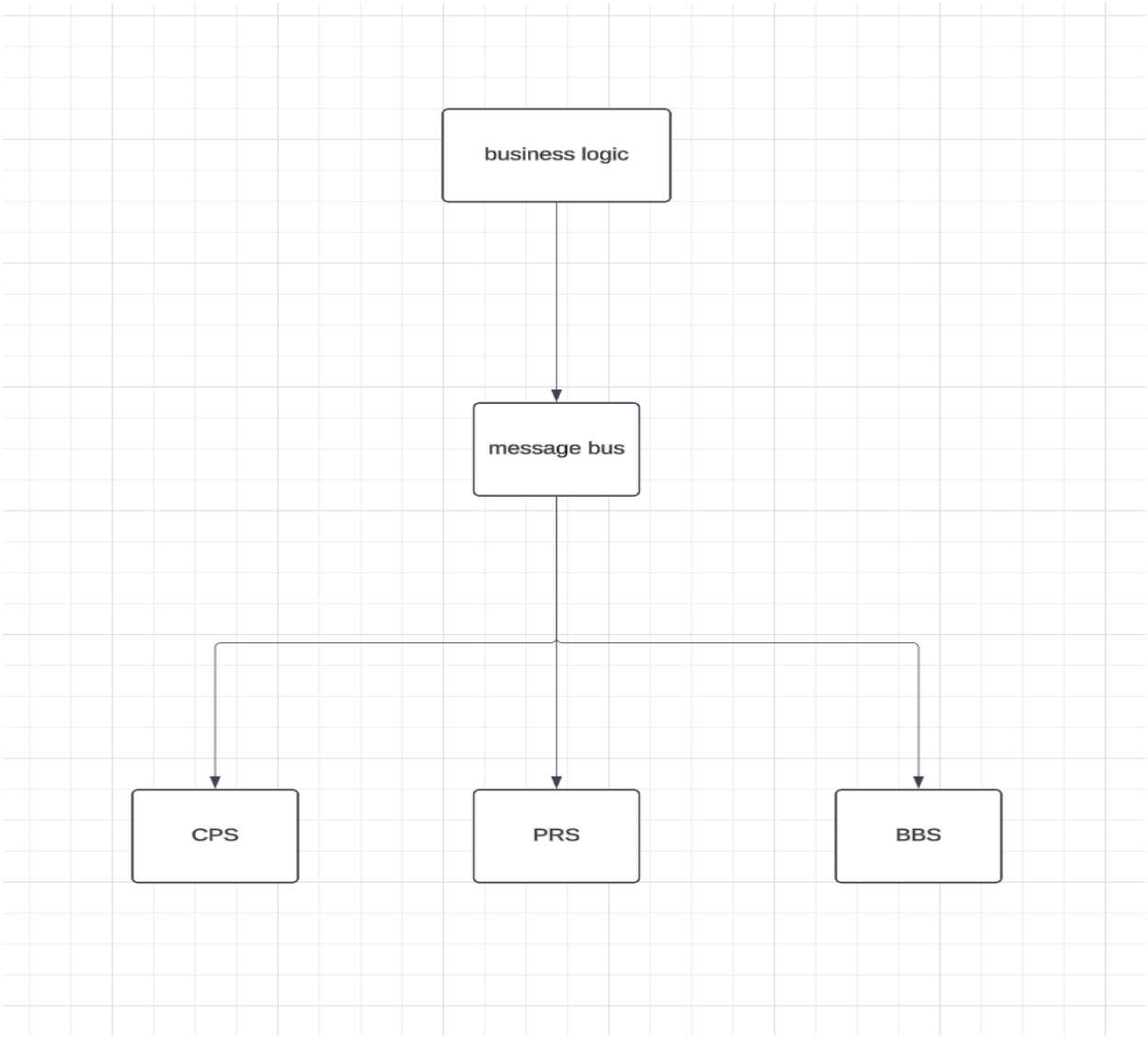
Figure 5 Dynamic Routing Pattern, Sequence diagram for services interactions

Dynamic Routing Pattern Benefits	**Dynamic Routing Pattern Liabilities**
Services do not need to deal with networking concerns or know each other's locations since all requests are handled through the message router.	Performance overhead.
Communications can be optimized as services dynamically become available or unavailable.	Single point of failure.
Efficient, predictive routing.	Potentially complex, unintuitive behavior when rules conflict.

Dow the following:

- Using the above description, select and draw the pattern that you think will best meet the messaging infrastructure requirements. Instantiate the pattern by describing the roles of its constituent participants, their responsibilities and

relationships, and the ways in which they collaborate.



- Fill out the table and identify the tradeoffs using this pattern.

Pattern	**Messaging Pattern**
Overview	**the diagram illustrates the connection between the Business Logic and the Message Bus, as well as the interactions with the systems CPS, RPS, and BBS**
Elements	Business Logic **Message Bus (CPS, RPS, BBS)**
Relations	The Business Logic communicates with the **Message Bus** to send and process requests. The **Message Bus** facilitates data exchange between the connected systems (**CPS**, **RPS**, and **BBS**).
Constraints	requiring a unified message format for compatibility

Tradeoffs	**Asynchronous communication improves efficiency and allows services to operate independently.** **Hard to predict system behavior before runtime due to asynchronous messaging.**

Applying Tactics

- **Based on your design decision for the BIzCo BIS, what quality would you like to improve?**

Availability: Ensuring the system works 24/7 and recover from failures in less than 5 min

Quality Attribute	Text Reference[i]
Availability	p. 87-95
Interoperability	p. 110-112
Modifiability	p. 121-125
Performance	p. 135-141
Security	p. 150-154
Testability	p. 164-168
Usability	p. 177-181

- **What tactics would you choose to improve your design of the messaging infrastructure for the system?**

Tactics for Improving Availability

Ping/Echo:
Check if the services are running by sending simple ping requests.

Redundancy:
Have backup systems for the Message Bus so if one fails, another takes over.

Failover:
Automatically switch to a backup system if the main system stops working.

- **Consider the tradeoffs. What are the issues associated with the selection of those tactics?**

Tactics for Improving Availability

Ping/Echo:
Check if the services are running by sending simple ping requests.

Redundancy:
Have backup systems for the Message Bus so if one fails, another takes over.

Failover:
Automatically switch to a backup system if the main system stops working.

- END -