# Scenario-Based Software Architecture Development for Hotel IT System

Course: SOF204 Software Architecture and Design Patterns - Profession: Dr.Tee Sim Hui

Name: Su Yanyu - Student ID: SWE2009513

Name: Liu Aofan - Student ID: SWE2009510

**Table of Contents**

[Scenario-Based Software Architecture Development for Hotel IT System 1](#_Toc89681928)

[1. Introduction 2](#_Toc89681929)

[2. Concrete scenarios 3](#_Toc89681930)

[3. Functionality requirements 11](#_Toc89681931)

[4. Analysis of the tactics 13](#_Toc89681932)

[5. Architecture model (Application layer & Technology layer) 20](#_Toc89681933)

[6. Software architecture views 25](#_Toc89681934)

[Integration of IoT 34](#_Toc89681935)

[1. Background for IoT in Hotel Industry 35](#_Toc89681936)

[2. Driving factors For IoT in Hotels 36](#_Toc89681937)

[3. How Hotels Can be Integrated with IoT 37](#_Toc89681938)

[References 43](#_Toc89681939)

[Contributions 44](#_Toc89681940)

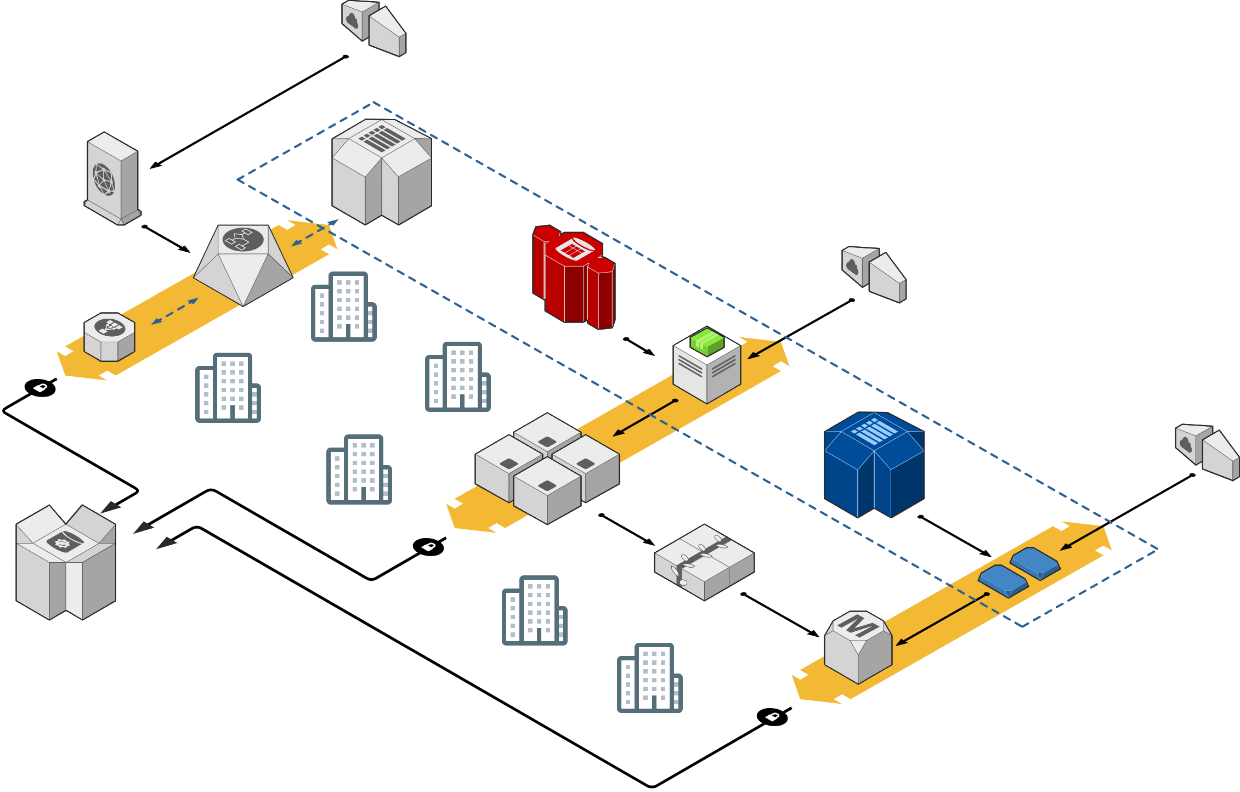
[Marking Rubrics 45](#_Toc89681941)

## Introduction

Many people have heard of software architecture, but few of them understand why we need software architecture and understand its true meaning. Good software architecture is a **blueprint** that depicts the system’s organization or structure. It almost determines everything in the IT system. The same is true for hotels.

Many hotels are huge giants with a big number of branches and countless employees. It is conceivable that, in the face of such a huge throughput, some IT systems cannot meet daily needs. Executives are bound to **upgrade IT systems** with a new architecture to meet daily needs in the end. Therefore, software architects, are supposed to establish a good software architecture to **build up the foundation** for the hotel IT system.

In addition to basic housing services, these hotels also have many corresponding supporting facilities, such as gymnasiums, data centres, logistics warehouses, and so on. A modern and international hotel is **never a single building**. The hotel is just the medium and entrance of the accommodation service. After they check-in, the user will get a series of information and personalized services.



1. Large-scale Hotel Chain

And our software architecture should be able to **adapt to these modules**. Finally, the architecture is designed to make them capable of running normally when dealing with such a large number of users. In addition, considering that our hotel has many IoT devices, integrating the organization's software architecture with the Internet of Things (IoT) and cloud systems is also needed.

## Concrete scenarios

1. **Availability:**

Scenario 1: Over the years, the hotel has stabilized its customer base and reputation, and its brand and reputation have been widely recognized by consumers. Our number of users becomes extremely large. Sometimes, especially on holidays, A large number of tourists booking hotel rooms may also order the hotel or use the hotel’s IT system. **System failure** may occur at this time, and it does **not** **provide** the service that is **consistent with its specification**. However, the system shall be able to run most time of year.

① Source of stimulus: Server unresponsive

② Stimulus: The system crashed or omit the certain query

③ Artifact: System processor (in a crash state)

④ Environment: Overloaded mode

⑤ Response: Website monitoring system records, processing personnel for emergency treatment. And being temporarily unavailable to some of the users.

⑥ Response measure: Restore the normal operation of the system in a short time (Server downtime is about 20s)

Scenario 2: When a large amount of data is queried or updated or lots of users are operating simultaneously, it will bring huge pressure to our database. This situation may occur when multiple accountants and users operate the accounting system at the same time, especially when some of them are trying to do illegal operations to the database.

① Source of stimulus: Software failure

② Stimulus: Component responded with incorrect values

③ Artifact: System processor, communication channel, and storage

④ Environment: Normal mode

⑤ Response: The system notifies the operator of the unexpected message, disables the event source causing the fault, and runs in degraded mode

⑥ Response measure: Usually, we measure the response by the availability percentage and time to repair. Unexpected messages shall not cause downtime and recover after a period of time.

1. **Modifiability:**

Scenario: The hotel industry has long-term characteristics, and many hotels have been in business for decades or even a hundred years. This puts a very high requirement on the system's modifiability. One possible situation is that developers wish to upgrade the system after a few years of operation.

① Source of stimulus: Developer

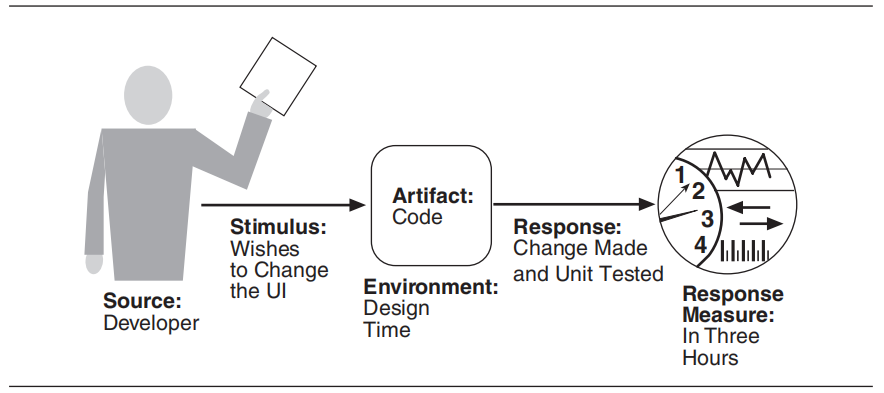
② Stimulus: Change the form of the website page, add / reduce a few functions

③ Artifact: User interface of the system. Code& Data

④ Environment: Build time& Design time

⑤ Response: The user's operation page was modified without side effects. The number and size of affected components should as small as possible.

⑥ Response measure: Complete the upgrade changes in about 25 minutes



1. Modifiability Scenario \*Adapted from Software Architecture in Practice by Len Bass
2. **Performance:**

Scenario 1: The concurrency of our system shall be as high as possible and avoid the race condition. In addition, we are not supposed to let the user wait for a long time. Normally, the server can return the result within 0.3s, but when the hotel is in a busy period, it may take 1-2s or more to return the result, which may lead to the loss of some customers.

① Source of stimulus: End users

② Stimulus: Arrival of stochastic events (e.g., User want to make a large-scale reservation of rooms, meeting rooms and restaurants)

③ Artifact: System or one or more components of the system

④ Environment: Overloaded mode

⑤ Response: A large number of transaction orders are processed at the same time

⑥ Response measure: The average waiting time for each transaction is three seconds

Scenario 2: The hotel provides services to lot of tourists during the high season. Our system can only carry a concurrent volume of 3000, but 5000 users will visit at the same time on Black Friday.

① Source of stimulus: A collection of users

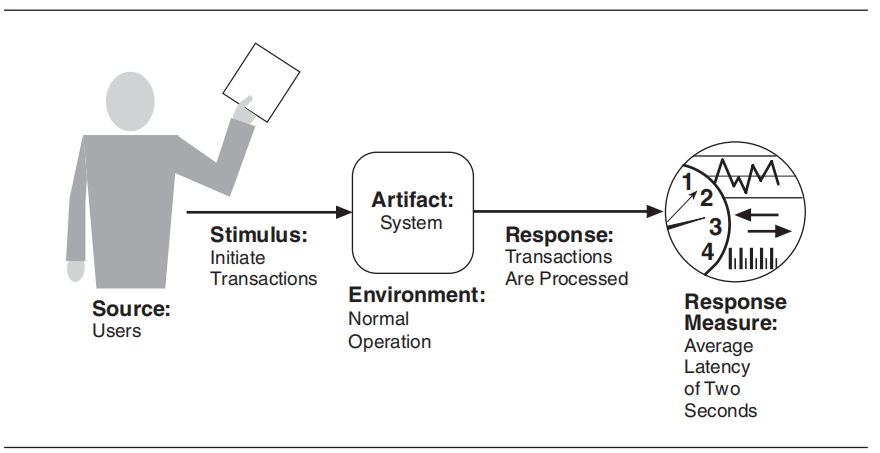
② Stimulus: The stochastic initiation of 5,000 transaction orders per minute

③ Artifact: System

④ Environment: Normal mode

⑤ Response: The system environment changes from normal mode to overload mode

⑥ Response measure: The transactions are processed with an average delay of five seconds



1. Performance Scenario \*Adapted from Software Architecture in Practice by Len Bass
2. **Security:**

Scenario 1: Hotels are also a key target of hackers. For example, in 2019, 41 Hyatt hotel payment systems in 11 countries around the world were hacked, and a large amount of essential data was leaked, including guest payment card names, card numbers, expiration dates and verification codes. The system architecture we designed may be hacked at some point.

① Source of stimulus: Unidentified hacker either from outside the organization or inside the organization.

② Stimulus: Trying to access the system database, access the system service, reduce the availability and steal users' information

③ Artifact: System data and a large amount of user information produced or consumed by the system

④ Environment: The user is offline. The system can be offline and disconnected from the network.

⑤ Response: Data and services and not allowed to be manipulated without authorization. Authenticate the identity of the visitor.

⑥Response measure: The security management system prevents the hacker from accessing system data and user information. And how much time the security department is used to find the illegal user.

Scenario 2: User's personal information has very high security requirements, and the hotel IT system and database must ensure that the relevant information of users will not be leaked. In some cases, users may need to log in and operate hotel accounts in different places.

① Source of stimulus: End users

② Stimulus: Trying to log in from different places (account authorization or operation has not been done before)

③ Artifact: User account security detection system

④ Environment: The user is online

⑤ Response: Send remote login security verification to the mobile phone number bound to the corresponding account

⑥ Response measure: The system successfully verified and authorized the login rights of user accounts (in a short time)

图示

描述已自动生成

1. Security Scenario \*Adapted from Software Architecture in Practice by Len Bass
2. **Testability:**

Scenario 1: After the IT system development is over, invite professional testing teams or internal testers to conduct various tests including unit testing and black box testing. Tester performs tests on a system component to be executed.

① Source of stimulus: Unit testers, acceptance testers and system testers, even automated testers.

② Stimulus: Perform tests on the system function

③ Artifact: A function of the central reservation system

④ Environment: Although test can happen in development time and compile time, but most of them are happen after finishing the development and when the system in running.

⑤ Response: The system provides an interface for controlling its behaviour and observing its output.

⑥ Response measure: One kind of measure can be the effort involved in finding fault with the system. For example, 75% path coverage can be achieved within two hours without any fault.

Scenario 2: The client tests the completion of the room cleaning service (Property Management System)

① Source of stimulus: The client

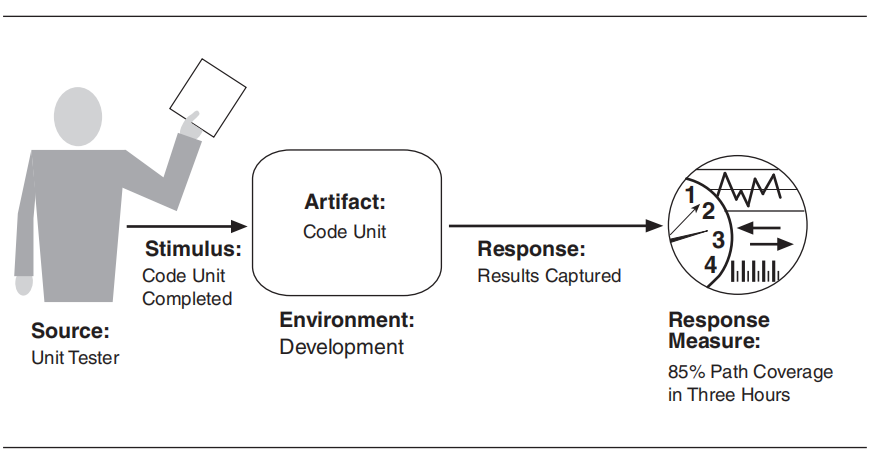
② Stimulus: Perform tests on the system through some operations by sensors

③ Artifact: The whole cleaning service function module

④ Environment: At deployment time

⑤ Response: The system is ready to provide the test environment and can provide the test result of hotel room cleaning service.

⑥ Response measure: Whether each part of the test report is completely generated if there is a 2% probability that the system will fail.



1. Testability Scenario \*Adapted from Software Architecture in Practice by Len Bass
2. **Usability:**

Scenario 1: The user is the always the source of the stimulus for usability. For example, a user may book a room by mistake.

① Source of stimulus: End user with specialized role

② Stimulus: The user wants to cancel room reservation

③ Artifact: System

④ Environment: At runtime

⑤ Response: The system deletes reservations from the server and informs the user.

⑥ Response measure: Cancellation takes less than two seconds.

Scenario 2: The user finished eating in the hotel restaurant and wanted to settle the meal fee through the hotel payment system.

① Source of stimulus: End user

② Stimulus: The user wants to make payment

③ Artifact: System

④ Environment: At runtime

⑤ Response: Wishes to complete the payment quickly

⑥ Response measure: Payment takes less than one second

Scenario 3: When using the hotel's online shopping mall, the user wants to sort the peripheral products according to specific conditions (price or sales volume) to facilitate retrieval and comparison.

① Source of stimulus: End user

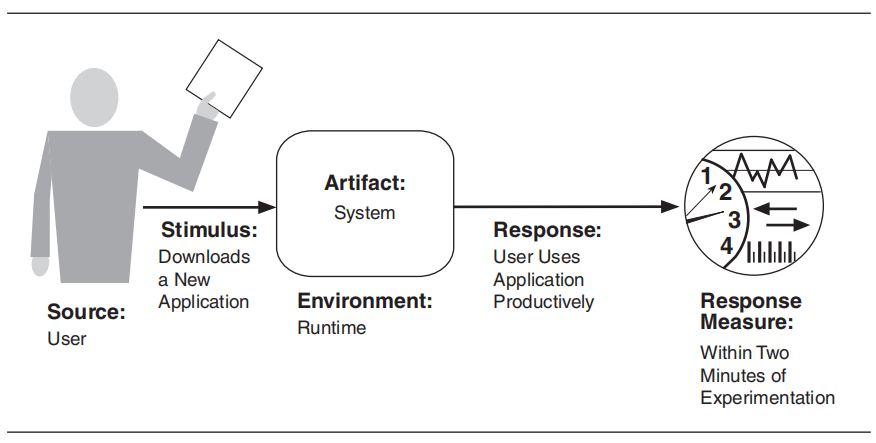
② Stimulus: The user wants to rearrange the commodity list (the price is from low to high, and the sales volume is from high to low)

③ Artifact: System

④ Environment: At runtime

⑤ Response: The web page successfully arranges and displays the commodity list as required

⑥ Response measure: The system meets the requirements and standards required by the user (with normal response time)



1. Usability Scenario \*Adapted from Software Architecture in Practice by Len Bass

## Functionality requirements

Defined as the function of a software system or component. Functional requirements are the functions and services that a system needs to provide. It can use a combination of inputs, behaviours, and outputs to express what the system hopes to achieve.

Here are the functional requirements for our hotel scenario software architecture:

1. All users will have access to the IT system using a unique user ID and corresponding password.
2. When the user enters the correct login credentials, the user's personal information will be loaded and displayed.
3. The IT system shall allow user to Log in with Google and Facebook account and its verification.
4. Personal account maintenance system shall allow users to set and modify personal information.
5. The IT system automatically validates customers against the Contact Management System.
6. Only registered employees have the right to view patient’s data.
7. The personal account maintenance system must have a mechanism to help users to find their passwords in case they forget their passwords.
8. Revenue data are hidden to all user except Only managerial level employees
9. When the user clicks the "Change Password" button, and options dialog box shall appear which enables the user to change his password.
10. The system shall be able to support the following language: English, Chinese, Japanese, and Korean will plug-in support for other languages.
11. The software system should be integrated with banking API
12. The central reservation system (CRS) should e-mail users when the hotel has accepted the reservation.
13. The POS and CRS shall support the following payment methods: PayPal, Alipay, Visa, eNETS and etc.
14. The central reservation system (CRS) is supposed to assign a unique order identifier (Order ID) to each transaction order, as well as room number.
15. The system must ensure that each transaction order is recorded and can be queried through the order ID.
16. The property management system (PMS) needs to be able to spell-check documents and provide autocorrect facilities.
17. The accounting system (AC) should calculate the accommodation, catering, and service costs of users.
18. When the user clicks the “Confirm” button, the details in the text fields will be captured and stored in the database.
19. The system shall allow users to query and print personal transaction records in the hotel.
20. The system should authenticate visitors to determine their access content and restrictions.
21. The system should allow staff to search and modify data in the database.
22. The human resource management system (HRMS) should allocate hotel staff reasonably to ensure that services can be completed.
23. The PICC system should be able to timely detect and report the shortage of disposable articles in hotel rooms.
24. The POS system should allow users to order meals online.
25. When the user presses the "Need to clean" button on the wall, the system must load the request and assign a cleaner to clean the room.

## Analysis of the tactics

1. **Availability tactics**: Tactics to achieve availability can be divided into fault detection, fault recovery, and fault prevention.
2. **Fault detection:**

Three commonly used tactics for detecting faults are ping/echo, heartbeat, and exceptions. Among them, **ping/echo** refers to the asynchronous request/response message pair exchanged between nodes, which is used to determine the reachability and round-trip delay through relevant network paths.

The concept of **heartbeat** tactics is similar to that of system monitor: one component sends out heartbeat messages periodically, and the other component listens to it. If the heartbeat fails, the initiating component is considered to have failed, and the fault correction component is notified.

**Exception** is easy to understand. When the system detects a certain fault, it will throw an exception, and then the exception handler will be executed in the same process that introduced the exception (Bosch and Molin, n.d.).

1. **Fault recovery:**

The fault recovery tactics can generally be refined into preparation for **recovery** (**preparation and repair**) and **reintroduction**. The preparation for recovery tactics includes the following things:

* Active / passive redundancy
* Exception handling
* Ignore faulty behaviour
* Retry
* Software upgrade
* Spare, rollback
* The degradation and reconfiguration

The **reintroduction** tactics include state resynchronization, non-stop forwarding (NSF), escalating restart and the shadow tactic. Among them, non-stop forwarding is a concept that originated in router design. Functionality is separated into two parts in the design, forwarding plane (data plane) and supervisory. NSF can guarantee that when the system fails for some reason, the forwarding plane (service) will not be interrupted during the active-standby switching process. In addition, after the system is restored, the device can re-establish the neighbour relationship and rebuild the routing table according to the routing information. (Capilla et al., 2016). Usually, after a router fails, neighbours at the routing protocol level will detect that the neighbour relationship between them is Down and then Up again after a while. This process is called neighbour relationship oscillation. This kind of oscillation will eventually lead to the emergence of routing oscillation, which will cause a black hole in routing when the router is restarted for a period of time, thus greatly reducing the reliability of the network. Non-stop forwarding is a technology to solve the above-mentioned routing oscillation problem.

1. **Fault prevention:**

Generally speaking, fault prevention tactics include four methods: **transactions**, **process monitor**, **removal from service**, and **exception prevention**. It’s worth mentioning that service removal tactics refer to removing a component of the system from operation (putting it in the service stop state) to carry out some activities to alleviate the potential system failure. The mechanism of process monitor is that once the failure in the process is detected, the monitoring process can delete the process that is not executed and create a new instance that is initialized to the appropriate state in the standby policy (CUNNINGHAM and SCOTT, 2004).

1. **Modifiability tactics**: Tactics for improving modifiability can be divided into localized modification, preventing ripple effects and deferring binding time.
2. **Increasing Cohesion - Localize modification:**

There are three ways of localization modification as follows: **increasing cohesion**, abstracting common services, and maintaining semantic coherence. In essence, the tactics reduce the cost by improving the cohesion of functions and limiting the modification to smaller module groups. During the design process, the policy assigns responsibilities to modules so that the expected changes are limited in scope (Dobrica and Niemela, 2002).

1. **Reduce Coupling - Prevent ripple effects:**

The **ripple effect** caused by modification leads to the necessity of making changes to modules not directly affected by it. Software architects commonly **reduce the coupling** between modules to prevent or mitigate the ripple effect (Jaiswal, 2019). There are many ways to reduce coupling, such as using encapsulation, raising the abstraction level, or using an intermediary and restricting communication path, which may effectively improve the modifiability of the system.

1. **Defer binding time:**

The concrete scenarios of modifiability sometimes include "allowing non-developers to modify", and the tactics of deferring binding time can support such scenarios when the system has the additional infrastructure, which is also an excellent tactic to improve the modifiability of the system (Kazman, Abowd, Bass and Clements, 1996).

1. **Performance tactics**: The goal of performance tactic is to generate a response to an event arriving at the system within some time-based constraint. Generally speaking, the response time can be divided into two parts: **processing time** (i.e., the system working to get the response) and **blocked time** (i.e., the user wait for the system to process his query).s

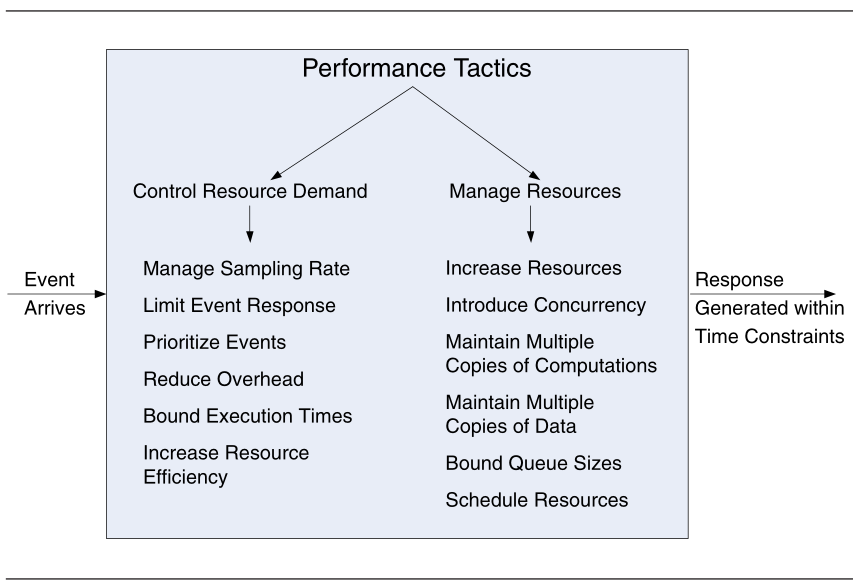
With this background, tactics to achieve performance can be divided into resource demand, resource management, and resource arbitration (Kruchten, Obbink and Stafford, 2006).

1. **Control Resource Demand:**

Simply put, the resource demand tactics include increasing computation efficiency, reducing computational overhead, controlling the frequency of sampling, and managing event rates. The two characteristics of demand are the number of resources consumed by each request and the time interval between events in the resource stream (the frequency of making requests in a stream). Taking increasing computation efficiency as an example, an important step in information processing is to apply an algorithm, and the improved algorithm will reduce the delay and improve the performance (Medvidovic and Taylor, 2000).

1. **Resource management:**

Increasing available resources, introducing concurrency, and maintaining multiple copies of computations are the most practical methods to manage resources. Take concurrency as an example: parallel processing of requests by the system can effectively reduce blocking time. Concurrency is generally introduced by creating additional threads to handle different active sets. In addition, properly allocating threads to resources (load balancing) can make the best use of concurrency (Muccini, Bertolino and Inverardi, 2004).



1. Performance Tactics \*Adapted from Software Architecture in Practice by Len Bass
2. **Resource arbitration:**

The core idea of resource arbitration is resource scheduling. When resource competition occurs, only by knowing the usage characteristics of each resource can we choose the compatible scheduling policy. (Perry and Wolf, 1992) Common scheduling policies include fixed-priority scheduling, dynamic priority scheduling, FIFO (First in First Out), and static scheduling. By applying the reasonable scheduling strategy to resources, we can plan resource allocation better, to achieve the best performance.

1. **Security tactics:** Tactics for improving security can be divided into detecting attacks, resisting attacks, reacting to attacks, and recovering from attacks (Taylor and Taylor, 2021).
2. **Detect attacks:**

Generally speaking, attack detection includes four aspects: intrusion detection, service denial detection, message integrity verification, and message delay detection. But it is arbitrary that "message delay detection" is included in the original set of detecting some kind of attack. Therefore, the collection of methods for detecting attacks should be replaced by intrusion identify, service denial detection, message integrity verification, and storage integrity verification.

1. **Resist attacks:**

Identifying subjects, authenticating subjects, authorizing subjects, limiting access, and separating entities are typical methods to resist attacks. However, as the tactics cannot be too general or too specific, the methods are modified as follows: authenticate subjects, authorize subjects, detect the origin of the message, manage security data and establish a secure channel. In addition, it may be a good choice to rename the tactic “Resist attacks” as “Stop or mitigate attacks” (Valipour, Amirzafari, Maleki and Daneshpour, 2009).

1. **React to attacks:**

Tactics for reacting to attacks generally include alerting subjects and applying institutional policies, which can effectively improve the security quality attribute.

1. **Recover from attacks:**

Same as above, tactics to recover from attacks also include applying institutional policies, together with another method “auditing actions”.

1. **Testability tactics**: Tactics to achieve testability can be divided into adding controllability and observability to the system state and limiting complexity in the system’s design.
2. **Control and observe system state:**

Control and observation of the system go hand-in-hand, which usually includes six kinds of tactics: **record / playback**, separate interface from implementation (**specialized interfaces**), **localize state storage**, **sandbox**, **abstract data sources** and **executable assertions**. Take the sandbox, which can isolate the system from the real world, as an example. The most common use form of sandboxing is to virtualize resources. The virtualizations of sandbox could be done for many different kinds of resources, such as system time, network and memory. Therefore, the experiment is not bound by the fear of revoking the experimental results (van Lamsweerde, 2003). It can be said that the sandbox tactic effectively enhances the testability of the system.

1. **Limit complexity:**

Complex software systems are difficult to test, so we need to reduce the complexity of the system as much as possible to ensure that system faults or failures can be tested and removed. Limiting complexity in the system’s design has two tactics, namely **limit structural complexity** and **limit nondeterminism**. Let’s take Limiting nondeterminism as an example: As we all know, Nondeterministic system is definitely harder to test than the deterministic system, so this tactic aims to find the sources of nondeterminism in the system and clear them away, thus increasing the testability of the system.

1. **Usability tactics**: Recall that usability is concerned with how easy it is for users to complete expected tasks and the type of support the system provides for users. Tactics for improving usability can be divided into runtime and design time.
2. **Support User Initiative – Runtime Usability:**

When the system is running, the usability will be effectively enhanced by **providing users with feedback** about what the system is doing and the ability to issue commands based on usability. The process of human-computer interaction can be divided into three categories: system initiative, user initiative, and mixed-initiative. System initiative tactics are a strategy used to predict the model of self-behaviour or user's intention, which includes maintaining a model of the system, maintaining a model of the user, and maintaining the model of the task. Take the task model as an example: the model is used to determine the context so that the system can understand the user's intention and provide relevant help, which undoubtedly greatly improves the usability of the system (van Vliet and Tang, 2016).

1. **Support System Initiative - Design time Usability:**

Developers will complete the implementation according to the User Interface (UI) design changes provided by usability engineers (Van Vliet and Tang, 2016). When the system take initiative, a **model of the user** (i.e., the tasks undertaken by the users or the system state itself) shall be provided to support its behaviour. Under normal circumstances, we may build a model that predicts our own behaviour or user intentions

This leads to the improvement of the modifiability tactics of semantic coherence-design-time tactics. The tactics separate the application program from the user interface, to ensure that the code of the user interface can be independently maintained and localized when the user interface changes frequently during development.

## Architecture model (Application layer & Technology layer)

1. **Application Layer:**
2. Application architecture view

**Documentation:** This view plays a very important role in designing or understanding the main structure and related data of the application and its subcomponents. The application architecture view can be used to decompose the structure of the application system being built to clarify the sub-components of the system and the application services (or application program interfaces) provided by them.

图示

描述已自动生成

1. Application structure view

**Documentation:** This view relates the subcomponents and data objects in application component integration by writing (access) and shows the management and operation of the four subcomponents of the property management system (PMS) for related data.

图示

描述已自动生成

1. Application usage view

**Documentation:** This view is customer-centric and focused on customer experience. This method with "service design" as its axis focuses on the development of services to be designed from the outside to the inside, highlighting the direct or indirect value created by products and services for customers and the organization itself. Application usage view spans multiple application services and is commonly used for visualization of customer value stream. At the same time, this view also shows how applications work together to support the whole business process, which can be used to identify or describe services required by business processes and other applications.

图示

描述已自动生成

1. Components of Application Layer

表格

描述已自动生成图形用户界面, 文本

中度可信度描述已自动生成

1. **Technology Layer:**
2. Technical infrastructure view

**Documentation:** The technical infrastructure view represents the application platform and can be used to simulate the configuration of the runtime environment and the deployment of business applications. The view contains elements such as application components, system software, technical services and communication networks, which together constitute a complete infrastructure.

图示

描述已自动生成

1. Technical generic view

**Documentation:** This view, as a simplified view of the overall structure of the organization, can be applied to the context diagram of programs, specific services or projects. The components or services in the view are related through the relationships of realization, composition and serving, and the structure and operation mode of the organization are abstracted.

图示

描述已自动生成

1. Components of Technology Layer

图形用户界面, 文本, 应用程序

描述已自动生成文本

描述已自动生成

## Software architecture views

1. **Achievement Realization View:**

**图示

描述已自动生成**

**Documentation:** This view simulates the achievement realization process of hotel group to reduce operation costs and maintenance costs. The realization view contains two important artifacts, the hotel logistics department and the customer relationship management department and realizes related services through the application components associated with them, so as to achieve the final result (outcome) of cost reduction.

1. **Application Architecture View:**

**图示

描述已自动生成**

**Documentation:** This view plays a very important role in designing or understanding the main structure and related data of the application and its subcomponents. The application architecture view can be used to decompose the structure of the application system being built to clarify the sub-components of the system and the application services (or application program interfaces) provided by them.

1. **Application Structure View:**

**图示

描述已自动生成**

**Documentation:** This view relates the subcomponents and data objects in application component integration by writing (access) and shows the management and operation of the four subcomponents of the property management system (PMS) for related data.

1. **Application Usage View:**

图示

描述已自动生成

**Documentation:** This view is customer-centric and focused on customer experience. This method with "service design" as its axis focuses on the development of services to be designed from the outside to the inside, highlighting the direct or indirect value created by products and services for customers and the organization itself. Application usage view spans multiple application services and is commonly used for visualization of customer value stream. At the same time, this view also shows how applications work together to support the whole business process, which can be used to identify or describe services required by business processes and other applications.

1. **Technical generic view:**

**图示

描述已自动生成**

**Documentation:** This view, as a simplified view of the overall structure of the organization, can be applied to the context diagram of programs, specific services or projects. The components or services in the view are related through the relationships of realization, composition and serving, and the structure and operation mode of the organization are abstracted.

1. **Goal Realization View:**

**图示

描述已自动生成**

**Documentation:** This view can refine high-level goals into lower-level and more specific goals and decompose these specific goals into requirements or constraints. The goal realization view usually uses "aggregation" to model the sub-goals generated by the refinement of the goal.

1. **Organization Structure View:**

**图形用户界面

描述已自动生成**

**Documentation:** This is the organization structure view of Property Management System. This view commonly represents the overall structure of an organization or group in a nested manner, which is used to determine the capabilities and related responsibilities of the organizational unit. Its scope is usually in a single level or aspect, including elements such as business participants, business roles, business collaboration and business interfaces.

1. **Organization Tree View:**

**图示

描述已自动生成**

**Documentation:** This is the organization tree view of the property management system (PMS). On the basis of the organization structure view, this view connects the business actors in various departments of the hotel through the representation of tree diagram, showing the business relationship of the organization / group.

1. **Strategic Value View:**

**图示

低可信度描述已自动生成**

**Documentation:** This view is used to visualize organizational policies. The strategic value view contains strategic value elements, from which all development activities in the organization can be directly or indirectly derived. By visualizing the strategic value, we can trace other elements related to the actual strategic implementation and reach a connection with the reality.

1. **Strategic View:**

**图示

描述已自动生成**

**Documentation:** This view models the organization's business strategy through such elements as "capability", "goals" and "strategy", thus establishing the relationship between the organization's objectives and strategies and linking them to the enterprise architecture through "capability". This view can be applied to "Goal-based Strategic Model", in which "Goals" constitute a hierarchical structure to decompose high-level goals into lower-level goals.

1. **Technical Infrastructure View:**

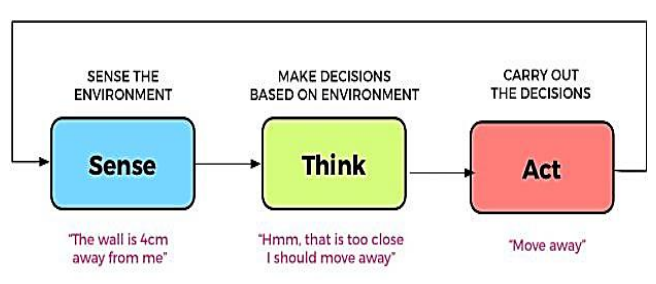
图示

描述已自动生成

**Documentation:** The technical infrastructure view represents the application platform and can be used to simulate the configuration of the runtime environment and the deployment of business applications. The view contains elements such as application components, system software, technical services and communication networks, which together constitute a complete infrastructure.

# Integration of IoT

In the IT field, the Internet of Things has received extensive attention from the government, academia, and industry because of its huge application prospects. It is the extension and expansion of the Internet to the physical world. The Internet can be used as one of the important ways to transmit information about the Internet of Things, whereas sensor networks are based on self-organizing networks.



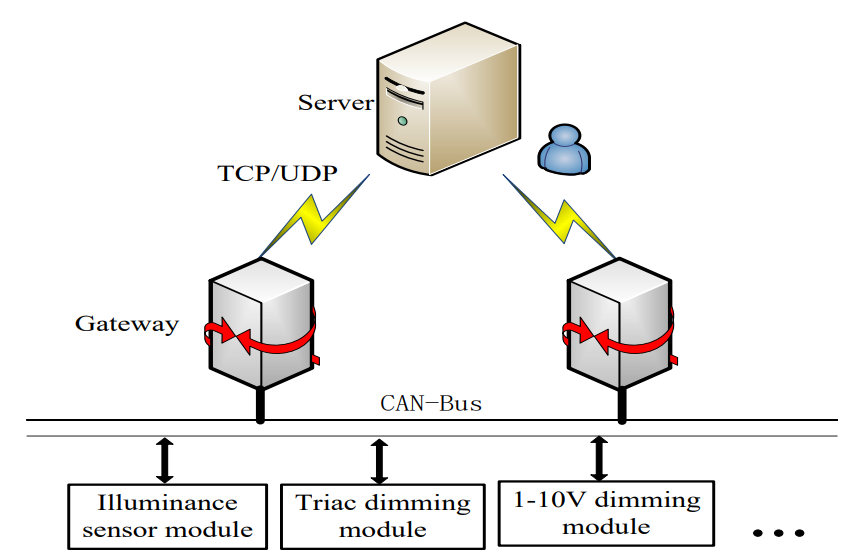
1. An Exmaple Scenario of Smart Hotel

As an important perception technology, sensor networks, as a key technology of the Internet of Things, have finally been widely used in the hotel industry, and related applications are represented by environmental monitoring. Any management system is a concentration of management theories and methods in this field. With the development of hotel management technology and practices, hotel IT systems are constantly changing, upgrading and optimizing. Some more advanced hotel groups may have hundreds of functional modules in their management system, which is extremely complicated.

## Background for IoT in Hotel Industry

As a hot topic in recent years, the Internet of Things (i.e., hereby and after we use abbreviation IoT) is gradually entering people's daily life and making people's lives more comfortable. Meanwhile, the hotel, an industry that is undergoing a transformation from a traditional industry to an Internet-based industry, urgently needs the influx of new technologies to provide better services.

The number of connected IoT devices has reached 7 billion in 2018 and is expected to grow to 21.5 billion by 2025. Therefore, as an important part of the construction of the information highway, IoT can provide infrastructure support for the digital transformation of hotels, and also provide strong data support for subsequent enterprise-level data operations.



1. Overall Architecture of IoT Hotel

IoT perfectly solves the smart hotel industry with two key points, one is product stability, and the other is energy saving. To this end, it is a very good solution to build a full-stack technology framework with an IoT cloud connection platform as the core.

## Driving factors For IoT in Hotels

With the functions and architecture of hotel group management systems are constantly updated and iteratively upgraded, IoT is the general trend. The driving factors of hotel integration with IoT are as follows:

1. The old IT architecture is **difficult to support** new functional modules. Mobile Internet and Internet of Things (IoT) have redefined the hotel management mode, so the hotel management software needs to add new modules to the original system architecture, which may lead to the gradual weakening of the support of the original underlying architecture for the upper applications, further resulting in untimely response and slow running efficiency.
2. In today's business environment, 94% of organizations implementing IoT solutions have seen a return on their IoT investment (Bosch and Molin, n.d.). Through real-time monitoring and tracking of daily assets given by the IoT devices, companies can provide customers with services ranging from shorter waiting times to cleaner and more energy-efficient environments to asset tracking of machines and equipment.
3. Management and control brought by hotel collectivization. The trend of hotel grouping is becoming more and more obvious, and the hotel group is gradually strengthening the control of its subordinate hotels, while IoT can make the group control easier to realize by applying the way such as Identity token card, NFC smart door lock (Taylor, 2021).

## How Hotels Can be Integrated with IoT

1. **Traffic Control**

Compliance with the government’s **social distancing regulations** and the implementation of new **restrictions on the number** of guests in its hotels have become mandatory measures since the new crown pneumonia epidemic. Traditional methods of manpower counting are not only inaccurate, but also costly.

The IoT-based people counting system can help track the number of people entering and leaving any given space in real time, with almost no counting errors, and reminding managers when the capacity threshold is reached. This is very important for hotels

1. **Indoor Environmental Quality Monitoring**

The COVID-19 outbreak has made people's concerns about sanitation and clean air in hotels and conference centers a top priority. **Indoor Environmental Quality (IEQ)** has become the top priority of hotel design. In the past, the main focus of indoor environmental quality was to reduce energy consumption, but it has turned to ensure that these spaces have clean air to breathe, and that the ambient temperature and light are the best for guest comfort. Whereas wireless IoT sensors play a vital role in IEQ monitoring-in-depth understanding of these key indoor climate factors can ensure the health and comfort of guests and employees.

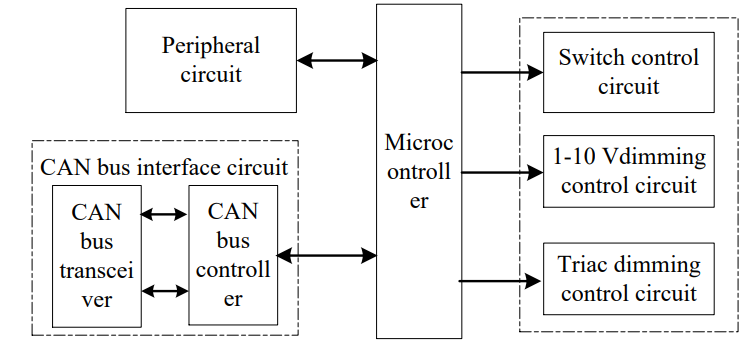
1. IoT-based air purifiers system can bring a fresher room environment to smokers.

2.The standard configuration of the room can be linked with the humidifier through the temperature and humidity sensor to realize the automatic adjustment of the room humidity.

1. **Energy Consumption**

As the second largest expenditure category of hotel industry employees, **energy expenditure** is a very important aspect for hotels. The application of multiple sensors in the Internet of Things can play an important role in **saving energy**.

In the conference hall, the sensor in the room can control the lights to automatically turn off, and the air-conditioning will be turned off after people leave. In addition, the configuration of energy-saving system can cooperate with air-conditioning sensors and window openers, so that the hotel air-conditioning can achieve energy-saving effects and so on. Here is a graph showing the hardware structure in IoT in hotels.



1. Hardware Structure of Control Part

\*Adapted From Lighting Control System for Smart Hotel Rooms by Jiajia Feng

These things are difficult to accomplish with one company, so it is extremely important to choose to establish alliances and cooperate with industry partners. The core of the hotel is to provide services to guests, and services can bring entrances, such as smart speakers, TV screens, and mobile phones. Through the integration of the plan and the cooperation of cross-border partners, the entrance and conversion can be opened up to bring value to customers.

1. **Fast billing**

Through the **detection and sensing facilities** based on IoT devices, the items sold in the room, such as instant noodles and mineral water, need to be paid in advance or clearly calculated into the bill when checking out of the hotel.

This can effectively prevent customers from evading and crediting them, and at the same time record the room The degree of damage to the items. If the items in the room are damaged, you can seek compensation from the customer through legal means.

1. **Data Analysis**

The Internet of Things devices can **record a lot of hotel information**. By **analysing this information**, the hotel can quickly understand the shortcomings. For example, services can be analysed, and hoteliers can also track the process at certain points. The collected data can be effectively evaluated through the system, allowing executives to provide useful insights and encouraging them to adjust existing services. This in turn enables them to enhance service and delivery options and enhance the profitability of the hotel.

1. **Quick Check-in**

With the widely application of IoT devices such as **RFID technology**, guests can quickly perform **identity authentication** and check in, for example, through ID card recognition module or online authentication module. Moreover, guests often forget to adjust the temperature of the room, even forget to turn off the TV or lights. The IoT solution can automatically help them with adjusting the temperature, turning off the lights and the TV when the guest leaves the room.

In addition, when conferences and other large-scale events are held in hotels, such as weddings and conferences, **long registration** and **badge verification** processes are required, as well as effective queue management of people in line. VIP guests expect a convenient and consistent experience and provide very fast service. They don't want to line up for a long time. The **IoT solution** of **biometric tokens** supporting facial recognition and queue management algorithms is very useful at this time.

1. **Contactless Payment**

Nearly three-quarters of customers agree that it is important for companies to provide **contactless payment options**. Under the current circumstances, many guests are worried about whether they will be **infected with COVID-19** from contact with employees. They very much hope that they can ensure their own safety to the greatest extent.

Therefore, the use of RFID based on the Internet of Things or other contactless payment methods will be very important. At the same time, upgrading to a digital point-of-sale system for contactless payment during checkout will resonate well with customers.

1. **Room Automation**

**Room automation** is another great idea for the Internet of Things in the hotel industry. This emerging solution can provide customers with fully customized services.

Some hotels will provide control terminals like mobile phones or tablets, and guests can control many functions of the room through these devices. These devices allow users to remotely operate heating and lighting systems, and even call attendants. Even some smart devices can carry out basic voice communication with customers and provide personalized services.

In addition, **branch hotels** opened by hotel chain companies in different parts of the world can also share guest data in CMS to ensure that guests experience the best in chain hotels.

1. **Mobile participation**

Now, mobile devices have become the world’s largest medium. More than 85% of the world’s population have at least one mobile device. Therefore, it is very important to allow guests to use mobile as a medium in hotels.

For example, we can use the Internet of Things NFC devices to allow guests to use their mobile phones as the key to the room or make a request to the front desk. At the same time, EAM or CMMS-based applications can be used as the backing of the hotel's internal tools.

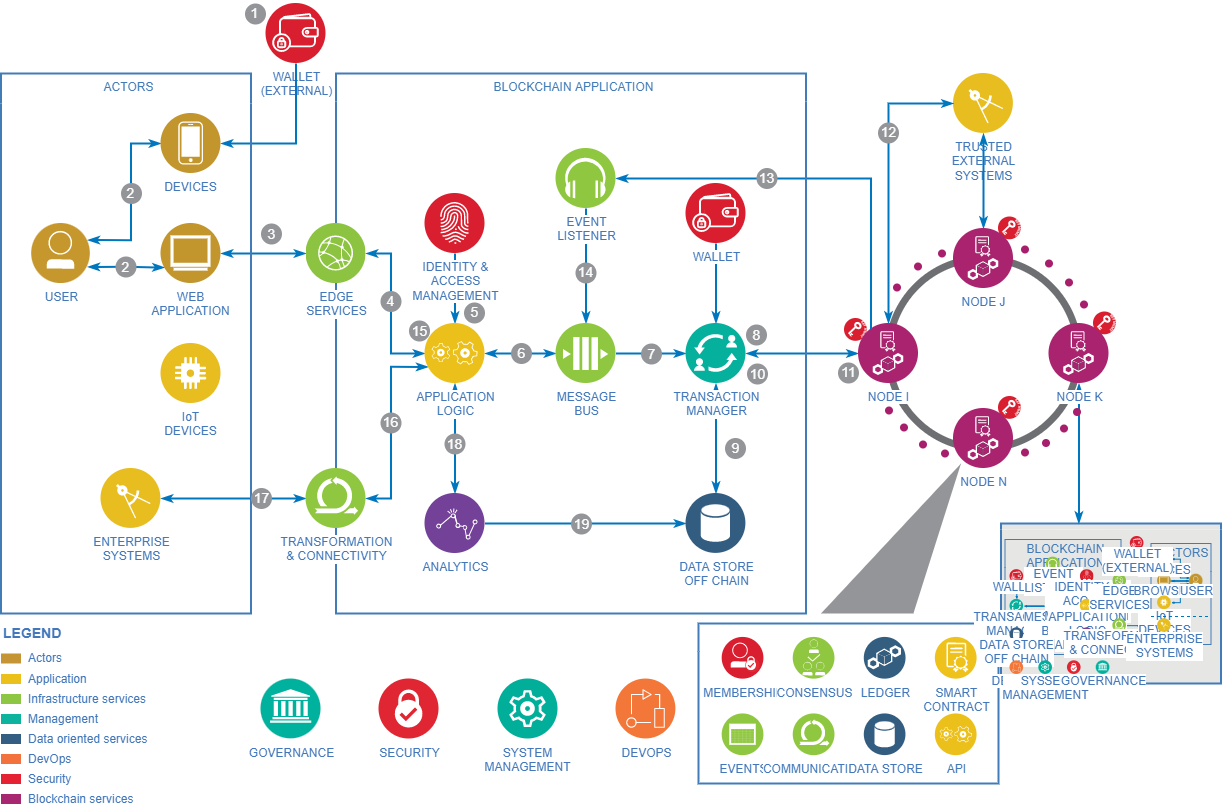
1. **Predictive Maintenance**

IoT can also track the maintenance needs of devices such as refrigerators and air conditioners. Predictive maintenance not only saves money, but also helps companies avoid equipment failures. Outside the hotel, IoT sensors can monitor the moisture in the landscape soil and detect water leakage.

In addition, varieties of modules that hotels already have can play a more powerful role with IoT:

1. PMS (Property Management System)
2. POS (Point Of Sale)
3. CRS (Central Reservation System)
4. AC (Accounting System)
5. PICC (Purchasing / Inventory / Cost Control)
6. HRMS (Human Resources Management System).

Finally, hotel companies like Best Western have been one of the early adopters of voice-controlled customer service. Voice Control, as an important application area of IoT, may expand substantially in the next five years. Here is a graph illustration of how these modules work with IoT devices.



1. An Example of IoT Based Hotel

**References**

Bosch, J. and Molin, P., n.d. Software architecture design: evaluation and transformation. *Proceedings ECBS'99. IEEE Conference and Workshop on Engineering of Computer-Based Systems*.

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., and Babar, M., 2016. 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, 116, pp.191-205.

CUNNINGHAM, H. and SCOTT, D., 2004. Software Architecture for Language Engineering. *Natural Language Engineering*, 10(3-4), pp.205-209.

Dobrica, L. and Niemela, E., 2002. A survey on software architecture analysis methods. *IEEE Transactions on Software Engineering*, 28(7), pp.638-653.

Jaiswal, M., 2019. Software Architecture and Software Design. *SSRN Electronic Journal*.

Kazman, R., Abowd, G., Bass, L. and Clements, P., 1996. Scenario-based analysis of software architecture. *IEEE Software*, 13(6), pp.47-55.

Kruchten, P., Obbink, H. and Stafford, J., 2006. The Past, Present, and Future for Software Architecture. *IEEE Software*, 23(2), pp.22-30.

Medvidovic, N. and Taylor, R., 2000. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1), pp.70-93.

Muccini, H., Bertolino, A. and Inverardi, P., 2004. Using software architecture for code testing. *IEEE Transactions on Software Engineering*, 30(3), pp.160-171.

Perry, D. and Wolf, A., 1992. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4), pp.40-52.

Taylor, J. and Taylor, W., 2021. Software Architecture. *Patterns in the Machine*, pp.63-82.

Valipour, M., Amirzafari, B., Maleki, K. and Daneshpour, N., 2009. A brief survey of software architecture concepts and service-oriented architecture. *2009 2nd IEEE International Conference on Computer Science and Information Technology*,

van Lamsweerde, A., 2003. From System Goals to Software Architecture. *Lecture Notes in Computer Science*, pp.25-43.

van Vliet, H. and Tang, A., 2016. Decision making in software architecture. *Journal of Systems and Software*, 117, pp.638-644.

**Appendix1**

**Contributions**

|  |  |
| --- | --- |
| **Name** | **Contribution** |
| **Liu Aofan** | 1. Write-Concrete scenarios  2. Check-Functionality requirements  3. Write-Detailed analysis of the tactics  4. Check-Archi modelling tool - 1-4 out of 8  5. Write-Archi modelling tool - 5-8 out of 8  6. Check-Description and Analysis of IoT Integration |
| **Su Yanyu** | 1. Check-Concrete scenarios  2. Write-Functionality requirements  3. Check-Detailed analysis of the tactics  4. Write-Archi modelling tool - 1-4 out of 8  5. Check-Archi modelling tool - 5-8 out of 8  6. Write-Description and Analysis of IoT Integration |

**Appendix2**

**Marking Rubrics**

**\*Note: the criteria numbers are corresponding to the numbers listed in Section E of this document**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Component Title** | **Project** | | | | **Percentage (%)** | **55** | |
| **Criteria** | **Score and Descriptors** | | | | | **Weight (%)** | **Marks** |
| **Excellent**  **(5)** | **Good**  **(4)** | **Average**  **(3)** | **Need Improvement**  **(2)** | **Poor**  **(1)** |
| **1** | Relevant and comprehensive concrete scenarios with no mistakes | Relevant and comprehensive concrete scenarios with a few minor mistakes | Adequate and relevant concrete scenario with some major mistakes | Inadequate but relevant concrete scenario with some major mistakes | Inadequate and irrelevant concrete scenario | **5** |  |
| **2** | Detailed and comprehensive analysis, design, and modelling with no or minimal mistakes | Comprehensive analysis, design, and modelling with some mistakes | Adequate analysis, design, and modelling with some major mistakes | Adequate analysis, design, and modelling with many major mistakes | Inadequate analysis, design, and modelling with substantial major mistakes | 4 |  |
| **3** | Detailed and comprehensive analysis, design, and modelling with no or minimal mistakes | Comprehensive analysis, design, and modelling with some mistakes | Adequate analysis, design, and modelling with some major mistakes | Adequate analysis, design, and modelling with many major mistakes | Inadequate analysis, design, and modelling with substantial major mistakes | 6 |  |
| **4** | Detailed and comprehensive analysis, design, and modelling with no or minimal mistakes | Comprehensive analysis, design, and modelling with some mistakes | Adequate analysis, design, and modelling with some major mistakes | Adequate analysis, design, and modelling with many major mistakes | Inadequate analysis, design, and modelling with substantial major mistakes | 6 |  |
| **5** | Detailed and comprehensive analysis, design, and modelling with no or minimal mistakes | Comprehensive analysis, design, and modelling with some mistakes | Adequate analysis, design, and modelling with some major mistakes | Adequate analysis, design, and modelling with many major mistakes | Inadequate analysis, design, and modelling with substantial major mistakes | 6 |  |
| **6** | Detailed and comprehensive analysis, design, and modelling with no or minimal mistakes | Comprehensive analysis, design, and modelling with some mistakes | Adequate analysis, design, and modelling with some major mistakes | Adequate analysis, design, and modelling with many major mistakes | Inadequate analysis, design, and modelling with substantial major mistakes | 10 |  |
| **7** | Sufficiently detailed and relevant description and analysis of the integration of the organization’s software architecture with either an IoT or a cloud system | A large extent of detailed and relevant description and analysis of the integration of the organization’s software architecture with either an IoT or a cloud system | Description and analysis of the integration of the organization’s software architecture with either an IoT or a cloud system, with many details missing. | Description and analysis of the integration of the organization’s software architecture with either an IoT or a cloud system, with complete irrelevancy | No description and analysis of the integration of the organization’s software architecture with either an IoT or a cloud system | 8 |  |
| **8** | The report complies completely with the format and the language used is clear | Most parts of the report comply with the format and the language used is clear | Most parts of the report comply with the format and the language used is considerably clear | Many parts of the report do not comply with the format and the language used is considerably clear | Most parts of the report do not comply with the format and the language used is not clear | 3 |  |
| **9** | All drawings are clearly documented | Most of the drawings are clearly documented | Some of the drawings are clearly documented | Most of the drawings are not documented. Documented drawings are mostly unclear | Minimal or no effort in documenting the drawings | 7 |  |
| **TOTAL** | | | | | | **55** |  |