**Quản lý khách sạn 5-STAR**

**Project Glossary (PG)**

**Version 1.0**

**Produced by:**

**Nhóm II**

**Lớp Xây dựng phần mềm TH2008**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11/04/2011 | 1.0 | Bản thảo lần 1 | Nhóm 15 |
|  |  |  |  |
|  |  |  |  |

Mục lục

1 Giới thiệu 4

1.1 Mục tiêu của tài liệu 4

1.2 Người đọc dự kiến 4

1.3 Tài liệu liên quan 4

1.4 Tổng quan về tài liệu 4

2 Những từ viết tắt 5

3 Các thuật ngữ chuyên môn 7

# Giới thiệu

Sau đây là những thông tin giới thiệu chung về tài liệu này.

## Mục tiêu của tài liệu

Tài liệu này hướng đến những mục tiêu sau:

* Định nghĩa các từ viết tắt trong đồ án Quản lý khách sạn 5-STAR.
* Định nghĩa các thuật ngữ được sủ dụng trong đồ án Quản lý khách sạn 5-STAR.
* Qua đó có thể cải thiện sự giao tiếp giữa các stakeholder và đội ngũ phát triển dự án.

## Người đọc dự kiến

Tài liệu này hướng đến những người đọc dự kiến sau:

* Stakeholders:
* Thầy phụ trách bộ môn.
* Đội ngũ phát triển:
* Đội ngũ quản lý.
* Đội ngũ yêu cầu.
* Đội ngũ thiết kế.
* Đội ngũ phát triển.
* Đội ngũ kiểm thử độc lập.

## Tài liệu liên quan

Tài liệu này định nghĩa các từ viết tắt và thuật ngũ được sử dụng trong những tài liệu sau:

* <Document Name>

## Tổng quan về tài liệu

Tài liệu được chia làm các phần như sau:

* *Giới thiệu*: giới thiệu chung về tài liệu này cho người đọc.
* *Những từ viết tắt*: định nghĩa các từ viết tắt trong đồ án Quản lý khách sạn 5-STAR.
* *Domain Terms*, which formally defines all domain-specific terms used on the <Project Name>.
* *Các thuật ngữ chuyên môn*: định nghĩa các thuật ngữ trong đồ án Quản lý khách sạn 5-STAR.

# Những từ viết tắt

Những chữ viết tắt sau được sử dụng trong dự án:

API Application Programmer Interface

ASP Active Server Pages

AVS Application Vision Statement

CGI Common Gateway Interface

CMP Configuration Management Plan

COM Component Object Model

CORBA Common Object Request Broker Architecture

CSS TBD

DBDD Database Design Document

DCOM Distributed Component Object Model

DHTML TBD

DP Deployment Plan

DNS Domain Name Service

EAC Electronic Auction Corporation

EAPIS External Application Programmer Interface Specification

EAS Electronic Auction System

FTP File Transfer Protocol

GIF

GUI Graphical User Interface

HI Human Interface

HIDD Human Interface Design Document

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IIOP Internet Inter-ORB Protocol

IP Internet Protocol

ISP Internet Service Provider

JPEG

JSP TBD

MIME Multimedia Internet Mail Extensions

MP Metrics Plan

MPEG

MTBF Mean Time Between Failures

MTS TBD

MTTF Mean Time To Fix

OBI Open Buying on the Internet

OLE Object Linking and Embedding

ORB Object Request Broker

PDD Package Design Document

PDF Portable Document Format

PG Project Glossary

RMI Remote Method Invocation

RMP Risk Management Plan

RP Release Plan

SET Secure Electronic Transactions protocol

S-HTTP Secure HTTP

SQL Structured Query Language

SSL Secure Sockets Layer

SWAD Software Architecture Document

SWRS Software Requirements Specification

SYSAD System Architecture Document

SYSDP System Development Plan

SYSRS System Requirements Specification

TBD To Be Determined

TCP Transmission Control Protocol

UI User Interface

UM Users Manual

URL Uniform Resource Locator

VRML Virtual Reality Modeling Language

WWW World Wide Web

XML Extensible Markup Language

# 

# Các thuật ngữ chuyên môn

This section defines common e-market terms and system development terms.

A

**abstraction** – a model of something that captures all of its essential characteristics while ignoring all of its diversionary details

**access control** – a security technique dealing with how users access is restricted to application functionality and data.

**ActiveX** – TBD

**activity** – a major unit of work consisting of a cohesive collection of workflows that produce a related set of work products.

**applet** – a Java program that is downloadable to a client over the Internet.

**application** – a major, fully-functional, stand-alone work product that is developed by a development organization for use by a user organization.

**application server** – a server computer that executes business logic and the business model classes.

**application vision statement** – the document in the customer set that communicates the customer’s vision of the application to the development organization. It documents the stakeholders of the application and communicates the business, operational, and quality goals, constraints, and business rules to their stakeholders.

**architecture** – the strategic design that captures the most important, pervasive design decisions and their rationales. An architecture must fulfill (and is therefore validated against) the architecturally significant operational and quality requirements. The architecture drives and constrains the tactical (i.e., detailed) design. See also *system architecture* and *software architecture*.

**architecture document** – a document formally capturing the architecture of an application. See also *software architecture document* and *system architecture document*.

**architecture set** – the set of work products relating to architecture.

**authentication** – a security technique used to determine if an actor actually is who he seems to be. Contrast with *authorization.*

**authorization** – a security technique used to determine if an actor has the required privileges to use specific system capabilities. Contrast with *authentication.*

B

**bricks-and-mortar** – a traditional physical business without an Internet presence.

**browser** – a software tool enabling the display and navigation of web pages on the World Wide Web.

**build** – a relatively small stage that forms an identified part of a phase.

**business goal** – a goal of the application that is primarily of management interest, such as cost, time to market, etc.

**business model** – a model of the business in terms of the terminology of the domain experts and users without regard to any specific system or software application. The main objectives of a business model are to formally represent the business in terms of the essential business objects and how they collaborate to perform the major business processes. The business model forms the basis for understanding and engineering (or reengineering) the business and its processes. The business model typically consists of a business-level use case model and associated object model.

C

**categorization** – the analysis of use case paths in terms of their criticality, frequency, probability of containing defects, and risk.

**catalog** – a database of goods and services for sale online.

**catalog management** – the part of a software application that manages the catalog.

**clicks-and-mortar** – a traditional physical business with an online Internet presence.

**client computer** – a relatively small personal computer or workstation in a multi-tier architecture that is directly used by an actor. Client computers typically offload much of the processing and persistence to larger server computers. Contrast with *server computer*.

**class** – the [partial] definition of a kind of object. A class is the implementation of a type (a single abstraction) and may implement one or more interfaces.

**class library** – a cohesive collection of classes related by inheritance and delegation.

**component** – an architecturally significant part of an application with well-defined inbound and outbound interfaces that encapsulate a cohesive set of responsibilities. Components are reusable, and are intended to be adaptable, but not modified by their users. Components are typically intended to be used in conjunction with other compatible components from a well-architected component framework.

**component architecture** – the essential hardware, software, personnel, and documentation components of a system, their responsibilities, their relationships, and their interfaces. Contrast with *conceptual architecture* and *concurrency architecture.*

**component framework** – a cohesive collection of collaborating components that have been architected to work together. The component framework defines the components, how they are related, the choice of technology (e.g., CORBA, DCOM, and RMI) used to connect them, and their specific interfaces.

**conceptual architecture** – the essential concepts (abstractions) of a system, their responsibilities, their relationships, and their interfaces. Contrast with *concurrency architecture* and *component architecture.*

**concurrency architecture** – the processes of an application, their responsibilities, their relationships, and how distribution units are allocated to them. Contrast with *conceptual architecture* and *component architecture.*

**configurability** – the degree to which an application can be configured into multiple variants (e.g., via internationalization).

**configuration auditing** – the auditing of a baseline to verify if it is complete, correctly identified, and properly versioned.

**configuration control** – the management of changes to baselined configuration items including requesting, evaluating the impact of, approving/rejecting, and implementing changes.

**configuration identification** – the identification and designation of components as configuration items and the identification of baselines.

**configuration item** – a work product that is baselined and placed under configuration control.

**configuration management** – the management of configuration items and baselines. Configuration management consists of the following tasks:

* Configuration identification.
* Configuration control.
* Configuration status accounting.
* Configuration auditing.

**configuration management plan** – the planning work product that documents the plans for performing configuration management.

**configuration management team** – a team that is responsible for configuration management.

**configuration status accounting** – the recording and reporting of information needed to effectively manage a configuration including the:

* Status of proposed changes, deviations, and waivers to the configuration.
* Implementation status of approved changes.

**constraint** – a mandatory design decision that is treated as a requirement in that it restricts the remainder of the architecture, design, and implementation.

**construction phase** – the phase during which most of the development of the application occurs. Contrast with *strategy phase*, *inception phase*, and *transition phase*.

**context diagram** – a kind of semantic net documenting the blackbox application, the relevant externals that interact (either directly or indirectly) with it, and the relevant relationships between them.

**control** – TBD

**convention** – a process document that constrains the delivery process. See also *standard*, *template*, *procedure*, *guideline*, and *inspection checklist*.

**customer** – the role played by a person who pays a development organization to develop an application.

**customer analysis** – the activity of the strategy phase during which the customer is analyzed.

**customer organization** – an organization that pays a development organization to develop an application.

**cookie** – a block of information transmitted from a server to a browser for storage. The browser sends the cookie back to the server on subsequent web requests to identify the session or provide application state.

**criticality** – an evaluation of how important a capability (e.g., use case path) is to the customer or user.

D

**database** – a software application that provides persistence.

**database server** – a server computer used to handle persistence and the associated communication with disk or tape libraries.

**decryption** – a security technique that uses a key to unscramble encrypted data so that it is readable. Contrast with *encryption*.

**delivery cycle** – the collection of phases associated with the sale, development, maintenance, and post-transition follow-up of multiple applications for a single customer. Contrast with *development cycle* and *life cycle*.

**design constraint** – a design decision treated as a requirement which therefore constrains the designer.

**development cycle** – the collection of phases during which a single application is developed and transitioned to a customer. Contrast with *delivery cycle* and *life cycle*.

**digital certificate** – an authentication technique consisting of a digital object in which a trusted third party attests to the binding of a user name to a public key.

**digital signature** – an unforgeable electronic signature consisting of a block of information that is attached to a message and that could only have been created by the sender of the message. A digital signature is typically encrypted with the message sender’s private key and is validated by the receiver when decrypted using the sender’s public key. The digital signature also often contains a digital timestamp to ensure currency.

**disk library** – an automated library used to store massive amounts of information on disks. A disk library typically contains disk drives for reading and writing information, access ports for entering and removing disks, and robots for moving disks between storage cells, drives, and access ports. Contrast with *tape library*.

**domain expert** – the role played by someone who provides authoritative information about a given domain.

**domain model document** – the document that formally captures the essential domain concepts, their responsibilities, their relationships, and how they collaborate to fulfill architecturally-significant requirements.

E

**e-business** – business that is primarily carried out via electronic means (e.g., over various networks).

**e-commerce** – the selling and buying of goods and services electronically (e.g., over the Internet using the World Wide Web).

**e-mall** – a website consisting of multiple e-stores.

**e-marketplace** – a market place consisting of multiple e-businesses connected electronically.

**e-store** – a virtual store at which goods and services are bought and sold over the Internet.

**e-strategy** – a customer’s strategy for doing business within its e-marketplace.

**e**-**tailer** – a retail store that operates totally online over the Internet.

**efficiency** – the degree to which the system effectively uses its memory and computing resources.

**encryption** – a security technique that uses a key to scramble data so that it is not readable until decrypted with an associated key. Contrast with *decryption*.

**endeavor** – an indirect producer representing a high-level venture undertaken to develop and maintain one or more related applications. See also *programme*, *project*, and *enterprise*.

**enterprise** – an endeavor consisting of all programmes within a business.

**exceptional path** – any use case path through a use case that captures an exceptional (error) situation. Contrast with *normal path*.

**extensibility** – the degree to which the system can be modified to meet changing requirements or goals.

**external** – any relevant thing that is outside of the application and that interfaces (either directly or indirectly) with the application.

F

**firewall** – a security mechanism consisting of a computer that limits access by a public network (e.g., the Internet) to a private system or network.

**frequency** – how often a use case path typically executes.

**fulfillment** – TBD

G

**goal** – a perceived need (i.e., informal requirement) at the customer/user level that drives the analysis and specification of engineering-level requirements (i.e., operational, quality, and design constraints).

**guideline** – a document the provides guidance on how to perform one or more related work units.

H

**home page** – the initial main webpage of a website.

I

**identification** – a form of security whereby the application must identify users before allowing them to use the application. Contrast with *authentication* and *authorization*.

**inception phase** – the relatively short-duration (6 weeks) phase during which the initial partial versions of the requirements and architecture of the application are completed so that the following construction phase can be estimated and planned.

**incremental** – a property of a development process whereby units of work are repeated to produce additional new work products or capabilities of work products. Development cycles are typically incremental because applications are too large and complex to be built all at once in a big bang fashion. Contrast with *iterative*.

**inspection** – a relatively formal verification and validation technique, the purpose of which is to identify defects in a work product, which may be documentation, software, or hardware.

**inspection checklist** – a work product specific checklist of questions used during an inspection to ensure that the inspector has considered all important potential defects.

**instantiation guideline** – a process framework guideline for selecting process components from an existing class library of such components to meet the needs of a specific project.

**integrity** – TBD

**iterative** – a property of a development process whereby units of work are repeated on existing work products to improve them (e.g., to fix defects and adapt to changes in requirements). Contrast with *incremental*.

**interface** – TBD

**internal build** – a build that does not result in a version of the application being delivered by the development organization to the customer organization. Internal builds are used internally by the development organization to help manage iterative and incremental development. Contrast with *release*.

**Internet** – the world’s largest public network of networks on which the World Wide Web is based.

**interaction** – a part of a use case path consisting of a single communication between an external and an application.

**interoperability** – the ease with which the system can be integrated with other system (e.g., browsers, legacy applications, and required databases).

**intranet** – an organizational-internal network based on the same technology as the Internet.

**intrusion detection** – a security activity involved with detecting the attempted access of an application by an unauthorized user.

J

**JPEG** – a file format for compressed bitmaps.

L

**language** – the medium for implementing a work product. See also *implementation language*, *modeling language*, and *natural language*.

**layer** – a major horizontal collection of software components in a layered software architecture. Each layer typically depends only with the layer immediately below it. For example, a software architecture may consist (top to bottom) of a presentation layer, a transaction layer, a domain layer, and a persistence layer. Contrast with *tier*.

**life cycle** – the collection of phases associated with the development and maintenance of a single application. Contrast with *development cycle* and *delivery cycle*.

**load balancer** – a hardware node responsible for allocating requests among multiple servers in order to equalize the loads handled by each server.

**local area network (LAN)** – an organizational internal network. Contrast with *wide area network*.

M

**market analysis** – the activity of the strategy phase during which the customer’s competitive marketplace is analyzed.

**method** – (1) a synonym for process and methodology. (2) an operation performed by an object.

**milestone** – a scheduled point in time during a development when some set of significant objectives is to be achieved.

**model** – an abstraction of something that captures its essential characteristics (for some purpose) while ignoring unimportant or diversionary details.

**modeling language** – a language for specifying or documenting models. For example, the Unified Modeling Language (UML) and the OPEN Modeling Language (OML). Contrast with *implementation language* and *natural language*.

N

**network** – the software and hardware that connects multiple computers.

**normal path** – a path through a use case that represents a normal sequence of interactions. Contrast with *exceptional path*.

O

**object** – a software component that models something in an application. An instance of a class that conforms to a type.

**operational availability** – the percent of time that the system must function correctly.

**operational goal** – any goal that captures a function to be performed by the application. Contrast with *quality goal*.

**operational requirement** – any requirement that specifies a function to be performed by an application. Contrast with *quality requirement*.

P

**package** – (1) a cohesive collection of software classes, interfaces, and other packages (i.e., a midsize software component). (2) a cohesive collection of work products.

**package architecture** – the architecture of a software application or component in terms of its packages, their responsibilities, their relationships, and their interfaces. Contrast with *type architecture* and *process architecture.*

**parallel** – the characteristic of a stage whereby multiple producers (roles, teams) are concurrently performing tasks (e.g., different producers are simultaneously working on different work products).

**path** – See *use case path*.

**path requirement** – See *use case path requirement*.

**payment** – TBD

**performance** – the speed with which the application executes its functions.

**phase** – a stage consisting of one or more related builds.

**plug-in** – a software module that is manually installed into the Netscape Navigator browser by the user to provide new browser capabilities.

**portability** – the ease with which the system can be moved from one environment (e.g., hardware, operating system) to another.

**postcondition** – an assertion that must hold following the successful execution of the associated function (e.g., use case path, class method).

**precondition** – an assertion that must hold prior to the successful execution of the associated function (e.g., use case path, class method).

**procedure** – a document that specifies the work products, producers, and tasks of the workflows for a specific activity or phase.

**process** – a project-specific cohesive collection of process components (i.e., work products, their producers, and the work units performed by the producers as they collaborate to produce the work products). Also known as a *method* or *methodology*.

**producer** – TBD

**project glossary** – the official glossary of all abbreviations, domain terms, and technical terms that are used on the project.

Q

**quality goal** – a non-operational goal of the application.

**quality requirement** – any non-operational requirement. Sometimes referred to as a quality factor or “ility”.

R

**release** – a build that results in a version of the application being delivered by the development organization to the customer organization. Contrast with *internal build*.

**reliability –** the average time that the application functions correctly without failure, typically measured in terms of mean time between failures.

**requirement** – any mandatory, externally-observable, and testable characteristic or behavior of an application.

**requirements analysis** – The subactivity of requirements engineering during which requirements are modeled in order to ensure that they are complete, consistent, understandable, etc.

**requirements elicitation** – the subactivity of requirements engineering during which requirements are identified by eliciting them from customers, domain experts, marketing, and users.

**requirements engineering** – the activity involved with all aspects of requirements

**requirements management** – the subactivity of requirements engineering during which requirements are baselined, placed under configuration control, and maintained during subsequent iteration.

**requirements reuse** – the subactivity of requirements engineering during which reusable requirements are identified in the reuse repository and reused (possibly with modification) on a specific project.

**requirements specification** – (1) the subactivity of requirements engineering during which requirements are specified in a requirements document. (2) a document that formally specifies the requirements of all or part of an application.

**requirements trace** – the mapping of requirements between work products at two different levels of abstraction, in order to ensure that all requirements have been handled. For example, the tracing of goals in the Application Visions Statement to requirements in the System Requirements Specification.

**reusability** – the degree to which the system can be used for purposes other than originally intended (e.g., as part of other applications)

**risk** – anything significant potential cause for project failure.

**risk management** – a delivery management activity during which project risks are identified, monitored, and risk avoidance and mitigation activities are identified and documented.

**risk management plan** – the plan that documents a projects risk management approach.

**robustness** – the degree to which the system continues to properly function under abnormal circumstances.

**role** – a cohesive part that is played by a person during the production of one or more versions of work products.

**router** – a server computer that routes client network communication to web servers, thereby performing load balancing and ensuring reliability.

S

**safety** – the degree to which the system does not directly or indirectly (e.g., via inactivity) cause accidental harm to life or property (e.g., loss of money or data)

**scalability** – the degree to which the system can scale (e.g., can handle more simultaneous users or clients, can store more information in its databases)

**script** – TBD

**security** – the degree to which an application protects itself from unauthorized access or modification. Security involves enforcing access control, ensuring communication integrity, encryption and decryption, identification, and authentication.

**server computer** – a relatively powerful computer in a multi-tier architecture that performs the majority of the processing and persistence of data. Contrast with *client computer*.

**sequence diagram** – TBD

**shopping cart** – an application that temporarily stores customer items to be purchased.

**software architecture** – the architecture of a software application in terms of its type architecture, package architecture, and concurrency architecture. Contrast with *system architecture.*

**software architecture document**

**stage** – a formally identified period or milestone during the delivery process. See also **cycle**, *phase*, *build*, and *milestone*.

**stakeholder** – the role played by any person who has a material interest in the application, its requirements, architecture, design, and implementation.

**standard** – a document that specifies the required content and format for a work product.

**storefront** – a website designed to provide the capabilities of an on-line store.

**strategy document** – the document that formally captures the customer’s e-strategy. This includes (but is not limited to) customer analysis, user analysis, market analysis, business case, and recommended applications.

**strategy phase** – the phase during which the overall e-business strategy of the customer is developed and initial projects are identified.

**system** – an application consisting of hardware, software, wetware, and paperware.

**system architecture** – the architecture of a system in terms of its conceptual architecture and component architecture. Contrast with *software architecture.*

**system architecture document** – the document that formally captures the system architecture in terms of the main drivers of the architecture, the conceptual architecture, the component architecture, and how these architectures implement the architecturally significant requirements.

**system requirements specification** – a document that formally specifies the operational and quality requirements of a system application.

T

**tailoring guideline** – a guideline for tailoring a process component to meet the needs of a specific project.

**tape library** – an automated library used to store massive amounts of information on magnetic tape cartridges. A tape library typically contains tape drives for reading and writing information, access ports for entering and removing tapes, and robots for moving tapes between storage cells, drives, and access ports. Contrast with *disk library*.

**task** – a functionally cohesive operation (reified as a work unit) that is performed by a direct producer. A single responsibility of a producer will be fulfilled by the execution of one or more tasks. A task results in the creation, modification, or evaluation of a version of one or more work products.

**team** – a cohesive group of roles (or aggregation of component teams) that collaborate to perform the tasks comprising the workflows that produce a cohesive collection of work products.

**technique** – a reified implementation (i.e., way of performing) of a task using the Strategy Pattern. In order to provide flexibility during instantiation and tailoring, direct producers may delegate the performance of their own tasks to techniques.

**tier** – a major horizontal collection of hardware components in a layered hardware architecture. Each tier typically communicates only with the tiers immediately above and below it. For example, a 4-tier hardware architecture may consist (top to bottom) of a client tier, a web-server tier, an application server tier, and a database server tier. Contrast with *layer*.

**time box** – to limit the duration of a stage by scheduling its starting and ending dates (e.g., by scheduling the milestone that marks its end).

**tool** – a software application that automatically produces or modifies a work product. For example, a word processor, a CASE tool, a compiler, etc.

**transition phase** – the phase during which the application is delivered to the customer and placed into operation.

U

**unit of work** – a functionally-cohesive operation that is performed by a producer during development and that is reified as an object to provide flexibility during instantiation and tailoring of a development process.

**usability** – the degree of effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of their tasks when installing and using an application.

**usage guideline** – a guideline for using the OPEN Process Framework.

**usage scenario** – an extremely specific way of using a blackbox application, defined as a single instance of a use case path. Usage scenarios are test cases for use case testing.

**use case** – a general way of using a blackbox application, defined in terms of use case paths. Example use cases for an automatic teller machine application could include deposit funds, withdraw funds, obtain account balances, make transfers, etc.

**use case diagram** – a diagram that documents which use cases are used by which externals and any relationships between the use cases.

**use case path** – a more specific way of using a blackbox application, defined in terms of preconditions, interactions between externals and the application, and postconditions. Example use case paths for the withdraw funds use case of an automatic teller machine application could include fast cash, successfully withdraw a specific amount, account overdrawn, over daily cash limit, etc. Use case paths form equivalence sets of usage scenarios.

**use case path requirement** – A textual requirement that summarizes a single path through a use case.

**use case requirement** – A textual requirement that summarizes a single use case.

**user analysis** – the strategy activity during which the users of an application are profiled and analyzed.

W

**wallet** – an application used to store user payment credentials (e.g., credit card information) and receipts.

**web page** – an HTML page that can be downloaded and displayed via the world wide web.

**web server** – a server computer that forwards client communication over the web to application servers.

**web site** – a cohesive collection of web pages belonging to a single customer and accessed through its home page.

**wide area network (WAN)** – TBD

**work flow** – A contiguous sequence of task performances whereby producers collaborate to produce a version of a work product.

**work product** – anything of value (e.g., document, diagram, application, class) produced by a direct producer during the performance of one or more tasks. During the evolution of a work product, work product versions are created. Therefore, each work product has an associated time-ordered set of work product versions.

**World Wide Web** – an Internet software application that allows users to communicate via web pages.