Car Parking System Analysis and Design Case Report

Friday, 15th April 2022

By

Codie Shannon

30013375

Assignment One

In

COMP.6209

System Analysis and Design

Codie Shannon
30013375

1. Current Manual Car Parking System

In this section the researcher assumes a brief scenario of a manual car parking system and explains the disadvantages.

A. Brief Scenario

A fifty-five-year-old male attempts to enter the parking complex where he is stopped in his tracks as he recalls he has forgotten his credit card. The man glances down at his mobile phone and comes to the realization that he must drive home to obtain his card. Once he has returned to the parking complex after obtaining his card, there is a massive waiting queue for clientele entering and exiting the complex.

After a long twenty-minute waiting period he is finally able to get his ticket from the pay station, before awaiting the entry arm to raise and driving through to the parking complex. Following this, the man slowly makes his way up the vehicle entry queue to the sixth floor where he locates a parking space and parks his car (SiegerParking, 2017).

The man then exits his vehicle with his credit card and parking ticket. Succeeding this, he goes to the store to buy a pie and drink before placing his credit card and parking ticket in his pocket and heading to work.

On his journey back to his vehicle he sees a dodgy character with a spray can. However, he ignores this and carries on walking to his vehicle (Rentcubo, 2021). Once he returns to his vehicle, he comes to the awareness that he has lost his parking ticket somewhere along the journey. Following this, the man contacts the guard where he is told he must pay a fifteen-dollar lost ticket fee before he can leave with his vehicle. After the man pays the lost ticket fee, he drives to the exit arm where he is let through by the guard (Foote, 2022).

B. Disadvantages

The problems identified within the current manual car parking system are an absence of online payment options, difficulty in locating parking spaces, inadequate management of users, and a defective paper-based record system.

The existing manual car parking system currently has an absence of online payment alternatives. The missed opportunity of online payment alternatives could cause the company thousands of dollars in potential earnings each year as available payment options like cash or cards are easily forgotten by potential clientele.

In the present system, time is lost for drivers searching for a parking space, this circumstance may seem trivial. However, in effect, as the parking complex is in the heart of the city this could cause congestion within the parking premises causing long waiting queues for drivers entering and exiting the complex. This situation could cause the company loss of income as drivers searching for a quick parking space may opt for another parking complex or a street park instead of using the company's parking premises (SiegerParking, 2017).

In the existing system, there is a lack of user verification which could lead someone who has the intent of causing vandalization, users harm or vehicle theft to access the parking complex (Rentcubo, 2021). This lack of user verification could cause loss of profits in maintenance costs or loss of potential clientele as users will be less inclined to use the parking premises if it receives a bad reputation for this type of misfortune.

The current paper-based record system is unreliable as users could easily damage or lose their ticket. This could be a hassle for clientele as they will need to contact the guard and pay a fifteen-dollar fee to get the situation resolved (Foote, 2022).

The current system is also insufficient for the purpose of data management as if a parking lot manager is required to search through the paper-based records to locate a piece of information, this operation may take hours (Rentcubo, 2021).

2. Automated Car Parking Systems

In this section, the researcher identifies an automated car parking solution and explains the functionality and benefits of such a system.

A. Functionality

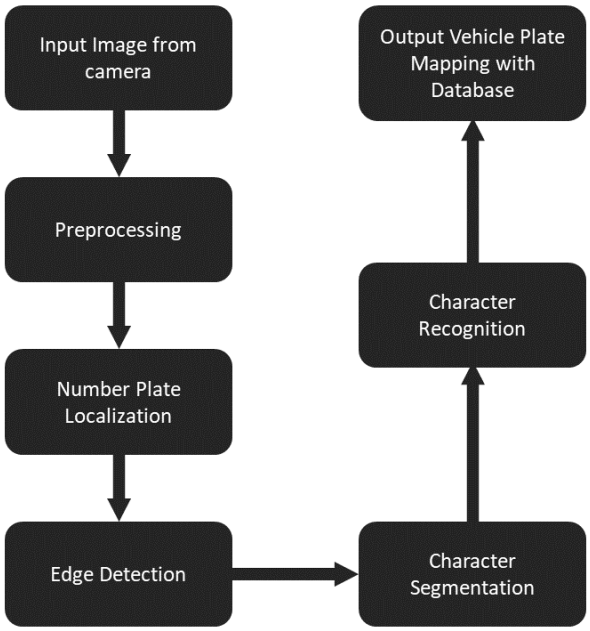
The parking industry is changing and it's time we did too. Gone are the days of user's using physical paper-based tickets to access parking complexes. The next generation of parking systems is here with ticketless, artificial intelligence driven technology.

The focus of this section will be the fixed in-lane, gated License Plate Recognition (LPR) system. The gated License Plate Recognition system controls access to the parking complex by utilizing single lane cameras to detect approaching vehicles (Sharma, 2021).

Once a vehicle's license plate is detected within an entrance lane by a single lane camera, the LPR system will take a picture of the license plate. Once captured, the LPR system will employ optical character recognition (OCR) software to convert the image to a text-based format readable by a computer system, as seen in figure 1 (Qadri & Asif, 2009).

Figure 1

Optical Character Recognition Process



(Vashishtha & Sharma, 2018)

Once converted, the LPR system will save the license plate's text-based format to a database alongside the customer's entrance time, license plate image, rate, and "in some cases, the make, model and colour of the car" (Sharma, 2021).

Once the vehicle has been recorded, the LPR system will utilize the license plate's text-based format, comparing it to a blacklist to determine whether to let the vehicle access the parking complex (Sharma, 2021). If the vehicle passes this check, the barrier arm will raise, and the vehicle will be allowed to drive through to the parking complex.

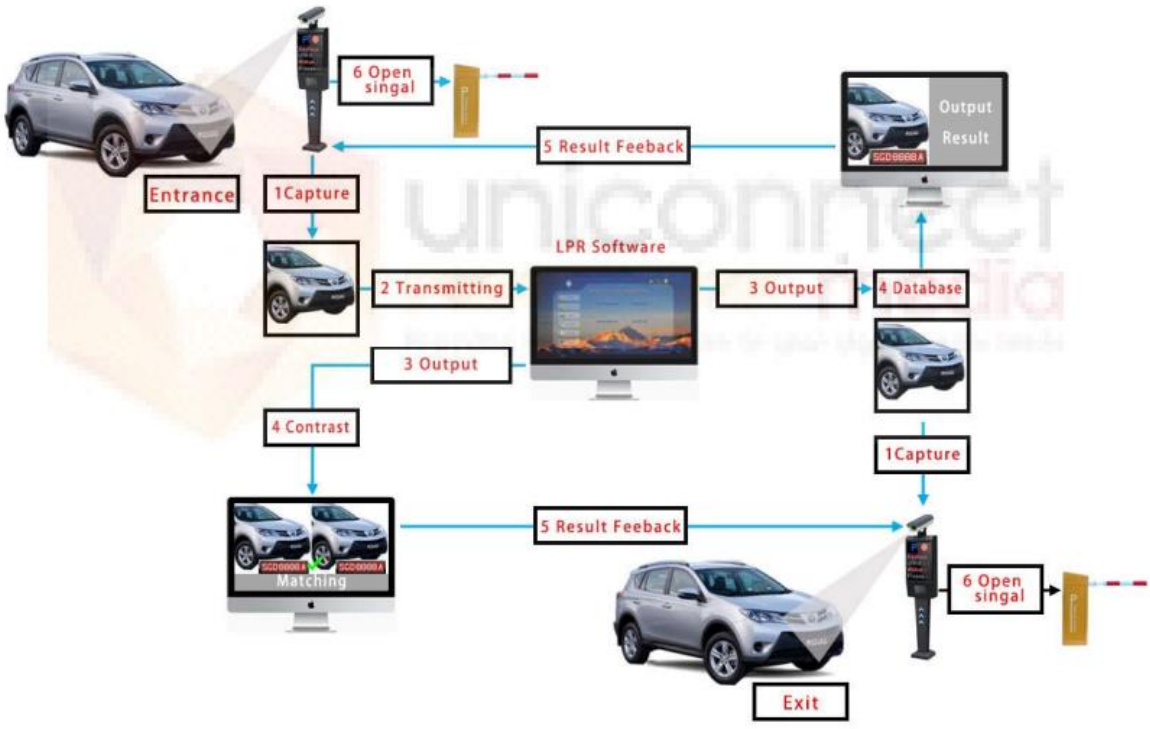
Once a customer is finished in the parking complex, the parking transaction commences. The customers have a wide range of payment alternatives such as a physical pay station, onsite by mobile app, mobile scan-to-pay, or pay in advance by mobile app or online. The LPR system can perform these payments by searching the vehicle database for reservation type, entrance time, license plate, and in some cases the make and model of the vehicle to calculate the fee due (Sharma, 2021).

When exiting the parking complex, the vehicle's license plate will be detected within an exit lane by a single lane camera. Once detected, the LPR system will take a picture of the license plate and convert it to a text-based format readable by a computer system (Qadri & Asif, 2009).

Once converted, the LPR system will search the vehicle database using the license plate's text-based format to verify payment. If a customer has not paid, they will be prompted to do so at the exit gate's pay station. Once the customer's payment is verified the exit barrier arm will raise and the customer will be released from the parking complex (see figure 2) (Sharma, 2021).

Figure 2

LPR System Process



(Uniconnect Systems, 2021)

B. Benefits

The LPR system provides a range of benefits for management and consumers alike according to Bechthold (2020) such as increased security, reduced traffic, decreased wait time, elimination of tickets and contactless entry.

The LPR system delivers increased security across the board. One of the substantial problems that appear for parking management in manual car parks are individuals with unauthorized vehicles. With LPR, this is no longer an issue as LPR technology allows companies to only allow authorized vehicles to enter the parking complex (Bechthold, 2020).

LPR technology acts as a vital crime deterrent as the parking premises contains multiple cameras placed around the perimeter. Alongside this, every vehicle entry and exit are recorded and saved to a database. As you can see, the LPR system behaves as a security system which scares potential criminals off (Bechthold, 2020).

The security of a LPR system does not just provide benefits for the parking management and parking complex, it also provides benefits for the customers. The added sense of security unequivocally provides customers and parking management with an unrivalled peace of mind (Bechthold, 2020).

LPR systems have increased the efficiency of parking complexes significantly as customers are no longer required to interact with a pay station or parking manager. LPR solutions permit authorized vehicles to enter and exit the parking premises instantly upon validating the license plate. This terminates long waiting queues at vehicle entrances and exits. In addition, customers no longer need to interact with pay stations or parking management out of necessity. Instead, customers are now able to pay onsite via mobile app, mobile scan-to-pay, or they can pay in advance via mobile app or the company's online website (Bechthold, 2020).

As LPR permits seamless entry and exit for authorized vehicles, less time is being consumed at the gate, cutting down vehicle waiting queues. This in-turn increases potential profits by allowing the entry lanes to process a substantially increased number of customers (Bechthold, 2020).

The continuous threat of COVID-19 causes potential clientele to be equitably concerned for their health and safety. Potential clienteles are searching for minimal contact with public surfaces alongside little contact with other human beings. Luckily, a substantial benefit smart parking provides for the health and safety of customers and parking management alike is a one hundred percent contactless system. The LPR system allows users and parking management to no longer worry about potential exposure to the virus alongside any other pandemic wave caused by a new virus in the future. Clientele can come and go from the parking complex without touching anything more than their vehicle (Bechthold, 2020).

The LPR system replaces physical paper-based records by admitting vehicles into the parking complex based on a vehicle's license plate. Not only does this save time for customers and the parking management team alike but it also saves money for the company. The expenses that add up for tickets is costly and one of the most instantly recognizable benefits of an LPR based system. Alongside this, LPR eradicates the annoyance of ticket swapping. Ticket swapping is difficult to manage and can be immensely tiresome and pricey for parking management (Bechthold, 2020).

Furthermore, customers can easily fool a manual car parking system into paying a cheaper rate than the total number of hours they have been in the parking complex by pretending to lose a ticket (Bechthold, 2020).

Thanks to the LPR system, customers and parking management have no more worries about ticket swapping, theft, or loss as these are no longer a possibility (Bechthold, 2020).

3. Object-Oriented vs Traditional Software Development

A. Traditional Software Development

The traditional software development model also known as the linear sequential model, or the waterfall model was first mentioned in an article written by Winston W. Royce in 1970. However, a name wouldn't be coined for the model until a paper written by T.E. Bell and T.A. Thayer was published in 1976 where it may have first been identified as "Waterfall" (Asmo, 2019).

The model follows a sequential, linear design approach where progress flows in a descending and singular direction only transitioning to the next phase once the current phase is completed, reviewed, and approved (Asmo, 2019).

Traditional software development methodologies usually consist of the following phases, requirements (determining capabilities of system), analysis (investigation of relevant entities), design (determining design and deployment of system), implementation (assembly of system), testing (comparison of finished product to requirements), and deployment (delivery of product to end-users) (Bala, 2018).

Traditional software development is best suited for simple projects which do not require any form of previous-phase revision to correct or alter developments (Asmo, 2019).

The rigid approach leaves very little flexibility in terms of client feedback. During the development process, valuable feedback from the client is overwhelmingly overlooked as the development stages leave little to no room for client input, only being collected during the initial phases of the development process (Martin, 2022).

The rigid approach leaves very little flexibility in terms of repairability for developers. During the development process, errors that appear may only be fixed during the phase in which they occur. Alongside this, the testing period approaches quite close to the end of the development cycle, leaving very little time to fix any issues that appear within the product, alongside the possibility of becoming quite troublesome and prolonging the development timespan (Martin, 2022).

B. Object-Oriented Development

Object-oriented software development is a model that views a system as a set of objects interacting with one another. The approach places emphasis on the exploitation of objects instead of the logic or actions needed to exploit them (Munassar & Govardhan, 2011). Examples of objects range from vehicles described by their colour, make, model, and gas type to even graphical user interface elements within an operating system like buttons, icons, and text.

Generally speaking, all object-oriented methodologies are similar in terms of phases and artifacts. However, the methodologies do differ in a few key aspects (Munassar & Govardhan, 2011).

Object-oriented methodologies usually consist of three phases analysis (establishment or ground work of the system), construction (assembly of the system),

and testing (comparison of the system to the system specifications defined during the analysis phase) (Munassar & Govardhan, 2011).

Object-oriented methodologies attempt to eliminate the flaws of their predecessors' traditional methodologies, by offering flexibility in the tools, and techniques utilized during the development phases. Alongside these key improvements, a few object-oriented methodologies offer alternative development paths intended for variations of different development types and sizes. An example of the flexibility given to developers in terms of tools are the options in which particular types of diagrams they would like to utilize (Munassar & Govardhan, 2011).

Since object-oriented software development provides such flexibility, the development team must select an appropriate methodology, alongside agreeing which artifacts are to be produced before they do any detailed planning, analysis, or scheduling (Munassar & Govardhan, 2011).

According to Munassar and Govardhan (2011), each methodology within the object-oriented software development family addresses the following aspects, the philosophy behind each individual phase, workflows, specific activities and artifact creation within each stage, alongside determining the dependencies and notations utilized between different kinds of artifacts. The methodologies will also commonly have the development team create static and dynamic behaviour models within each stage.

Static modelling is the process in which decisions are made about what the physical or logical parts of the system are and how they should be linked together. Dynamic modelling on the other hand, is the process in which decisions are made about how the system should behave (Munassar & Govardhan, 2011).

Object-oriented methodologies contain the following diagrams, use case diagrams, class diagrams, communication diagrams, deployment diagrams, and the sequence diagrams. Although these methodologies include these diagrams, a portion of them go unutilized (Munassar & Govardhan, 2011).

Use case diagrams are utilized to describe the way in which a system is employed by other systems, or its end users. A use case diagram displays how a system's use cases are correlated to one another and how end users interact with the system. Use case diagrams utilize a bubble to represent use cases, alongside stick figures to depict end users (Munassar & Govardhan, 2011).

Class diagrams display which classes exist in the system itself. The class diagram employs labelled boxes to represent classes. Alongside this, class diagrams reveal how objects of these classes can be joined together (Munassar & Govardhan, 2011).

Communication diagrams focus heavily on the linkage between objects (Munassar & Govardhan, 2011).

Deployment diagrams display the method in which the finalized product will be deployed to a singular or multiple devices. Deployment diagrams may consist of all kinds of components including dependencies, files, processes, and machines (Munassar & Govardhan, 2011)

Sequence diagrams display interactions similar to the communications diagram. However, as the name implies, the sequence diagram places a large emphasis on the sequence of the objects rather than the linkage. Sequence diagrams are utilized

amidst subsystem design; however, they are equally relevant to dynamic modelling during analysis, design, and even requirements capture. Sequence diagrams represent messages (commands) as arrows flowing between top labelled vertical bars that depict objects. In the diagram, time flows down the page in an incremental fashion (Munassar & Govardhan, 2011).

C. Preferred Approach: Object-Oriented Software Development

The researcher believes object-oriented software development would be the best approach for the development of the car parking system.

The approach is suitable for the car parking system, as it views the system as a group of objects interacting with each other. Alongside this, the method places a strong emphasis on the manipulation of objects instead of the actions needed to manipulate them. This is relevant inside the parking system as it will view each individual customer/vehicle as a separate object to be manipulated (Munassar & Govardhan, 2011).

The methodology also offers a host of benefits for the parking system's development including scalability, reusability, flexibility, and recurrency, alongside fast-paced development, and lower development cost.

Scalability of the car parking system can be managed through selecting the correct Object-Oriented approach before development begins. The great thing about object-oriented approaches is that they come in all types and sizes, geared towards different systems (Munassar & Govardhan, 2011).

Reusability is a key benefit of OOD (Object-Oriented Development) as it prevents developers from reinventing the wheel, instead, they can locate tools from past works to aid in the development of the car parking system, i.e., tools or objects (self-contained boxes filled with functionality). This may severely cut down the development time and cost of the project as less work will have to be performed in order to create the system (Universalteacherpublications, 2008).

Flexibility is provided to system developers in the form of tools during the development stages. System developers are able to utilize different tools such as textual representations (natural English), and a variety of graphical representations (diagrams) for their system during the development stages. The representations allow developers to better understand what they are developing (Wamicha, n.d.; Munassar & Govardhan, 2011).

Recurrency is the key difference between Object-Oriented approaches and Traditional approaches. What is meant by this is, unlike traditional approaches which have a rigidity to them, Object-Oriented approaches allow you to go back and revise previous phases before moving forward with the current phase. This allows for an immense number of modifications to be performed within individual phases without breaking the approaches (Toolshero, n.d.).

4. Object-Oriented Software Development Methodologies

A. Available Object-Oriented Methodologies

This section discusses four available methodologies that could be used for the automation of the car parking system.

A. Object Process Methodology (OPM)

Object Process Methodology (OPM) provides a methodology which focuses on the system as a whole rather than specific aspects of a given system i.e., the structure, behavior, and function. This is because OPM views processes as two sides of the same coin (Dori, n.d.; Wamicha, n.d.).

Objects are physical pieces of information (the input and output) that exist inside the system, while processes are the converters that are utilized to transform objects. Objects and processes converged, specify the behavior, structure, and function aspects of a modelled system within a singular framework (Dori, n.d.).

The primary motivation behind the development of Object Process Methodology was to provide system developers with a methodology which could provide a precise picture of how their system would be structured and how each component within the structured system would behave and function (Wamicha, n.d.).

OMT may be utilized to build two kinds of system representations, these are graphical representations and textual representations. Common entities shared between these representations are object, state, and process entities. This approach is formal yet intuitive (Wamicha, n.d.).

B. Rational Unified Process (RUP)

Rational Unified Process (RUP) is an iterative and agile methodology. The method is iterative as all phases are reoccurring throughout the project's development. The approach is agile because phases of the methodology may be revised until components in their development stages can be adapted to meet the requirements and objectives of the product (Toolshero, n.d.).

RUP is comprised of four phases, inception (statement of conception and resources), elaboration (assessment of resources and project architecture), construction (development and testing of product), and transition (releasing the product) (Toolshero, n.d.).

The method can be viewed through two dimensions, time, and process. The time dimension is represented by a horizontal axis which is expressed through such things as iterations, phases, and cycles. The time dimension can be defined as a dynamic organization of the process overtime (Toolshero, n.d.).

The second or process dimension can be perceived through a vertical axis. This dimension showcases the static aspect of the methodology and may be described through the use of artefacts, workflow, activities, and workers (Toolshero, n.d.).

C. Object Modelling Technique (OMT)

Object Modelling Technique (OMT) methodology was developed in 1991 by James Rumbaugh, Michael Blaha, William Premertani, Federick Eddy, and William Lorensen (Foundation, 2010).

OMT is a methodical approach to object-oriented analysis, design, and implementation that places a strong emphasis on constructing a model of objects from the real world and then utilizing it to develop object-oriented systems (tutorialspoint, n.d.).

The methodology is utilized to develop easily maintainable object-oriented systems. Alongside this, providing the purpose of allowing class attributes, methods,

inheritance, and association to be simply expressed utilizing the approach's models, namely the object model, dynamic model, and functional model (Liu, n.d.).

The methodology is comprised of four stages Analysis (identification of real-world situations), System Design (system architecture creation), Object Design (classification of remaining elements, relationships, and operations necessary for implementing a solution to the identified problem), and implementation (development of the system) (tutorialspoint, n.d.).

James Rumbaugh once stated, there is several reasons to use OMT, among them being making communication with customers easier, simulating the entities before constructing them, alongside aiding in reducing complexity using visualization to express ideas (Liu, n.d.).

Present day, the utilization of OMT in real-world scenarios is immense. The approach is predominately utilized by software developers to achieve full lifecycle development while targeting software modelling and designing (Liu, n.d.; tutorialspoint, n.d.).

D. Booch Methodology

The Booch methodology was created and authored by Grady Booch in 1992, while he was working for Rational Software. The Booch approach would later be revised in 1994 (Guha, n.d.). Booch is comprised of five activities (macro processes), Conceptualization (establishment of requirements and goals), Analysis (analysis and development of the systems base-model), Design (system architecture creation), Evolution or Implementation (refinement and development of the system), and Maintenance (refinement of the system and elimination of bugs) (Brown, n.d.).

Booch covers the design and analysis aspects of an Object-Oriented development system. The approach utilizes six distinct diagrams to perform this feat (Brown, n.d.).

The methodology prescribes a macro and micro development process. The macro process acts as a management framework for the micro processes and could take weeks or months. The key priority of the macro process is technical management of the development system. The technical management system is less focused on the object-oriented design aspect and more fascinated with how well the project performs with the requirements set forth, and the timeframe in which it is produced. Within the macro process, the traditional stages of analyzation and design are heavily preserved (Brown, n.d.).

Each macro development process consists of independent micro development processes. The micro processes represent tasks completed daily by a singular team member or small group of software developers. The process is comprised of the subsequent stages, Identification of classes and objects, and the corresponding semantics, relationships, interfaces and implementation (Brown, n.d.).

B. Tools and Techniques for each Methodology

A. Object Process Methodology (OPM)

In OPM, a system can be developed utilizing two types of representations, graphical representations, and textual representations. These representations are comprised of three kinds of entities, objects (things), states (conditions), and processes (actions that transform the state of a single object or collection of objects). Objects and states possess the ability to exist for a specific duration of time (Wamicha, n.d.).

The graphical representations are known as the object process diagrams (OPD). The OPDs are utilized to describe the objects, the links between objects, and the processes which transform the state of an object or collection of objects. The diagrams display both structural and procedural links utilizing a hierarchy of diagrams starting with by far the most generalized one, the system diagram. The lower grade diagrams provide an immense amount of detail (Wamicha, n.d.).

Structural links provide a direct link between different objects, and processes. The structural links also provide a graphical guide on how the system will be structured. Alongside this, procedural links provide an object to process link, and supply a visual of how the system will change over a given timespan (Wamicha, n.d.).

The textual representation is known as object process language (OPL). OPL was developed in such a manner that each construct of the OPDs can be explained using English, otherwise known as natural English. An OPD construct is any link between different objects, processes, or between objects and processes. An example of a construct would be, a link between a customer's ticket order, and a process for transforming the ticket order (Wamicha, n.d.).

The primary benefit of utilizing object process language is that it is easily readable and understood by non-technical individuals. Therefore, providing an easily comprehensible system for non-technical clientele who intend on implementing the system. In the long stretch, permitting customers to take part in the system development process beginning in the initial stages of the system's development (Wamicha, n.d.).

B. Rational Unified Process (RUP)

Rational Unified Process sets out a building-block-based system to utilize during development. The components that make up this system are artefacts (resources created during production), workers (responsibilities and expertise of team members), activities (set tasks for team members, most commonly creating or updating artefacts), and workflows (an illustrated sequence of activities, in an effort to produce trackable progress and artefact creation) (Airbrake, 2017).

The workflow can be separated into six distinct core engineering workflows, business modelling, requirements, analysis and design, implementation, test, and deployment (Airbrake, 2017).

The initial workflows are business modelling, and requirements. The business modelling workflow is where the scope of the project should be defined. Following this, the requirements workflow is utilized to outline every potential requirement of the project which could appear during the software development lifecycle (SDLC). These workflows act as an established baseline for the project and help to steer it towards perfection (Airbrake, 2017).

The ensuing workflow is analysis and design. During this workflow, the requirements from the previous phase are taken and transformed into a design that can accurately and efficiently be implemented (Airbrake, 2017).

The succeeding workflow is implementation. In this workflow, the design of the system is taken from the previous phase and implemented. Once implemented, the functions are organized into layers that make up the whole system. Following this, the test workflow begins. The test workflow utilizes testing methods to locate faults

within the product, allowing developers to repair the faults before release (Airbrake, 2017).

The last or deployment workflow is the workflow in which the product is finally released to customers. This workflow encompasses the entire delivery and release process, guaranteeing the product reaches the customer as expected (Airbrake, 2017).

Rational Unified Process also consists of three core supporting workflows, project management, configuration and change management, and environment (Airbrake, 2017).

The project management workflow is where all tasks handling project management take place, from pushing design goals to overcoming delivery constraints (Airbrake, 2017).

Configuration and change management workflow is utilized to describe a variety of artefacts developed by the development team, preferably ensuring minimal overlap or squandered endeavors conducting similar activities that produce a conflicting or identical artefact (Airbrake, 2017).

The environment workflow manages the setup and management of all software development environments throughout the development team, meaning, the environment workflow will manage the software and tools, which are going to be utilized throughout the software development lifecycle (SDLC) (Airbrake, 2017).

C. Object-Modelling Techniques (OMT)

Object Modelling Technique begins with setting out a problem statement based upon requirements provided by the stakeholders. Ensuing this, the methodology splits up the problem statement, assigning the pieces into the object, dynamic, and functional models, kicking off their establishment (Patel, 2019).

The object model (OM) is the beating heart of the methodology. The object model predominately concerns itself with classes, the relationships between classes, alongside class attributes and operations (Liu, n.d.). The approach utilizes a diagram comparable to the entity relationship diagrams (ERD) used in Yourdon. Relationships that can be drawn using the diagram of the object model include aggregation (a relationship where one object is the owner of other objects), association (the representation of unidirectional or bidirectional relationships between two classes), and inheritance (a relationship where a singular or multiple classes are built upon a base class) (Nexsoftsys, 2022).

In the object model, abstract or concrete classes are represented as a rectangular box. The rectangular box consists of three layers, the class name, the class attributes, and the class operations. The three layers are separated with a full line. A class name with a normal font represents a concrete class while italicized font indicates an abstract class (Liu, n.d.; Frost, 1993). An abstract class is a class which can be subclassed. However, it cannot be instantiated, and may or may not include abstract operations (Oracle, 2022).

Relationships in the diagram are most often illustrated using lines connecting the base classes to subclasses. When a single relationship is illustrated, a straight line is shown connecting the base class to the subclass. However, when multiple subclasses are connected to a single base class, a connection line will be shown

spanning the subclasses, connecting each of them to one another. Once the connection line connects each of the subclasses, a root line is drawn from the base class to the spanning connection line, connecting each of the subclasses to the base class. These relationships can also be stacked (Frost, 1993).

Inheritance (the relationships between the base class and subclasses) is shown within the diagram using a triangular shaped pointer positioned in the center of the root line, pointing towards the base class. Similar to inheritance, aggregation (the relationship where one object is the owner of the other objects) is displayed within the diagram utilizing a diamond symbol positioned closest to the base class end of the root line (Frost, 1993). Association can be displayed in many forms, the first form within the diagram simply displays a line with the number of associations. Additionally, a label may be added to the center of the line to specify the contents being shared between the classes. The second form of aggregation is a line with an empty or open circle which symbolizes optional associations. The third and last form of aggregation is a line with a filled or whole circle which indicates multiple associations (Frost, 1993; Liu, n.d.).

The dynamic model is comprised of a state transition diagram (STD) for each of the classes in the object model. The state transition diagram focuses mainly on the interaction between objects through events, states, and transitions. The notation utilized for the state transition diagram is the one defined by Harel in 1987, which allows for decomposable states (Frost, 1993; Liu, n.d.).

The functional model (FM) primarily centers its attention on data storage, data flow, alongside the different processes (Liu, n.d.). The model does this by utilizing a hierarchy of data flow diagrams (DFDs) in a similar manner to the Yourdon method. However, a downside of functional models is that they are weakly integrated with the dynamic and object models, meaning that they provide little to no use in tying together the project. An upside to functional models however, is the fact that they provide a context diagram which projects rely on to define the scope of the system (Frost, 1993).

According to Stuart Frost (1993), “it may be possible to strengthen the real-time aspects of the method by introducing the concept of a processor and task model using DFDs”.

D. Booch Methodology

The Booch notation was developed in 1992, alongside the Booch methodology. The notation is utilized alongside the methodology to aid in the development of systems. The Booch notation represents classes as cloud shapes and the classifications of these classes are performed utilizing six distinct diagrams, namely class, object, state transition, interaction, module, and process diagrams (Liu, n.d.; ConceptDraw, n.d.). The class and module diagrams are of static structure, while the state transition diagrams are of dynamic structure (Liu, n.d.).

Class diagrams describe the layout of a system in a manner displaying the system’s classes, and their operations, methods, and attributes, alongside their child object relationships. Classes are interpreted as boxes broken into three main compartments, the top, middle, and bottom. The top section conventionally contains the name of the class, centered, with the initial character being capitalized. The following (or middle) compartment is comprised of the class attributes (variables). Attributes are often formatted to the left with the first letter being lowercase. The

ensuing and last compartment is comprised of the operations, which the class can execute. They are commonly laid out to the left with the first letter being lowercase (ConceptDraw, n.d.).

The succeeding diagram in the Booch notation is the object diagram. The object diagram allows developers to display a partial or full view of their system. Developers also have the choice to pick examples from pre-existing structured sections of modelled systems, utilizing them separately, as a combination, or as drafts, and editing them to build upon what's already there (ConceptDraw, n.d.).

The ensuing diagram is the state transition diagram. The diagram is utilized from the early stages of development. State transition is utilized to define machine-based events, meaning, the diagram is utilized for defining a machine that is comprised of numerous states which takes input (events) from outside sources and switches states based upon prefixed conditions. An example of a prefixed condition could be, if a car is not on the blacklist, allow the car to enter the parking complex. Thus, the developers should be able to grasp an idea of what events could occur and what effects should happen based upon those events (ConceptDraw, n.d.).

The following and last diagram of the Booch notation is the interaction overview diagram. the interaction overview diagram is utilized for the purpose of visualizing activities in a sequence within the system's design. Each individual activities sequence is interpreted as a frame, and within these frames are nested diagrams (ConceptDraw, n.d.).

C. Recommended Methodology

The OOD (Object-Oriented Development) methodologies mentioned above are well established methods of system development across industries and have been well assessed within a variety of studies. Amidst the mentioned methodologies, the researcher believes RUP (Rational Unified Process) is the methodology best suited for developing the car parking system.

The reason why RUP is the best methodology for developing the car parking system is because of a few key reasons including integration, accuracy, evolution, and a minimization of risk factors, alongside the availability of a variety of online tutorials and training material (Sousa, 2009).

Integration is a key process within the methodology as it occurs throughout the SDLC (software development lifecycle), more specifically in the construction phase. This saves time for the project and provides a full view of the system which helps in locating potential flaws (bugs) (Mrsic, 2017).

RUP emphasizes the need (and proper implementation of) accuracy within documentation. This emphasis on documentation could save time in the long run for the development of the car parking system as problems are less likely to show up during the phases of the methodology due to this emphasis (Dopico, 2020).

Evolution is a focal point within the methodology as it allows developers to handle changing requirements regardless of the source i.e., from the project itself or from a stakeholder. This is an important feature for the methodology to have for the development of the car parking system as it will allow the system's development to change depending on the whims of the stakeholders, alongside the requirements of the project (Mrsic, 2017).

Risk factors are eliminated during the very early stages of the (SDLC) as the methodology places an emphasis on completing the high-risk factors components of the system first before moving on to lower risk factors. Alongside this, risks defined in one iteration of the development cycle can be prevented in the following cycles (AskingLot, 2020).

5. Problem Statement Matrix

Project: Automated Carpark System for the Spring Street Parking Building	**PROJECT MANAGER:** Murray Foote
CREATED BY: Codie Shannon	**LAST UPDATED BY:** Codie Shannon
DATE CREATED: 19/03/2022	**DATE LAST UPDATED:** 19/03/2022

Brief Statements of Problem, Opportunity, or Directive	Urgency	Visibility	Annual Benefits	Priority	Proposed Solution
Defective paper-based record system which causes problems such as ticket swapping, loss, and damage.	1 month	10	$500,000	Urgent	Replace part of the system
Opportunity for online payment options bringing more potential users due to ease of use.	4 months	6	$300,000	High	New development
Congestion is generated in the parking complex due to vehicles searching for unoccupied spaces.	1 month	10	$350,000	Urgent	Replace part of system
Poor customer management causing potentially dangerous users to enter the parking complex and commit acts of vandalism, violence, or vehicle theft.	7 months	3	$100,000	Medium	New development

References

Airbrake. (2017, March 14). What is Rational Unified Process and how do you use it? https://airbrake.io/blog/sdlc/rational-unified-process

AskingLot. (2020, May 11). *What are the advantages of unified process in software engineering?* https://askinglot.com/what-are-the-advantages-of-unified-process-in-software-engineering

Asmo. (2019, August 22). Waterfall Project Management: an Overview. *Zenkit.* https://zenkit.com/en/blog/waterfall-project-management-an-overview/

Bala, A. (2018). Comparison between Traditional and Object-Oriented Approach in Software Engineering. *International Journal of Research, 05, 01*, 2077-2082. https://journals.pen2print.org/index.php/ijr/article/download/11869/11198#:~:text=Traditional%20approach%20rely%20on%20sequential,important%20phase%20for%20software%20development%20

Benchthold, C. (2020, December 14). *The advantages of LPR.* Parqex. https://www.parqex.com/the-advantages-of-lpr/

Brown. (n.d.). *Booch Methodology.* https://bcanotes4u171863936.wordpress.com/booch-methodology/

ConceptDraw. (n.d.). *Booch OOD Diagram.* https://www.conceptdraw.com/How-To-Guide/booch-ood-diagram

Dopico, A. (2020, November 6). *What are the advantages of Unified Process?* JanetPanic. https://janetpanic.com/what-are-the-advantages-of-unified-process/

Dori, D. (n.d.). *Object-Process Methodology and its Applications for Complex, Real-Time Systems Modeling* [Report]. Technion, Israel institute of Technology and Massachusetts Institute of Technology. https://incoseil.org/check_download_nopass.php?forcedownload=1&file=84-dov%20dori.pdf&no_encrypt=true&dlpassword=87151

Foote, M. (n.d.). *Parking Building images* [Folder]. COMP.6209: Systems Analysis and Design in Moodle.

Foundation. (2010). *Object-Modeling Technique.* Academic. https://en-academic.com/dic.nsf/enwiki/135418

Frost, S. (1993). *Rumbaugh's OMT - the method behind C++ designer.* ACCU. https://accu.org/journals/overload/1/1/frost_1337/

Liu, A. (n.d.). *Rumbaugh, Booch and Jacobson methodologies.* OpenGenius. https://iq.opengenus.org/rumbaugh-booch-and-jacobson-methodologies/

Martin, M. (2022, February 18). *What is Waterfall Model in SDLC? Advantages and Disadvantages.* Guru99. https://www.guru99.com/what-is-sdlc-or-waterfall-model.html

Mrsic, M. (2017, August 9). *Rational Unified Process (RUP).* ActiveCollab. https://activecollab.com/blog/project-management/rational-unified-process-

rup#:~:text=The%20benefits%20of%20RUP,specifically%20in%20the%20construction%20phase

Munassar, N. M., & Govardhan, A. (2011). Comparison between Traditional Approach and Object-Oriented Approach in Software Engineering Development. *International Journal of Advanced Computer Science and Applications, 2(6)*, 131-138. https://www.researchgate.net/publication/344933723_Comparison_between_Traditional_Approach_and_Object-Oriented_Approach_in_Software_Engineering_Development

Nexsoftsys. (n.d.). *What is Association, Aggregation, Composition and Inheritance in Java with example?* https://www.nexsoftsys.com/articles/association-composition-aggregation-inheritance-java.html

Oracle. (n.d.). *Abstract Methods and Classes.* https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

Patel, P. (2019, March 11). *Software Engineering | Object Modeling Technique (OMT).* Geeksforgeeks. https://www.geeksforgeeks.org/software-engineering-object-modeling-technique-omt/#:~:text=Object%20Modeling%20Technique%20(OMT)%20is,static%20structure%20of%20the%20system

Qadri, T. M., & Asif, M. (2009). *Automatic Number Plate Recognition System for Vehicle identification using Optical Character Recognition* [Report]. Department of Electronic Engineering, Sir Syed University of Engineering & Technology. http://www.emmersion.com.au/wp-content/uploads/2015/05/AUTOMATIC-NUMBER-PLATE-RECOGNITION-SYSTEM-FOR-VEHICLE1.pdf

Rentcubo. (2021, December 6). *The problem with Manual Parking System.* https://rentcubo.com/the-problem-with-manual-parking-system/

Sharma, A. (2021, October 26). 8 reasons why LPR needs to make its way into your Parking Program's Budget. *Parking Industry.* https://www.parkingindustry.ca/parking-technology/8-reasons-why-lpr-needs-to-make-its-way-into-your-parking-programs-budget

SiegerParking. (2017, December 12). *The challenge of Manual Car Parking Systems.* https://puzzleparkingsystem.wordpress.com/2017/12/12/the-challenge-of-manual-car-parking-systems/

Sousa, S. (2009). *The advantages and disadvantages / best practices of RUP Software Development.* My PM Expert. http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-rup-software-development.html

Toolshero. (n.d.). *Rational Unified Process (RUP).* https://www.toolshero.com/information-technology/rational-unified-process-rup/

Tutorialspoint. (n.d.). *OOAD Object Modeling Techniques Q/A #1.* https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_modeling_qa1.htm

Uniconnect Systems. (2021). *LPR Parking System* [Photograph]. https://www.uniconnectsystems.com/other-services/car-parking-system-solutions/

Universalteacherpublications. (2008). *Advantages of Object Oriented Approach.* http://www.universalteacherpublications.com/univ/free-asgn/2008/mcs32/page2.htm#:~:text=Faster%20Development%3A%20OOD%20(Object%20oriented,locating%20and%20fixing%20problems%20easier.

Vashishtha, M., & Sharma, H. (2018, November 1). Artificial intelligence API for Car Parking Management System using stm32f407 and image processing using OpenCV[Photograph]. *Electronicforu.* https://www.electronicsforu.com/electronics-projects/prototypes/artificial-intelligence-api-car-parking-management-system

Wamicha, E. (n.d.). *What is Object Process Methodology?* Study. https://study.com/academy/lesson/what-is-object-process-methodology.html