HAJEE MOHAMMAD DANESH SCIENCE AND TECHNOLOGY UNIVERSITY
DINAJPUR-5200

Course Code: CSE 305

COURSE TITLE: SOFTWARE ENGINEERING

PROJECT TITLE

PERSONAL ASSISTANT(PA): A VOICE RECOGNISING SCHEDULE MANAGEMENT SYSTEM

Submitted By:

Name: Hazera Khatun

Student ID: 2102001

Session: 2021

Level: 03 **Semester: I**

Department: Computer Science and Engineering

Submitted To:

Pankaj Bhowmik

Lecturer

Department of Computer Science and Engineering

Hajee Mohammad Danesh Science and Technology University

DATE OF SUBMISSION: NOVEMBER 24, 2024

Introduction

This project introduces an outstanding experience of an artificial companionship which can plan, edit, remind and find one's schedules as per their command which can be manual, or voice controlled. Humans are forgetful in nature. They need some assistance from others to keep track of their plans and schedules and manage them efficiently to avoid unexpected consequences resulting from forgetfulness for any personal or unknown reason. But not everyone can afford a personal assistant. This project allows people to enjoy the assistance of an electronic device which can manage their schedules with a simple voice command as if they were talking to their personal assistant and it can remind them their plans the time they want (on the date of the plan, some days before or until some days before the plan) and the way they want (alarm, notification, mail, message, voice notification) without any mistake.

Users: People from all aspect of life i.e., student, office worker, teachers, housewives, businessmen and all kind of occupation and age more than 3 can be user of this project application.

Deciding SDLC model

The Software Development Life Cycle (SDLC) is a structured process that outlines the steps for developing software.

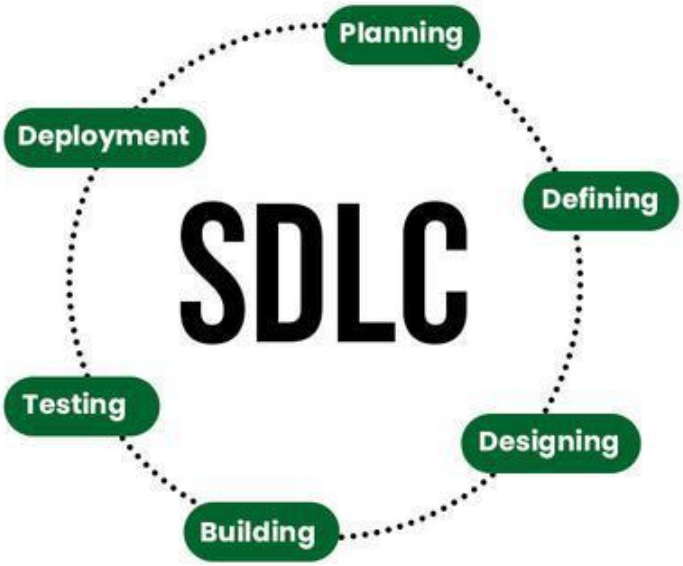


Fig-1: Graphical representation of the various stages of a typical SDLC

To this day, we have more than 50 recognized SDLC models in use. But None of them is perfect, and each brings its favourable aspects and disadvantages for a specific software development project or a team. We choose the **incremental model** for the following reasons:

1. Parallel development can be planned
2. Results are obtained early and periodically
3. Progress can be measured
4. Better suited for large and mission critical projects
5. Supports changing requirements
6. Easier to manage risk

Incremental model brief description

The incremental model is not a separate model. It is necessarily a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, each cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.[1]

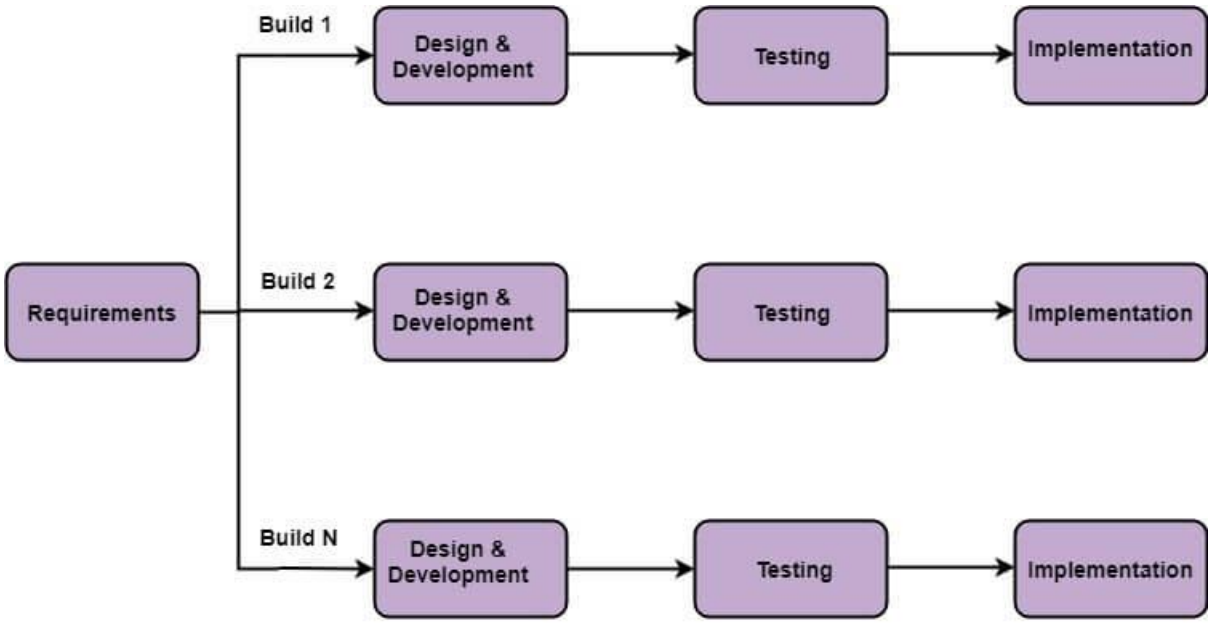


Fig: Incremental Model

We apply various stages of incremental model to get the final product of *Personal Assistant (PA): A Voice Recognising Schedule Management System*.

1. Feasibility study

As name suggests feasibility study is the feasibility analysis, or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view. Feasibility study is carried out based on many purposes to analyse whether software product will be right in terms of development, implementation, contribution of project to the organization etc. [2]

We covered the following aspects for the feasibility study of out project: [3]

1. **Technical Feasibility:** This aspect assesses whether the proposed software project is technically possible with the available technology, resources, and expertise. It considers software development tools, hardware requirements, integration with existing systems, and potential technical challenges.
2. **Operational Feasibility:** Operational feasibility evaluates whether the software will fit smoothly into the organization's existing processes and whether users can adapt. It

considers factors like user training, change management, and the impact on daily operations.

3. **Economic Feasibility:** This part of the study focuses on the financial aspects of the project. It includes a cost-benefit analysis to determine if the project is economically viable. This analysis involves estimating the development costs, maintenance costs, potential benefits, and return on investment (ROI). It helps stakeholders understand whether the project is financially justified.
4. **Scheduling Feasibility:** Scheduling feasibility assesses whether the project can be completed within the required timeframe. It considers factors like project scope, resource availability, and potential risks that might impact the project schedule.
5. **Legal and Regulatory Feasibility:** This examines whether the proposed software project complies with legal and regulatory requirements. It may involve data privacy, intellectual property rights, and industry-specific regulations.
6. **Market Feasibility (if applicable):** For commercial software projects, market feasibility examines whether there is a demand for the proposed software in the target market. It involves market research, competition analysis, and potential market adoption.

The outcome of a feasibility study of the project is **Go Decision.** That means, if the study concludes that the project is feasible regarding technical, operational, economic, scheduling, and legal aspects, the project can proceed to the next phase.

2. Requirement Analysis

Software requirement analysis simply means complete study, analysing, describing software requirements so that requirements that are genuine and needed can be fulfilled to solve problem. There are several activities involved in analysing Software requirements.

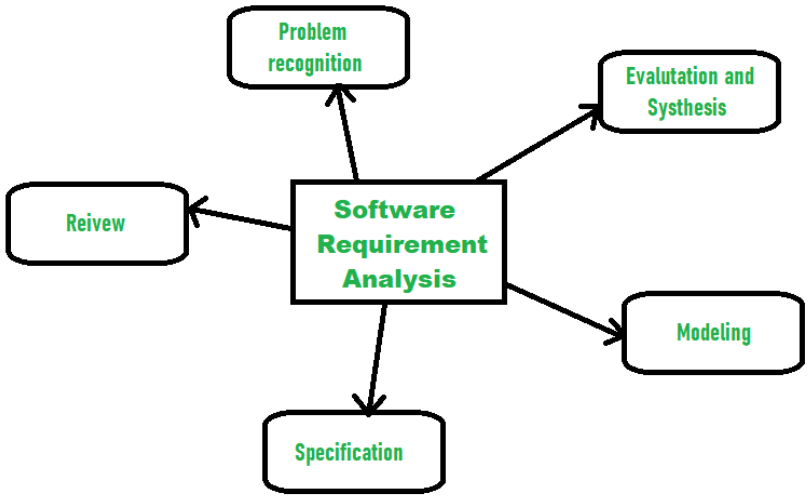


Fig-3: Activities involved in analysing Software requirements.

2.1 System Requirements

System requirements describe the technical specifications and constraints that a system must meet to fulfill user requirements. System requirements for *Personal Assistant (PA): A Voice Recognising Schedule Management System* include:

i. **Event management**: The ability to create, update, delete, search, and list event details

ii. **Event setup**: The ability to set up an event with a starting and ending date-time, event name, category, and placeholders for other details

iii. **Event modification**: The ability to modify event details after the event is scheduled, but not after the event is over

iv. **Data backup**: The ability to back up data periodically and instantly

v. **Auto archiving**: The ability to automatically archive data

vi. **Cross-browser compatibility**: The ability to work across browsers

vii. **Responsiveness**: The ability to be responsive

viii. **Graphical appeal**: The ability to be graphically attractive

ix. **Unicode support**: The ability to support Unicode-enabled fonts

x. **Multilingual support**: The ability to support multiple languages

xi. **Security testing**: The ability to pass security tests at the application level and architecture level

2.2 Functional Requirements

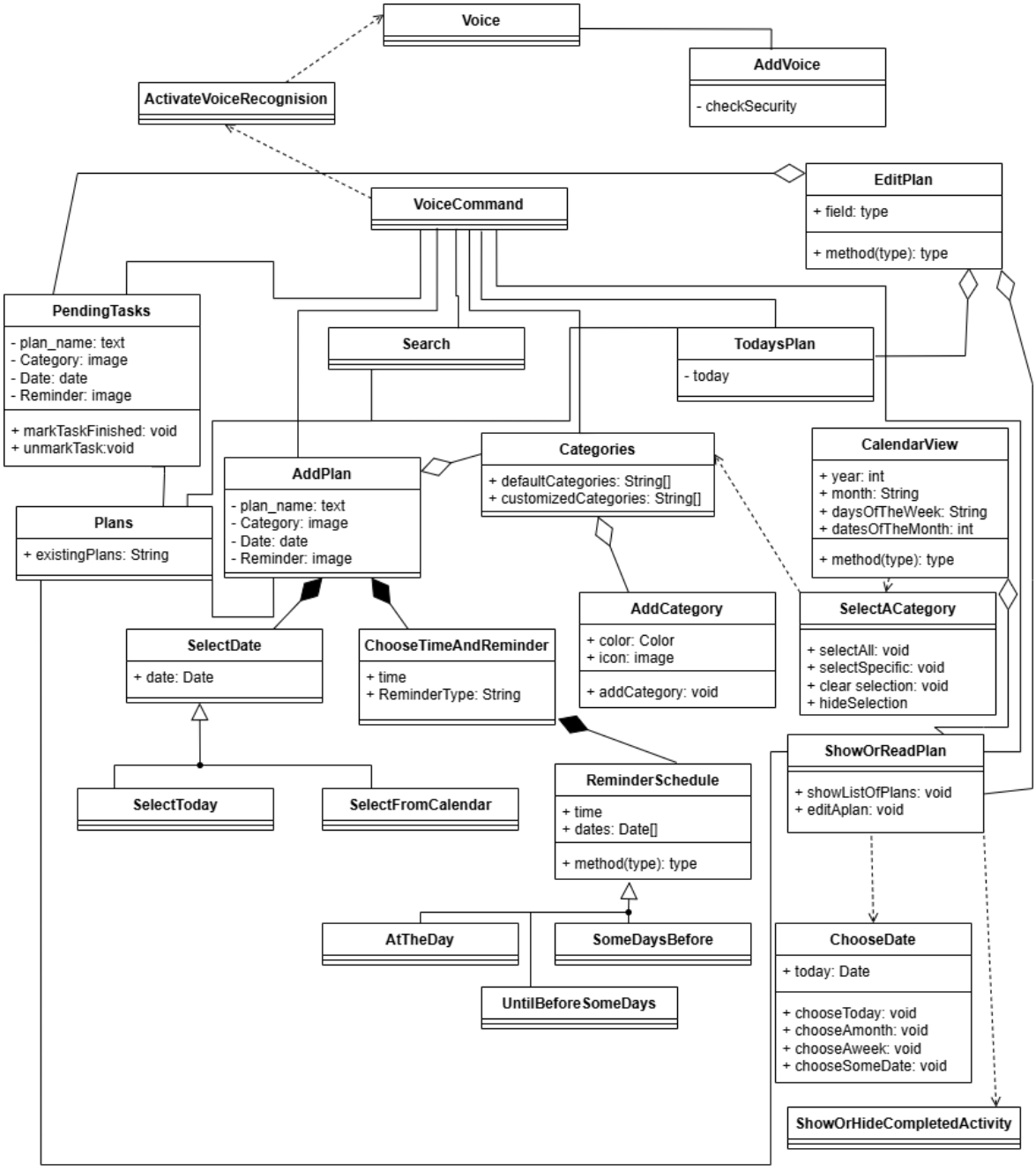
Functional requirements for *Personal Assistant (PA): A Voice Recognising Schedule Management System* include:

- **Voice command control:** The system must be able to recognize the voice and translate commands to perform specific tasks as the user asks.
- **Secured access:** To add a voice as its user, the system must check if he is the authorized user using the device's security system.
- **Addition of plans:** The system should be able to add new plans as per the user's command ensuring default values for the plan category, date, time and reminder type in case not mentioned.
- **Attractive calendar view:** In a calendar view, the user should be able to see the dates of a specific month (by default the present month) with planned schedules of some category (shows all category by default) marked.
- **Efficient search:** The system must perform accurate search operation for the user to find their plans.
- **Display/Read pending task:** The system must display the tasks that are not completed.
- **Display/Read today's plan:** The system must display or read aloud the tasks to be performed for the day.
- **Manage category:** The user should be able to categorize their plans or choose one from existing categories.

- **Notification management:** The system should be able to notify the user their plans the time they want (on the date of the plan, some days before or until some days before the plan) and the way they want (alarm, notification, mail, message, voice notification) without any mistake.

3. Design

The design phase is a stage where software developers define the technical details of the product. We include a class diagram for the *Personal Assistant (PA): A Voice Recognising Schedule Management System.*



The classes are described as follows:

- The **Voice, AddVoice, ActivateVoiceRecognition** and **VoiceCommand** classes are used for controlling voice operations.
  - Voice class stores the voices that has been added and used to verify voice command.
  - The AddVoice class ensures that the person trying to add a voice for further use is not an invader by doing a security check using device security system (password/ fingerprint/ pin etc.).
  - ActivateVoiceRecognition class ensures the system is ready to perform voice command with a specific signal from the user (specific speech/ tap on microphone icon visible in the system).
- **AddPlan** class is used to add new plan specifications (plan name, plan category, date, time and reminder type) as per the user's command ensuring default values in case not mentioned. Value for Date is by default Today (the very date the system being used) or can be selected from a calendar using **SelectDate** class and its specialized classes: **SelectToday** and **SelectFromCalendar**. Values for Reminder Time, Type and Schedule are by default 12 pm, alarm and the date specified for the plan consecutively. These values can be changed using **ChooseTimeAndReminder, ReminderSchedule** and its specialized classes: **AtTheDay, SomeDaysBefore, UntilBeforeSomeDays**.
- **EditPlan** class can edit the existing plans.
- **PendingTasks** class used to show the list of the tasks not finished yet and can be filtered using dates and category.
- **Categories** and **AddCategories** class is used to store default and customized categories or add new ones.
- **Search** class can search a plan/schedule from the system.
- **CalendarView** class shows a calendar that reperesents the dates of a specific month (by default the present month) with planned schedules of some category (shows all category by default) as marked (underlined or bolded). This class is dependent on the class **SelectACategory** for the selection of specific categorie(s).
- **ShowOrReadPlan** class shows the plans list and their details filtered as per the user wishes. It is dependent on the classes **ChooseDate**, **ShowOrHideCompletedActivity** to filter plans to display.

4. Develop

It is important to follow secure coding practices to ensure that the software is developed with security in mind. This may include using static analysis tools to identify potential vulnerabilities, following secure coding standards, and conducting regular code reviews.

This project needs voice recognition and high security. In this case, the programming language **Python** is better suited for *Personal Assistant (PA): A Voice Recognising Schedule Management System* because Python is the most widely used for artificial intelligence (AI) development, including voice and speech recognition. It's also one of the easiest languages to work with and is usually chosen because it supports most APIs and libraries.

5. Testing

The testing phase is integral to any software development project. It identifies bugs and errors introduced during the development phase, ensuring higher software quality and preventing costly mistakes down the line. Effective testing improves the overall customer experience and enhances the product's market reputation. [4]

5.1 Types of Testing [4]

There are several types of testing that you may encounter in the testing phase of the SDLC. Each type serves a specific purpose and helps ensure the software meets its intended requirements.

1) **Unit Testing:** Unit testing(opens in a new tab) is the process of isolating individual components or units of code.
2) **Integration Testing:** Integration testing(opens in a new tab) ensures that a system's different components work together as expected. This is important because even if individual units of code function correctly, they might not interact well with each other.
3) **System Testing:** System testing(opens in a new tab) evaluates the complete software system to verify it meets functional, performance, and security requirements. It involves testing various scenarios and edge cases to ensure the software handles real-world conditions effectively.
4) **Acceptance Testing:** Acceptance testing(opens in a new tab), or user acceptance testing (UAT), checks the software against user requirements and expectations. It's a key phase where potential issues can be identified and addressed before the software is released to the public.

5.2 Execution of test cases

Step no.	Description	Inputs	Expected Result	Actual Result	Status	Comment
1	Add a plan	a. plan name b. category c. date d. time and reminder	Plan added to the existing schedule list	As expected.	Pass	Successfully added plan
2	Edit a plan	Chage the a. plan name and/or b. category and/or c. date and/or d. time and reminder	The edited Plan displayed in the list.	The plan was edited and displayed	Pass	Plan Successfully edited

3	Search a plan	Eating with friends	No result as this plan was not in the list	Showed no result	Pass	Perfect searching operation
4	Show plans	a. set date: today b. set category: All	Shows three plans that were scheduled for today	Showed the plans perfectly	Pass	Successful output
5	Voice command testing	Command the system the step no. 1-4	Read aloud the corresponding expected results of step 1-4	As expected.	Pass	Successful voice command execution

Note: As the aim of this project report is to show the way we can use SDLC model in a project application, the project was not actually built. So, we considered the test case results to be successful in each test case scenarios.

6. Deployment [5]

This is the final step in the software development life cycle and delivers the final product to the customer in a live production environment. After the product deploys, the product is ready for customers to use.

To deploy the product in the market, we do a cost-benefit analysis. **Cost-benefit analysis**, or CBA, is a data-driven approach to evaluating a project or decision's financial benefits and costs from a business perspective. By forecasting profitability through a CBA, teams can work to avoid financial loss. The key components of cost-benefit analysis are costs, benefits, timeframes, and discount rates. These components help project managers efficiently calculate a business's costs and benefits.

We can assess the following costs throughout the CBA process:

- **Direct costs:** We can trace direct costs to producing a specific product or service, including labor, materials, supplies, and wages.
- **Indirect costs:** We can't link indirect costs to producing goods or services. These costs include office rent, administrative salaries, utilities, and overheads.
- **Intangible costs:** We can identify intangible costs, but measuring them in monetary value is difficult. Examples of intangible costs include decreases in productivity, loss of goodwill, and customer dissatisfaction.
- **Opportunity costs:** Opportunity costs refer to choosing one project or strategy over another. For instance, allocating resources to develop a new feature for a software project rather than improving existing features represents an opportunity cost of potentially enhanced user satisfaction and retention.

After identifying the costs, it’s crucial to recognize the benefits of projects that CBA measures:

- **Tangible benefits**: Tangible benefits are easily quantified and measured in terms of monetary value. Examples include revenue growth, cost savings, and increased efficiency.
- **Intangible benefits:** Similar to intangible costs, intangible benefits are difficult to measure in monetary value. These benefits include enhanced reputation, employee satisfaction, and customer loyalty.

When conducting a cost-benefit analysis, you must consider both short-term and long-term costs and benefits:

- **Short-term:** Short-term cost-benefit analysis gives an idea of the immediate results we can expect from our project. For example, hiring temporary staff for a project increases immediate payroll expenses.
- **Long-term:** Long-term analysis provides a broader picture of the project's feasibility. For instance, investing in new equipment involves maintenance and replacement costs.

Note: As the aim of this project report is to show the way we can use SDLC model in a project application, the project was not actually built. So, the costs and benefits for the project was not negotiated in monetary term.

Future Enhancement

There are some more creative ideas for the project that was not designed in the beginning but was left for future development according to user responses and acceptance feedback. Some of them are described below:

1. **Online platform:** This system can be developed as online platform for easier access from any place and any device and ensure preventing data loss.
2. **Social network:** The system can be updated as a social network that can connect people to form some community to plan meetings dynamically considering the free schedule times of the persons associated.
3. **Manage profile:** Each user can decide their plans to be public or private in a community for efficient meeting planning.
4. **Create meeting:** Users connecting in a community can request a meeting or reject a meeting request.

Conclusion

The *Personal Assistant (PA): A Voice Recognising Schedule Management System* enhances an artificial intelligence companionship enjoyment in addition to managing ones daily or periodical planning avoiding bizarre consequences resulting from the forgetfulness of important tasks. It also satisfies user's inner comfort of being assisted by a PA (personal assistant) with extremely low chances of mistakes and discords. Looking ahead, this project paves the way for ongoing enhancements, adaptability to user needs, and a sustained impact on digital well-being.

References

[1] Incremental Model (Software Engineering) - javatpoint. (n.d.). www.javatpoint.com. https://www.javatpoint.com/software-engineering-incremental-model

[2] GeeksforGeeks. (2024, May 10). *Types of feasibility study in software project development*. GeeksforGeeks. https://www.geeksforgeeks.org/types-of-feasibility-study-in-software-project-development/

[3] *Feasibility Study in Software Engineering - javatpoint*. (n.d.). www.javatpoint.com. https://www.javatpoint.com/feasibility-study-in-software-engineering

[4] Talreja, A., & Agile, T. (2024, July 18). *Testing phase in SDLC: ensuring quality and reliability*. https://teachingagile.com/sdlc/testing

[5] Atlassian. (n.d.). *Cost benefit analysis: What is it and how to do it*. https://www.atlassian.com/work-management/strategic-planning/cost-benefit-analysis