Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Academic Year : 2024-25

Experiment No.1
Apply Software Process Models for a case study
Name: Krisha Chikka
TE/1 Rollno.: 30
Date of Performance: 10/07/2024
Date of Submission: 15/07/2024

Aim: To apply Software Process Models for a case study.

Objective: To prepare Waterfall Model, Spiral Model, RAD Model for Currency Detector for Visually Impaired.

Theory:

Software Process Model

A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed.

A model will define the following:

- The tasks to be performed
- The input and output of each task
- The pre and post conditions for each task
- The flow and sequence of each task

The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.

Waterfall Model:

The waterfall model is a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project. Each activity in the waterfall model is represented as a separate phase arranged in linear order.

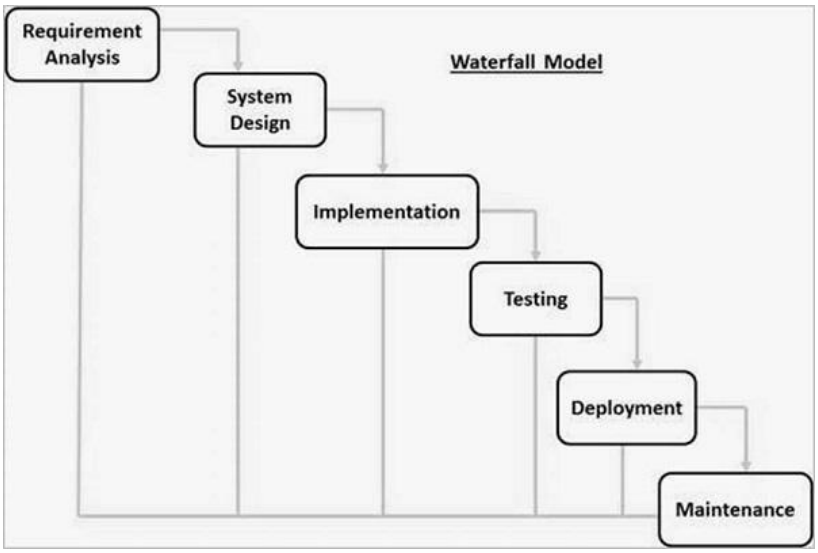
It has the following phases:

- Requirements
- Design
- Implementation
- Testing
- Deployment
- Maintenance

Each of these phases produces one or more documents that need to be approved before the next phase begins. However, in practice, these phases are very likely to overlap and may feed information to one another.

The software process **isn't linear**, so the documents produced may need to be modified to reflect changes.

The waterfall model is easy to understand and follow. It doesn't require a lot of customer involvement after the specification is done. Since it's inflexible, it can't adapt to changes. There is no way to see or try the software until the last phase.



The waterfall model has a rigid structure, so it should be used in cases where the requirements are understood completely and unlikely to radically change.

Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.

- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Spiral Software Process Model

A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed.

A model will define the following:

- The tasks to be performed
- The input and output of each task
- The pre and post conditions for each task
- The flow and sequence of each task

The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.

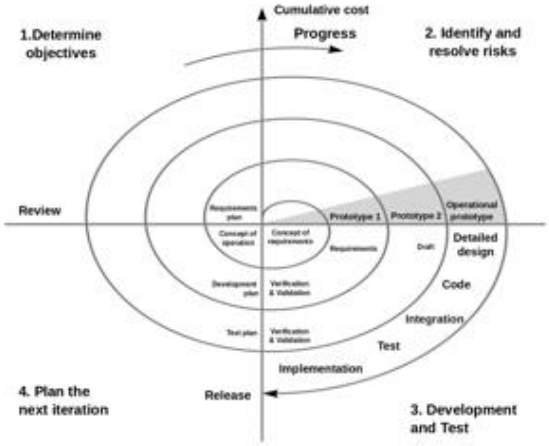
Spiral Model:

The spiral model is a risk driven iterative software process model. The spiral model delivers projects in loops. Unlike other process models, its steps aren't activities but **phases** for addressing whatever problem have the greatest risk of causing a failure.

It was designed to include the best features from the waterfall and introduces risk-assessment. You have the following phases for each cycle:

1. Address the highest-risk problem and determine the objective and alternate solutions
2. Evaluate the alternatives and identify the risks involved and possible solutions
3. Develop a solution and verify if it's acceptable
4. Plan for the next cycle

You develop the concept in the first few cycles, and then it evolves into an implementation. Though this model is great for managing uncertainty, it can be difficult to have stable documentation. The spiral model can be used for projects with **unclear needs** or projects still in research and development.



Application

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e., learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model −

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

RAD Software Process Model

A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed.

A model will define the following:

- The tasks to be performed
- The input and output of each task
- The pre and post conditions for each task
- The flow and sequence of each task

The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.

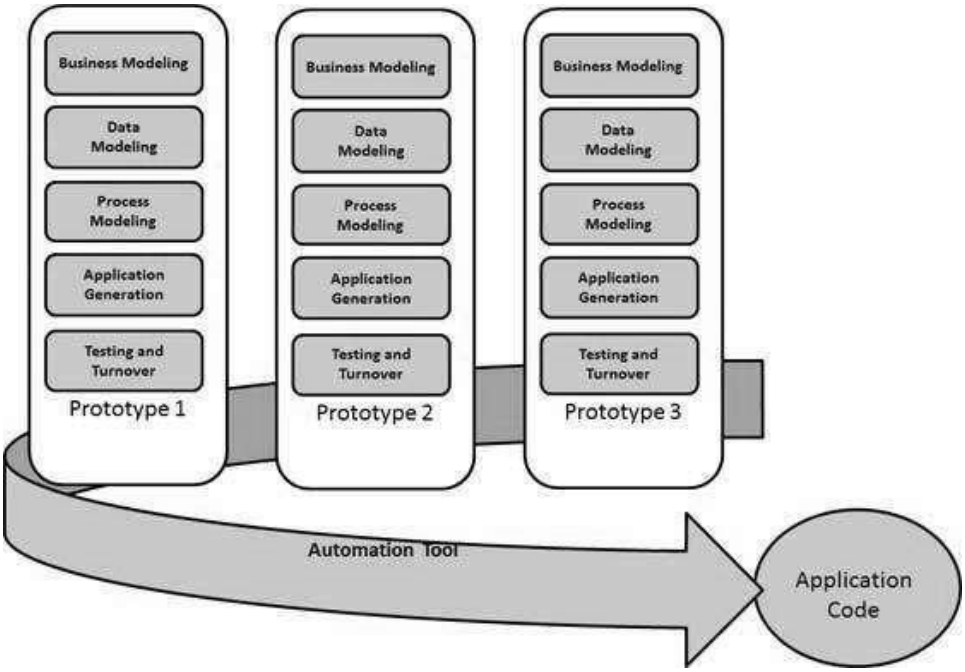
RAD Model:

The Rapid Application Development (RAD model) is based on iterative development and prototyping with **little planning involved**. You develop functional modules in parallel for faster product delivery. It involves the following phases:

1. Business modelling
2. Data modelling
3. Process modelling
4. Application generation
5. Testing and turnover

The RAD concept focuses on gathering requirements using focus groups and workshops, reusing software components, and informal communication.

The RAD model accommodates changing requirements, reduces development time, and increases the reusability of components. But it can be complex to manage. Therefore, the RAD model is great for systems that need to be produced in a short time and have known requirements.



Application

RAD model can be applied successfully to the projects in which clear modularization is possible. If the project cannot be broken into modules, RAD may fail.

The following pointers describe the typical scenarios where RAD can be used −

- RAD should be used only when a system can be modularized to be delivered in an incremental manner.
- It should be used if there is a high availability of designers for Modelling.
- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

Solution:

Waterfall Model:

Currency Identification Application for Visually Impaired Individuals

Document Owner: Krisha Chikka

Version: 1.0

Date: 22/07/2024

Status: Draft

Case Study:

To design a waterfall model for currency detector for visually impaired.

PROBLEM STATEMENT:

Visually impaired individuals encounter significant challenges when identifying currency during everyday transactions, such as making purchases. This difficulty often leads to reliance on others or increased risk of errors, which can undermine their independence and confidence in handling financial matters. To address this issue, there is a need for a solution that enables these individuals to independently and accurately identify different denominations of currency. The proposed solution is to develop a mobile application designed specifically for visually impaired users. This application will utilize the device's camera to detect and recognize currency notes. It will also incorporate voice recognition and activation commands for ease of use, allowing users to interact with the app hands-free. Furthermore, the application will provide currency denomination information through audio output, ensuring that users receive clear and immediate feedback on the currency they are handling.

1. RESEARCH PHASE

1. Project Scope :

- Defining the specific goals of the currency detector project for visually impaired users and develop a currency detector for visually impaired individuals that accurately identifies and communicates denominations of currency notes.
- Boundaries: Focus on recognizing a predefined set of currency notes and ensuring usability by visually impaired users.

2. StakeHolder Interviews:

Stakeholders:

- Project Sponsor

- Project Manager

- Development Team

- End Users: Visually impaired individuals

We will have to conduct interviews with stakeholders (such as visually impaired individuals, caregivers, and experts in accessibility) to understand their needs, challenges, and expectations.

3. Research Reviews:

Review existing technologies: Study current currency detectors and accessibility tools.

Relevant literature: Explore research on image processing algorithms, audio feedback systems, and accessibility standards.

4. Requirements Gathering:

1. The system shall activate upon hearing a predefined voice command ("Activate" or equivalent).
2. Upon successful currency detection, the system shall accurately announce the denomination in a clear and understandable audio format suitable for visually impaired users.
3. The application interface shall comply with accessibility standards to ensure usability for users with visual impairments.
4. The text-to-speech output for currency denomination shall include options for adjusting speed and volume to accommodate user preferences.

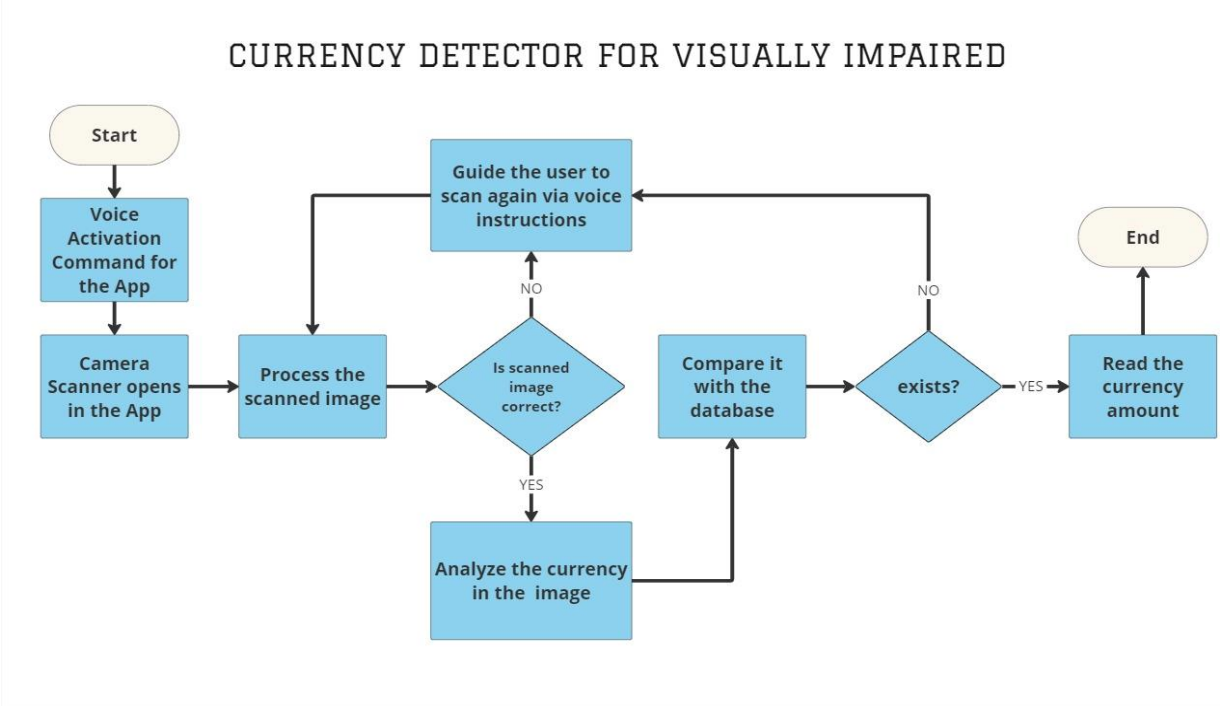
5. Kickoff Meeting :

Discuss project functions and gather initial requirements.

Define functional and non-functional requirements based on stakeholder inputs and research findings.

2. DESIGN PHASE

Flowchart:



Design, views by the stakeholders

Develop a flowchart outlining the currency detection process.

Incorporate stakeholder feedback to ensure clarity and usability.

3. DEVELOPMENT PHASE

Implement the currency detector according to the design specifications established in the previous phase. This involves writing code, integrating necessary hardware components (such as sensors and audio output devices), and ensuring compatibility with accessibility standards.

1. Implement software for image processing and denomination recognition.
2. Integrate camera module for capturing currency images.

3. Develop audio feedback system for communicating denomination to users.

Progress Reviews:

- Regularly review and assess development progress.
- Ensure adherence to design specifications and accessibility standards.

4. TESTING PHASE

Revision of the testing will be performed

I. Unit Testing:
   Test image processing algorithms for accuracy.
   Verify audio feedback system functionality.

II. Integration Testing:
   Ensure seamless interaction between image capture, processing, and audio output.

III. System Testing:
   Conduct end-to-end tests with visually impaired users.

Validate accuracy and usability under various conditions.

Ensure seamless interaction between image capture, processing, and audio output.

5. DEPLOYMENT PHASE

1. Pilot Deployment:

- Install currency detectors at pilot sites for initial user feedback.
- Gather feedback to refine the device and improve user experience.

2. User Training and Documentation:
   - Provide training sessions for users on how to use the currency detector.
   - Prepare documentation that is accessible and easy to understand.
3. Final Deployment:
   - Deploy the finalized product to stakeholders.
   - Support ongoing maintenance and updates based on user feedback and technological advancements.

Spiral model:

Document Owner: Krisha Chikka

Version: 1.0

Date: 22/07/2024

Status: Draft

Case Study:

To design a Spiral model for currency detector for visually impaired.

1. Project Initiation

Project Scope Definition:

The project aims to develop a mobile application for visually impaired individuals to identify currency notes using voice commands and camera-based recognition technology. The application will provide real-time audio feedback on currency denominations.

Stakeholders:

- Project Sponsor

- Project Manager
- Development Team
- End Users: Visually impaired individuals

Initial Research:

Conducted consultations and user research to understand the needs of visually impaired users and reviewed existing solutions in the market.

Requirements Gathering:

- Functional Requirements:
  - Implement voice command interface for initiating currency identification.
  - Develop camera-based recognition system for accurate currency identification.
  - Provide real-time audio output announcing currency denomination.

- Customer Feedback:

Collect preliminary feedback from visually impaired individuals regarding their needs and preferences for currency identification tools. This will help refine initial requirements and ensure the project aligns with user expectations from the start.

2. Planning and Designing

First Spiral Cycle:

Prototype Development:

- Develop initial prototypes focusing on voice command processing and basic camera integration.
- Conduct usability tests with visually impaired users to gather feedback on prototype features and accessibility.

Design Refinement:

- Revise flowcharts and interface designs based on user feedback and initial testing results.

- Ensure alignment with functional and non-functional requirements, emphasizing accessibility and ease of use.

- Customer Feedback:

Gather detailed feedback on the initial prototypes from users, focusing on voice command accuracy, camera performance, and overall usability. Use this feedback to make informed adjustments in design and functionality.

3. Development and Implementation

Second Spiral Cycle:

Development Phase:

- Implement voice command interface and integrate camera functionality based on refined design specifications.

- Develop initial currency recognition algorithms and integrate with prototype system.

- Conduct internal testing to verify basic functionality and integration of core features.

Incremental Build:

- Release iterative builds to gather user feedback on usability and performance.

- Enhance system capabilities based on feedback and additional requirements identified during initial testing.

4. Evaluation and Risk Management

Third Spiral Cycle:

Evaluation Phase:

- Perform comprehensive testing including unit, integration, and system testing to validate functionality and performance.

- Solicit feedback from stakeholders and end users to assess satisfaction with application features and usability.

Risk Management:

- Continuously monitor and mitigate identified risks throughout development cycles.

- Adjust development priorities based on emerging issues and user feedback to ensure project success.

- Customer Feedback:

Collect feedback from users during each build iteration to understand improvements and additional needs. This includes assessing the effectiveness of voice commands, accuracy of currency recognition, and overall user satisfaction.

5. Deployment and Maintenance

Final Spiral Cycle:

Deployment Preparation:

- Prepare application for deployment on major app stores, adhering to platform-specific guidelines and standards.

- Conduct final performance testing and security audits prior to deployment.

Customer Feedback:

- After deployment, gather feedback from users through reviews, surveys, and usage data.

- Evaluate the feedback to identify common issues and areas for improvement.

- Use the insights to refine and enhance the application in subsequent iterations, ensuring it evolves to better meet user needs and expectations.

RAD model:

Document Owner: Yash Chavan

Version: 1.0

Date: 22/07/2024

Status: Draft

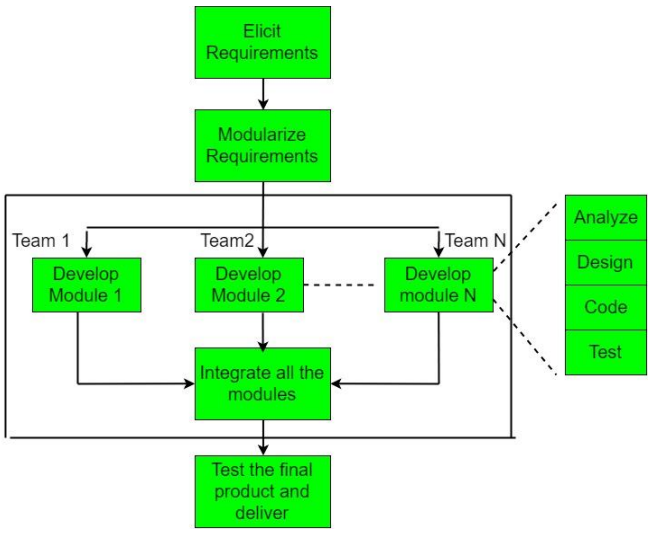
Case Study:

To design a Spiral model for currency detector for visually impaired.

PROBLEM STATEMENT:

Visually impaired individuals encounter significant challenges when identifying currency during everyday transactions, such as making purchases. This difficulty often leads to reliance on others or increased risk of errors, which can undermine their independence and confidence in handling financial matters. To address this issue, there is a need for a solution that enables these individuals to independently and accurately identify different denominations of currency. The proposed solution is to develop a mobile application designed specifically for visually impaired users. This application will utilize the device's camera to detect and recognize currency notes. It will also incorporate voice recognition and activation commands for ease of use, allowing users to interact with the app hands-free. Furthermore, the application will provide currency denomination information through audio output, ensuring that users receive clear and immediate feedback on the currency they are handling.

RAD Model Solution for Currency Detection Application



1. **Elicit Requirements**

Objective: Gather and define the needs of visually impaired users to create a detailed list of requirements for the currency detection application.

Tasks:

- Conduct consultations and user research to understand the challenges faced by visually impaired users in identifying currency.
- Document both functional and non-functional requirements:
    - **Functional:** Implement a voice command interface, integrate camera-based recognition for currency, and provide real-time audio feedback.
    - **Non-Functional:** Ensure compliance with accessibility standards, implement robust security measures, and guarantee high usability.

Outcome: A detailed list of requirements outlining the project's scope and deliverables.

2. Modularize Requirements

Objective: Break down the requirements into manageable development modules.

Tasks:

- **Module 1: Voice Command Interface**
    - **Analyze:** Define the capabilities of voice commands and outline user interaction scenarios.
    - **Design:** Develop user flows and processes for handling voice commands.

- **Code:** Implement and integrate voice command functionalities.
- **Test:** Verify the accuracy and reliability of the voice command processing.

- **Module 2: Camera-Based Recognition System**
  - **Analyze:** Identify the specific requirements for recognizing currency using the camera.
  - **Design:** Create camera integration plans and recognition algorithms.
  - **Code:** Develop and integrate camera functionalities and recognition algorithms.
  - **Test:** Validate the accuracy of currency identification through the camera.
- **Module 3: Database Comparison and Output Display**
  - **Analyze:** Develop an AI model to compare scanned images with database records.
  - **Design:** Design the interface for integration between the application and the database.
  - **Code:** Implement the logic for database comparison and audio output.
  - **Test:** Ensure that audio feedback is accurate and timely.

3. Develop Modules 1...N

Objective: Build and implement each module as specified.

Tasks:

- **Module 1: Voice Command Interface**
  - **Develop:** Code voice command functionality and integrate with other app features.
  - **Test:** Ensure accuracy and reliability of voice recognition through various accents and speech patterns.
- **Module 2: Camera-Based Recognition System**
  - **Develop:** Implement camera integration and currency recognition algorithms.
  - **Test:** Validate the system's accuracy and performance under different conditions.
- **Module 3: Database Comparison and Output Display**
  - **Develop:** Build AI for currency comparison and integrate with the app's database. Implement audio feedback for currency identification.
  - **Test:** Check the accuracy of currency identification and ensure timely audio feedback.

4. Integrate All Modules

Objective: Combine all individual modules into a cohesive final product.

Tasks:

- Integrate developed modules (voice command interface, camera-based recognition, audio feedback, accessibility, and security features).
- Perform integration testing to ensure seamless functionality across all components.

Outcome: A unified application with all modules working together harmoniously.

5. Test the Final Product

Objective: Confirm that the final product meets all quality standards and is ready for user deployment.

Tasks:

- Conduct final performance testing and security audits.
- Prepare the application for deployment on app stores, adhering to platform-specific guidelines.
- Finalize user documentation and guides.

Outcome: A fully functional, tested, and ready-to-deploy application.

6. Deliver

Objective: Finalize and deploy the application.

Tasks:

- **Performance Testing:** Ensure the app performs well across devices.
- **Security Audits:** Verify that the app is secure and compliant with data protection standards.
- **Prepare for Deployment:** Follow app store guidelines and complete the submission process.
- **Finalize Documentation:** Create user guides and technical documentation.

Outcome: A fully functional, tested, and deployable application with complete documentation.

Conclusion:

In conclusion, the comparison of the RAD, Spiral, and Waterfall models highlights the strengths and weaknesses of each approach in software engineering. The RAD model excels in rapid development and user feedback, making it suitable for projects with tight timelines. The Spiral model offers a robust framework for risk management and iterative development, ideal for complex projects. In contrast, the Waterfall model provides a structured and straightforward methodology, best for projects with well-defined requirements. Ultimately, the choice of model should align with project goals, team dynamics, and stakeholder needs.