Scientific Research Publishing

Journal of Software Engineering and Applications, 2024, 17, 553-570
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

HV Process Model of Software Development

Hemant Kumar, Vipin Saxena

Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, India
Email: hemant20192@gmail.com, profvipinsaxena@gmail.com

How to cite this paper: Kumar, H. and Saxena, V. (2024) HV Process Model of Software Development. *Journal of Software Engineering and Applications*, **17**, 553-570.
https://doi.org/10.4236/jsea.2024.177032

Received: April 19, 2024
Accepted: July 12, 2024
Published: July 15, 2024

Abstract

Software Development Life Cycle (SDLC) is one of the major ingredients for the development of efficient software systems within a time frame and low-cost involvement. From the literature, it is evident that there are various kinds of process models that are used by the software industries for the development of small, medium and long-term software projects, but many of them do not cover risk management. It is quite obvious that the improper selection of the software development process model leads to failure of the software products as it is time bound activity. In the present work, a new software development process model is proposed which covers the risks at any stage of the development of the software product. The model is named a Hemant-Vipin (HV) process model and may be helpful for the software industries for development of the efficient software products and timely delivery at the end of the client. The efficiency of the HV process model is observed by considering various kinds of factors like requirement clarity, user feedback, change agility, predictability, risk identification, practical implementation, customer satisfaction, incremental development, use of ready-made components, quick design, resource organization and many more and found through a case study that the presented approach covers many of parameters in comparison of the existing process models.

Keywords

Software Process Model, Software Development, Software Engineering, Software Risk Management and Software Quality

1. Background

Primary aim of developing a software process model is to carry out effectively the different development stages of software engineering. There are steps in each phase of software development with switching from one stage to another stage.

Software industries have motive to develop efficient software products within specified timeline of software development. Generally, timeline of development of software product is setup by project leader as per need of client and by conducting a formal meeting with client and manpower involved during development of software product. The quality of software products is based upon the selection of a process model based on the duration of a software project. In the literature, there are numerous software development process models, like Waterfall [1], Prototype [2], Iterative and Incremental [3], Rapid Application Development (RAD) [4], Agile Process Model [5], Spiral Model [6] and many more which have several advantages and disadvantages but play vital roles during the development of small, medium and long term software projects. Waterfall model is the base of most of the software development process models. Some of the software development process models cover risk management activity which is an important activity during development and improper handling of such activity leads to failure of the development of software product. The Waterfall model has the limitations that the freezing of requirements occur at the time of collection of the requirements from the client. Another limitation of the model is that there is concept of baseline, once the team moves ahead into the next phase of software development then cannot move into the previous phase. Iterative and incremental model is suitable for the short term project especially related to the update of software product in which the earlier phases of development are considered as the prototype. RAD model has fixed limitation of the total development period which is divided into two equally parts but does not consider the risks at any stage of development. It is generally uses to develop prototype software. Agile process model is also uses for the iterative development of software products and successive refinements produce high quality of software but does not cover the risks at any stage of software development. Spiral model is suitable for the large project as it covers the risk items during development of the software products but does not cover the customer satisfaction; user feed-back is not available, and many more concepts of software development are not covered in this process model.

2. HV Process Model

To handle risk management activity at each level of development and other concepts during software development, a new software process model is proposed in the present work which is named as a HV software process development model and is useful for the software industries as it covers risks involved during development of software product at every stage of development. The proposed HV process model is represented below in **Figure 1**.

The idea in this process model is based on the requirements that may be completed by the client during the development of software projects. The proposed model is explained below in brief:

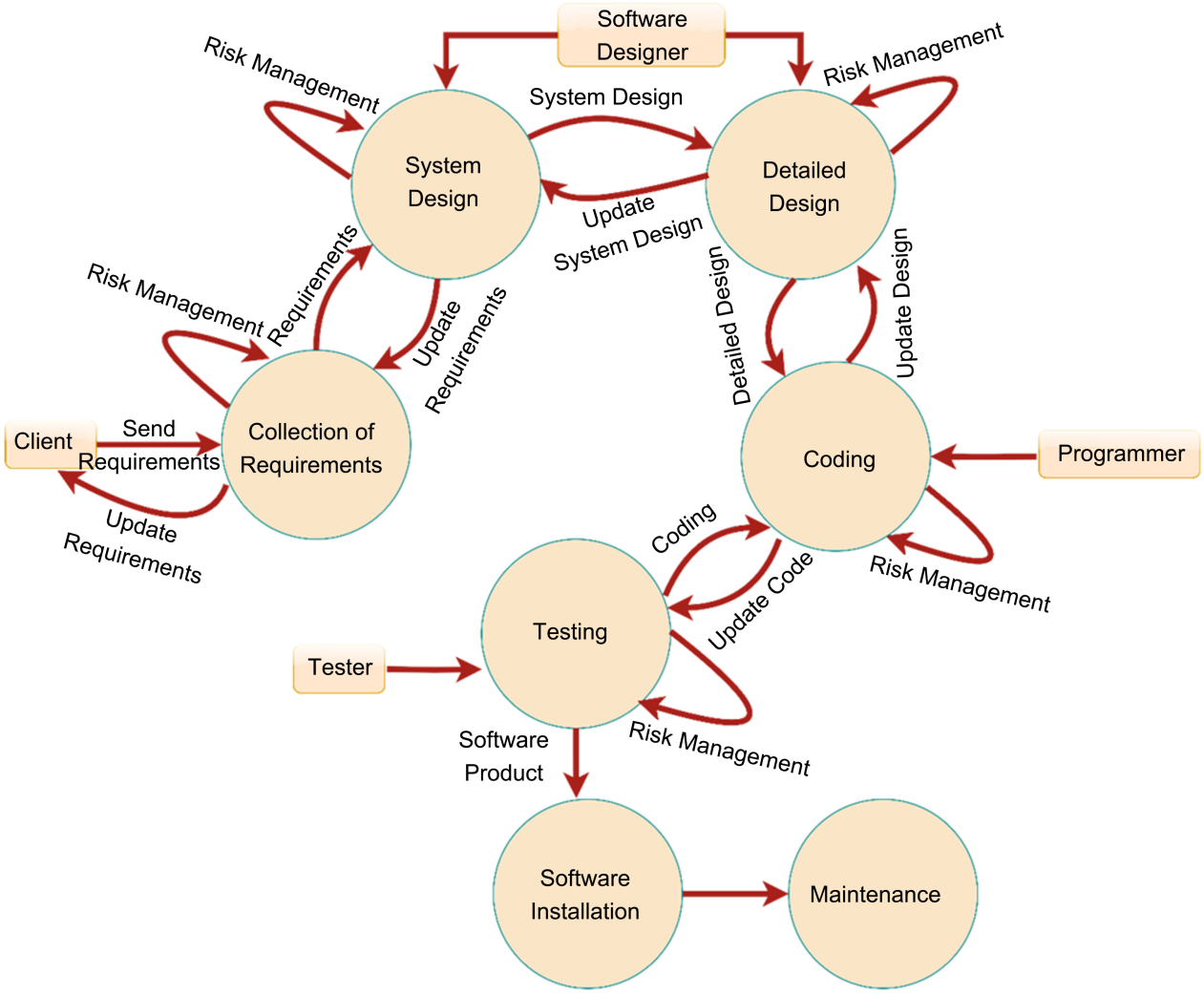


Figure 1. HV software development process model.

1) Collection of Requirements

Before the development of the software product, the collection of requirements is a major activity based on the inputs given by the client. The development of software products is assigned to a team consisting of project leader, system analyst, system administrator, software designer, software programmers, software testers, etc. The requirements given by the client are discussed through a meeting among the team members and these are finalized by the committee members including client. For the medium and long term software projects, the requirements are written in the form of a document called as a Software Requirement Specification (SRS) document. The proposed HV process model has the flexibility to update/modify the requirements during the development of software product as one can move backward for updating the requirements added by the client at the later stage of software development. Another factor is the costing related to added of the requirements at later stage, which may also be added by the project leader to the overall cost of the project. The reason is to

propose such activity that the software industries are functioning around 365 × 24 × 7 hours and involved manpower is also allowed to work from home. Due to this reason, there is possibility for timely delivery of the software product to the client. This activity is performed by considering the risk items involved in the requirements which are to be minimized by the project leader. The various kinds of suggested risk items are explained later in the present work.

2) System Design

After preparing the requirement document *i.e.* SRS, a proper analysis of supporting software and hardware is done which are required for the development of software products and thereafter, a design based on the object-oriented or latest software design technology is proposed in the form of a blueprint called software design. This design has the flexibility to interact with the proposed classes and objects encapsulated. In the HV Process model, when the requirements are modified by the client, then the system design has the flexibility to update the proposed system design of the software product. These types of activities are performed by the software designers and are approved by the project leader. This phase also covers the risk items which may occur during the period of system design which may be optimized through mathematical techniques by the project leader. The various kinds of risk items involved during this phase are described later.

3) Detailed Design

Further, the HV Process model consists of detailed designing of software products which involves the activities of component-level designing in the form of blueprint based on either object-oriented or the latest technology supported by the programming languages. The procedure of each component involved in the system design is to be written particularly nearer to programming language. Further, the HV process model containing detailed design has the flexibility to change as per the need of the client. The changes in the detailed designs shall be approved by the project leader. The component design is updated in such a way that the complexity shall not be increased and overall will not affect the complexity of the system design. The various kinds of the risk items involved in the detailed design shall be considered and optimized under the risk management activity. The various kinds of risk items involved during this phase are described later.

4) Coding

The software coding is an integral part of the development of software product. In the HV process model, coding is proposed according to the blueprint design by the software designer and all rules and regulations in terms of minimization of complexity of code are applicable. In the HV Process model, the code has the flexibility to change/update when the requirements are changed/updated by the client and accordingly, the cost of development may be modified by the project leader. Further, the model consists of the various kinds of risk items involved during the coding phase which may be minimized through managerial activity performed by the project leader. The various kinds of risk items involved

during this phase are described later.

5) Testing

Software testing is one of the major activities involved in the software development. It involves a systematic and adaptive process for evaluating different parts (modules or sub-modules) of a computer program to be written in the latest programming language. The methodology aims to ensure thorough testing coverage while accommodating the modular structure of software products. The entry of test cases into the testing loop depends on the software modules. In the HV Process model, testing strategies are based upon the selection of test cases either from the requirements or from the software design *i.e.* covering black and white box testing strategies. Further, testing has its own features which are not available in the other software process models and the feature is that the client may also provide the test cases which are used to check the quality and blueprint of the software product. The test case proceeds to be executed, and the results are stored. A passing test case contributes to the validation of the associated module, while a failing test case triggers the generation of a detailed failure report. The testing process iterates continuously until all test cases are executed, ensuring a comprehensive evaluation of each module or sub-module. At the end of the testing process, reports are generated, encompassing detailed reports for failed test cases and a comprehensive report for untested modules or sub-modules. This iterative and adaptable approach creates a resilient testing framework in the HV process model that aligns seamlessly with the modular structure of the software. Further various risk items are also considered in this phase which shall be minimized by the project leader and the various kinds of risk items involved during this phase are described later.

6) Software Installation and Maintenance

After careful execution of the stages mentioned above for the development of the software product, the software installation and maintenance is not a development activity while during these activities, the software product may be installed at the end of client or customers. As per the feedback of the client and customer, the updated version of software product is to be designed or maintained under maintenance activity.

3. A Case Study

Let us consider a case study that works according to the HV process model. Creating a login page in software development requires careful planning to ensure that it meets user needs and operates smoothly. In this case study, HV process model is used as a framework that combines flexibility, responsiveness to user feedback, and a systematic approach to software development. Unlike traditional models, the HV process model allows for changes in user requirements while staying within budget constraints, enabling the development team to work efficiently.

Let's dive into the details of each phase of the present case study, starting with requirements collection. In this initial phase, the team members of the software

project will interact with users, discuss the login page needs, and create a comprehensive requirements document. Join us on this exploration of the HV process model and see how it supports a dynamic and efficient approach to creating a login page.

3.1. Collection of Requirements

Before delving into the intricacies of software development, it's crucial to establish a clear understanding of what the client envisions. The requirements collection phase lays the foundation by assembling the client's needs, desires, and specifications. Through collaborative discussions and meticulous documentation, a comprehensive requirements document emerges, guiding the subsequent phases of the development journey as shown in the following Table 1.

Table 1. Collection of requirements for login page design.

Requirement ID	Requirement Description
REQ-001	Capture user credentials: username and password
REQ-002	Implement a "Forgot Password" feature for account recovery
REQ-003	Enable users to register for new accounts
REQ-004	Ensure password security measures (e.g., encryption)
REQ-005	Support multi-factor authentication (if required)
REQ-006	Design a responsive layout for various devices and browsers
REQ-007	Customize the login page to align with the brand's aesthetics
REQ-008	Provide error messages for incorrect login attempts
REQ-009	Include a "Remember Me" option for user convenience
REQ-010	Integrate with backend systems securely (using Python)

3.2. System Design

With a detailed understanding of the client's requirements, the system design phase steps into the limelight. This is where the architects of the software ecosystem design the high-level blueprint, mapping out the system's structure and interactions. The focus is on ensuring scalability, adaptability, and a solid foundation for the forthcoming detailed design and coding phases. The following Table 2 gives a representation of the high-level system design or blueprint for the login page.

Table 2. System design for login page design.

Component	Description
Frontend	Implement the login page using HTML, CSS, and JS
Backend	Develop the backend logic using Python

Continued

Database	Store user credentials securely
Security	Implement encryption for password storage
User Interface	Design a clean and intuitive user interface
Responsiveness	Ensure the login page is responsive on all devices
Error Handling	Set up error messages for various scenarios
New Registration	Develop the registration feature for new accounts
Forgot Password	Implement functionality for password recover
Remember Me	Include an option for users to stay logged in
Multi-Factor Auth	Integrate support for multi-factor authentication
Branding	Customize the login page to match the brand's style
Testing	Plan and execute testing for both frontend and backend
Deployment	Deploy the login page to a staging and production environment
Monitoring	Implement monitoring for performance and security
Maintenance	Outline procedures for ongoing maintenance and updates

The above table provides a structured overview of the essential design features for the login page.

3.3. Detailed Design

As the system design sets the overarching framework, the detailed design phase dives into the finer details. Here, components are meticulously crafted, each line of code conceptualized, and every module's intricacies are addressed. The objective is to translate the high-level blueprint into a comprehensive plan that developers can follow to breathe life into the software. The following Table 3 shows a representation of the detailed design for the login page.

Table 3. Detailed design for login page design.

Component	Sub-Component	Description
Frontend	HTML Structure	Define the structure of the HTML elements for the login page
	CSS Styling	Specify the styling details, such as colors, fonts, and layout
	JavaScript Functionality	Implement client-side functionality for interactive features
Backend	Python Routing	Establish routes for handling login, registration, and other actions
	User Authentication Logic	Develop logic to verify user credentials securely
	Password Encryption	Implement a secure method for encrypting and storing passwords
Database	User Table Schema	Define the structure of the user table, including necessary fields
	Data Storage	Specify the database system and methods for data storage

Continued

Security	HTTPS Implementation	Ensure secure communication between the client and server
	Cross-Site Scripting (XSS)	Implement measures to prevent XSS attacks
	Cross-Site Request Forgery (CSRF)	Protect against CSRF attacks with token validation
User Interface	Login Form Design	Design the visual layout and elements of the login form
	Error Message Display	Specify how error messages will be displayed for user feedback
Responsiveness	Media Queries	Implement responsive design using CSS media queries
Error Handling	Validation Checks	Define client-side and server-side validation checks
	Logging and Monitoring	Implement logging for error tracking and monitoring
New Registration	Registration Form Design	Design the visual layout and elements of the registration form
	User Input Validation	Ensure valid user input during the registration process
Forgot Password	Password Reset Mechanism	Design the process for users to reset their passwords
Remember Me	Persistent Login Token	Implement a secure mechanism for 'Remember Me' functionality
Multi-Factor Authentication	Integration	Integrate support for additional authentication factors
Branding	Branding Guidelines	Follow branding guidelines for colors, logos, and overall theme
	Customization Options	Provide options for clients to customize the login page appearance
Testing	Test Cases	Develop test cases for unit testing, integration testing, and user acceptance testing
	Testing Environment Setup	Set up testing environments for various testing phases
Deployment	Staging Deployment Process	Define the process for deploying the login page to a staging environment
	Production Deployment Process	Outline the steps for deploying the login page to the live production environment
Monitoring	Performance Monitoring	Set up tools and processes for monitoring page performance
	Security Monitoring	Implement mechanisms to monitor and respond to security events
Maintenance	Update Procedures	Define procedures for updating the login page with new features or fixes
	Support and Bug Fixing	Establish a system for addressing user-reported issues and bugs

The above table provides a detailed breakdown of the components and sub-components involved in the design of the login page, offering a comprehensive guide for the implementation phase.

3.4. Coding

Armed with a detailed design, developers step onto the coding canvas to weave intricate lines of logic. The coding phase is where algorithms are implemented, functions come to life, and the software takes shape. It's a journey from conceptualization to realization, where the artistic finesse of coding meets the precision demanded by the design. The complete coding for login design phase is given below:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Login Page</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
            margin: 0;
            background-color: #f2f2f2;
        }

        #loginContainer {
            background-color: #ffffff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            padding: 20px;
            text-align: center;
            max-width: 400px;
            width: 100%;
        }

        form {
            display: flex;
            flex-direction: column;
            align-items: center;
            max-width: 300px;
            margin: 0 auto;
        }

        label {
            margin-bottom: 8px;
            text-align: left;
            width: 100%;
        }

        input {
            padding: 10px;
            margin-bottom: 16px;
            width: 100%;
        }

        button {
            background-color: #4CAF50;
            color: white;
            padding: 12px;
            border: none;
            cursor: pointer;
            width: 100%;
        }

        #rememberMeContainer {
            display: flex;
            align-items: center;
            width: 100%;
            text-align: left;
            margin-bottom: 16px;
        }

        #rememberMeCheckbox {
            margin-right: 8px;
        }
```

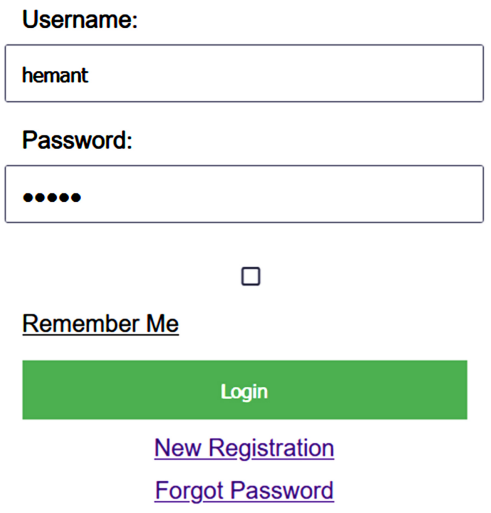
```
        #rememberMeCheckbox label {
            margin: 0;
        }

        #registrationLink,
        #forgotPasswordLink,
        #rememberMeContainer {
            margin-top: 10px;
            color: #555;
            text-decoration: underline;
            cursor: pointer;
            display: block;
        }
    </style>
</head>
<body>
    <div id="loginContainer">
        <form id="loginForm" action="/login" method="post">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>

            <label for="password">Password:</label>
            <input type="password" id="password" name="password"
required>

            <div id="rememberMeContainer">
                <input type="checkbox" id="rememberMeCheckbox"
name="rememberMe">
                <label for="rememberMeCheckbox">Remember Me</label>
            </div>

            <button type="button" onclick="login()">Login</button>
        </form>

        <form id="registrationForm" action="/register" method="post"
style="display:none;">
            <label for="newUsername">New Username:</label>
            <input type="text" id="newUsername" name="newUsername"
required>

            <label for="newPassword">New Password:</label>
            <input type="password" id="newPassword" name="newPassword"
required>

            <button type="button"
onclick="register()">Register</button>
        </form>

        <form id="forgotPasswordForm" action="/forgot_password"
method="post" style="display:none;">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required>

            <button type="button" onclick="resetPassword()">Reset
Password</button>
        </form>

        <div id="registrationLink"
onclick="showForm('registrationForm')"><a href="#">New
Registration</a></div>
        <div id="forgotPasswordLink"
onclick="showForm('forgotPasswordForm')"><a href="#">Forgot
Password</a></div>
    </div>
```

```
    <script>
        function login() {
            var rememberMeChecked =
document.getElementById('rememberMeCheckbox').checked;

            if (rememberMeChecked) {
                document.cookie = "rememberMe=true; expires=Thu, 31 Dec
2030 12:00:00 UTC; path=/";
            } else {
                document.cookie = "rememberMe=false; expires=Thu, 01
Jan 1970 00:00:00 UTC; path=/";
            }

            document.getElementById('loginForm').submit();
        }

        function register() {
            document.getElementById('registrationForm').submit();
        }

        function resetPassword() {
            document.getElementById('forgotPasswordForm').submit();
        }

        function showForm(formId) {
            document.getElementById('loginForm').style.display =
'none';
            document.getElementById('registrationForm').style.display =
'none';
            document.getElementById('forgotPasswordForm').style.display
= 'none';

            document.getElementById(formId).style.display = 'block';
        }
    </script>
</body>
</html>
```

```
from flask import Flask, render_template, request, redirect, url_for,
flash, make_response
from werkzeug.security import generate_password_hash,
check_password_hash
import datetime

app = Flask(__name__)
app.secret_key = 'secret_key'

users = [{'username': 'user', 'password':
generate_password_hash('password')}]

@app.route('/')
def index():
    remember_me = request.cookies.get('rememberMe')
    return render_template('index.html', remember_me=remember_me)

@app.route('/login', methods=['POST'])

def login():
    username = request.form['username']
    password = request.form['password']
    remember_me = True if request.form.get('rememberMe') else False

    user = next((user for user in users if user['username'] ==
username), None)
```

```
    if user and check_password_hash(user['password'], password):
        flash('Login successful', 'success')

        if remember_me:
            resp = make_response(redirect(url_for('index')))
            resp.set_cookie('rememberMe', 'true',
expires=datetime.datetime.now() + datetime.timedelta(days=365))
            return resp
    else:
        flash('Invalid username or password', 'error')

    return redirect(url_for('index'))

@app.route('/register', methods=['POST'])
def register():
    new_username = request.form['newUsername']
    new_password = request.form['newPassword']

    if not any(user['username'] == new_username for user in users):
        hashed_password = generate_password_hash(new_password)
        users.append({'username': new_username, 'password':
hashed_password})
        flash('Registration successful', 'success')
    else:
        flash('Username already exists', 'error')

    return redirect(url_for('index'))

@app.route('/forgot_password', methods=['POST'])
def forgot_password():
    email = request.form['email']
    flash('Password reset email sent to {}'.format(email), 'info')
    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(debug=True)
```

In the above coding, a snapshot of the output of the above code:



The following manual test cases are considered and represented below in the following Table 4.

Table 4. Test cases for login page design.

Test Case ID	Description	Test Steps	Expected Result	Pass/Fail
TC-01	Verify Page Elements	1) Open the login page. 2) Check for username, password, and login button. 3) Verify the presence of labels for username and password.	Page contains username field, password field, and login button. Labels for username and password are present.	Pass
TC-02	Attempt Login with Valid Credentials	1) Enter valid username and password. 2) Click the login button.	Login is successful. User is directed to the expected page or receives a success message.	Pass
TC-03	Attempt Login with Invalid Username	1) Enter an invalid username 2) Enter a valid password . 3) Click the login button.	Login fails. Appropriate error message is displayed.	Pass
TC-04	Attempt Login with Invalid Password	1) Enter a valid username. 2) Enter an invalid password 3) Click the login button.	Login fails. Appropriate error message is displayed.	Pass
TC-05	Attempt Login with Empty Fields	1) Leave both username and password fields empty. 2) Click the login button.	Login fails. Error messages prompt the user to fill in both fields.	Pass
TC-06	Verify Remember Me Functionality	1) Check the “Remember Me” option. 2) Enter valid credentials. 3) Click the login button. 4) Close and reopen the browser. 5) Revisit the login page.	Username is pre-filled when the page is revisited after closing and reopening the browser.	Pass
TC-07	Check Page Responsiveness	1) Open the login page on a desktop browser. 2) Resize the browser window to various dimensions.	Page layout adjusts appropriately to different screen sizes.	Pass
TC-08	Verify Styling and Branding	1) Ensure login page adheres to specified styling rules. 2) Verify branding guidelines for colors, logos, and overall theme.	Login page adheres to specified styling rules and branding guidelines.	Pass
TC-09	Test New Registration Link	1) Check for the presence of a registration link. 2) Click on the registration link.	Navigates to the registration page or a related section.	Pass
TC-10	Test Forgot Password Link	1) Check for the presence of a “Forgot Password” link. 2) Click on the “Forgot Password” link.	Navigate to the password recovery page or a related section.	Pass

4. Risk Management in HV Process Model

During the development of the software project, various kinds of risks occurs known as risk items which may hamper the scheduled delivery of the software product to the client. The following prominent risk items are listed below which are resolved based on priority so that the development process should not be hampered.

In the below Table 5, five phases of software development are considered and various types of risk items and corresponding losses are mapped as $F: R \rightarrow L$ which is considered as a learning function in which loss L depends on the risk item R. Hence set of R is taken as $R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5$ in which $R_1 \rightarrow \{R_{11}, R_{12}, \ldots, R_{1I}\}$, $R_2 \rightarrow \{R_{21}, R_{22}, \ldots, R_{2J}\}$, $R_3 \rightarrow \{R_{31}, R_{32}, \ldots, R_{3K}\}$, $R_4 \rightarrow \{R_{41}, R_{42}, \ldots, R_{4L}\}$ and $R_5 \rightarrow \{R_{51}, R_{52}, \ldots, R_{5M}\}$ and similar interpretation is given to the set of L as $L_1 \cup L_2 \cup L_3 \cup L_4 \cup L_5$ in which $L_1 \rightarrow \{L_{11}, L_{12}, \ldots, L_{1I}\}$, $L_2 \rightarrow \{L_{21}, L_{22}, \ldots, L_{2J}\}$, $L_3 \rightarrow \{L_{31}, L_{32}, \ldots, L_{3K}\}$, $L_4 \rightarrow \{L_{41}, L_{42}, \ldots, L_{4L}\}$ and $L_5 \rightarrow \{L_{51}, L_{52}, \ldots, L_{5M}\}$. The training set is defined as (R_1, L_1), (R_2, L_2), (R_3, L_3), (R_4, L_4) and (R_5, L_5). The mapping of risk item to corresponding loss is considered as one-to-one mapping controlled by the mapping function.

Table 5. Risk items during the development of login page design.

Name of Phase	*Risk Item Number*	*Name of Risk Item*	*Explanation*	*Loss due to Risk Item*
Collection of Requirements	R_{11}	Unclear User Expectations	Ambiguities in understanding user expectations may lead to requirements misunderstandings.	L_{11}
	R_{12}	Inadequate Stakeholder Communication	Lack of effective communication with stakeholders may result in missing important requirements.	L_{12}
	…	…	…	…
	R_{1I}	Evolving User Needs	Changing user needs during the requirement collection phase can introduce scope creep and challenges.	L_{1I}
System Design	R_{21}	Insufficient Architecture Planning	Poor planning of the system architecture may result in scalability and performance issues.	L_{21}
	R_{22}	Technology Integration Risks	Challenges associated with integrating different technologies may lead to implementation difficulties.	L_{22}
	…	…	…	…
	R_{2J}	Late Identification of Design Constraints	Identifying design constraints late in the system design phase may require rework and adjustments.	L_{2J}
Detailed Design	R_{31}	Lack of Modularity	A design lacking modularity may lead to code maintenance challenges and reduced flexibility.	L_{31}
	R_{32}	Coding Standards Not Followed	Non-compliance with coding standards may impact the maintainability of the software.	L_{32}
	…	…	…	…
	R_{3K}	Insufficient Documentation	Inadequate documentation may hinder the understanding of the detailed design by the development team.	L_{3K}

Continued

Coding	R_{41}	Coding Errors	Introduction of errors during the coding phase may result in functional or security issues.	L_{41}
	R_{42}	Code Duplication	Repetitive code segments may increase the likelihood of introducing bugs and maintenance challenges.	L_{42}
	…	…	…	…
	R_{4L}	Non-Adherence to Coding Standards	Deviating from coding standards may lead to inconsistencies and decreased code quality.	L_{4L}
Testing	R_{51}	Incomplete Test Coverage	Lack of coverage for all aspects of the software may result in undiscovered defects.	L_{51}
	R_{52}	Test Data Issues	Issues with test data may lead to incomplete testing and inaccurate assessment of system behavior.	L_{52}
	…	…	…	…
	R_{5M}	Regression Test Challenges	Managing and executing regression tests efficiently is crucial for ensuring system stability.	L_{5M}

Let us compute the expected loss due to risk items in the present case study. There are five stages of the software development in the first stage, $R_{1I} \in R$ is the input and $L_{1I} \in L$ is the output and it can be evaluated from $F_1(R_{1L})$. the probability distribution $P(R_{1I}, L_{1I})$ over R_{1I} and L_{1I} is defined by over the training set (R_{11}, L_{11}), (R_{12}, L_{12}), …, (R_{1I}, L_{1I}), which are discrete random variables. The risk associated with $F_1(R_{1I})$ is given by

$$R_1 = \sum_{I=1}^{N} R_{1I} * L_{1I} \tag{1}$$

Similarly, the other stages, it is given by,

$$R_2 = \sum_{J=1}^{N} R_{2J} * L_{2J} \tag{2}$$

$$R_3 = \sum_{K=1}^{N} R_{3K} * L_{3K} \tag{3}$$

$$R_4 = \sum_{L=1}^{N} R_{4L} * L_{4L} \tag{4}$$

$$R_5 = \sum_{M=1}^{N} R_{5M} * L_{5M} \tag{5}$$

The total risk involved during the development of software is given by

$$R = \sum_{I=1}^{5} R_I \tag{6}$$

Average risk

$$R_A = \frac{1}{5} \sum_{I=1}^{5} R_I \tag{7}$$

Let's illustrate the risk calculation for the different phases of proposed case study. The computations are listed below in the following **Table 6**.

Table 6. Computation of total risks during development of login page design.

Name of Phase	Risk Item	Probability with Associated Risk (R)	Loss (L)	Risk Value (R × L)	Overall Risk
Collection of Requirements	R_{11}	0.2	0.1	0.02	0.08
	R_{12}	0.15	0.2	0.03	
	R_{13}	0.1	0.3	0.03	
System Design	R_{21}	0.25	0.15	0.0375	0.1075
	R_{22}	0.2	0.25	0.05	
	R_{23}	0.1	0.2	0.02	
Detailed Design	R_{31}	0.18	0.12	0.0216	0.0666
	R_{32}	0.15	0.18	0.027	
	R_{33}	0.12	0.15	0.018	
Coding	R_{41}	0.22	0.2	0.044	0.1106
	R_{42}	0.18	0.22	0.0396	
	R_{43}	0.15	0.18	0.027	
Testing	R_{51}	0.2	0.15	0.03	0.093
	R_{52}	0.18	0.2	0.036	
	R_{53}	0.15	0.18	0.027	

From the above table, it is observed that there are more chances of risk in the coding phase while less chances in the phase of detailed design. It may vary from one phase to another phase. The total risk in all phases is $R = R_1 + R_2 + R_3 + R_4 + R_5 = 0.4567$. The proposed model is compared with the software development process models available in the literature and compiled below in Table 7.

Table 7. Existing software development process models Versus HV process model.

Technical Aspect	Water-fall Model [1]	Prototype Model [2]	Incremental Model [3]	RAD Model [4]	Agile Model [5]	Spiral Model [6]	HV Process Model
Requirement Clarity	Yes	Medium	Yes	Yes	Incremental Change	Yes	Yes
User Feedback	No	Yes	No	No	No	No	Yes
Change Agility	Low	Medium	High	No	High	High	High
Predictability	Low	High	Low	Low	High	Medium	High
Risk Identification	No	No	No	Yes	Yes	Yes	Yes
Practical Implementation	No	Medium	Low	No	High	Medium	High
Customer Satisfaction and Incremental Development	No	No	No	No	Incremental Customer Satisfaction and Development	No	Yes

Continued

Use of Ready-made Components	No	No	No	No	No	No	Yes
Risk Identification at Each Stage	No	No	No	Yes	Yes	Yes	Yes
Systematic Sequence	Yes	No	No	No	No	No	Yes
Iterative Sequence	No	No	Yes	No	No	No	Yes
Iterative Risk Management	No	No	No	No	No	Yes	Yes
Understandability	Simple	Intermediate	Intermediate	Intermediate	Much Complex	Hard	Intermediate
Requirement Definition	Yes	No	No	No	No	No	Yes
Quick Design Clarity	No	No	No	No	No	No	Yes
Resource Organization	Yes	Yes	Yes	Yes	No	No	Yes

5. Strengths and Limitations of HV Process Model

From Table 7, it is predicted that every software development process model has some strengths and limitations; therefore, the following are the strengths of the HV process model:

- Applicable for medium and long term software project size;
- Suitable for software projects of duration for more than one year;
- Covers maximum of risk items and management during development of software projects;
- Provides excellent customer satisfaction;
- Resources are well organized as it is applicable for software projects having more than one year duration;
- Iterative risk items identification and management;
- Requirements are not frozen during any phase of software engineering;
- Flexibility of updating the requirements, design and accordingly coding of software project as per the client's need.

The followings are the limitations of the HV process model:

- Not suitable for the software project for less than one year duration;
- Heavy resources are required for development of software products;
- Software products are developed in a series of increments of the client's need;
- Requires hand-on experience for development of long term software projects;
- Requires high experience of software programmers.

6. Concluding Remarks

The presented work is unique process model which may be used by the software industries for minimization of the risk factors involved during the development of the software. From the comparisons with the existing software development process models, it is found that this will produce high-quality software products. The risk factors may vary from small to large software products while the pre-

sented approach will take care of all kinds of software products. In the future, the HV process model may be implemented by researchers over the various kinds of software products to be developed by the software industries.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Royce, W.W. (1970) Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON*, **26**, 328-388.

[2] Grimm, T. (1998) The Human Condition: A Justification for Rapid Prototyping. *Time Compression Technologies*, **3**.

[3] Larman, C. and Basili, V.R. (2003) Iterative and Incremental Developments. A Brief History. *Computer*, **36**, 47-56. https://doi.org/10.1109/MC.2003.1204375

[4] James, M. (1991) Rapid Application Development. Macmillan.

[5] Aoyama, M. (1997) Agile Software Process Model. *Proceedings Twenty-First Annual International Computer Software and Applications Conference* (*COMPSAC'97*), Washington, 13-15 August 1997, 454-459. https://doi.org/10.1109/CMPSAC.1997.625042

[6] Boehm, B. (1986) A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, **11**, 14-24. https://doi.org/10.1145/12944.12948