Vesiputousmalli

Ryhmä: INTIM23A6

Tekijä: Timo Poutiainen

Vaatimukset

Valitsemani ohjelma oli palkanlaskentaohjelma, joka sisälsi palkkalaskelmat, sivukustannukset, asiakaslisäosan C#-ohjelmakielellä Visual Studionissa. Asiakkaat voivat muuttaa tietojaan ja nähdä, mitä heidän palkkansa sisältää, kuten nettopalkan, ennakonpidätyksen, työeläkemaksun ja työttömyysvakuutusmaksun. Lisäksi ohjelma kertoo sinulle tulevat palkkapäivät ja niihin liittyvät palkka- ja satunnaiset kulut seuraavan 30 päivän aikana. Päätin tehdä konsoliohjelman. Mukana on vaatimusmäärittely ja UML-kuvaajat.

Desing

Sovellus on konsolisovellus joten valikot toimii kirjoittamalla pyydetty syöte. Tarkoitus tehdä palkanlaskentaohjelmaan aluksi kolme valikkoa: 1. Työntekijöiden lisäys ja tietojen muutto, 2. Palkka- ja sivukulutiedot, 3. Tulevat palkkapäivät. Ensimmäinen valikko johtaa uuteen valikkoon missä voi lisätä henkilön ja hänen palkkansa ja paluuvalikon päävalikkoon. Toisessa valikossa näkee kaikki palkka- ja sivukulutiedot kuten ennakkopidätys, työeläkemaksu, työttömyysvakuutusmaksu sekä työntantajanmaksut kuten sairausvakuutusmaksu. Kolmannessa valikossa näkyisi tulevat palkkapäivät ja palkka- ja sivukulut 30 päivän ajalta.

Kehitys

Ensimmäisen valikon koodaus helpohko, vaikeuksia tuli lisätä valikon sisälle valikko (työntekijöiden palkkojen lisäys) niin että koodi ei bugaa. Jälkimmäisessä valikossa valitsi niin että on helppo lisätä uusia henkilöitä ja heidän palkkojaan menuna. Piti valita näytetäänkö palkkatiedot siellä erillisenä valikkona vai ainoastaan päävalikossa.

Development phase include implementation involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.

Testaus

Valikkojen toimivuus ja testaus oli ensimmäinen vaihe testauksessa. Piti katsoa että päävalikko ja sen jälkeiset valikot toimivat ja pystyy palamaan takaisin päävalikkoonkin. Henkilöiden listauksessa muistaa kirjatut henkilöt ja heidän palkkansa ja tämän tiedon voi avata toisesta valikosta.

In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.

Käyttöönotto

Käyttöönottaessa ohjelma toimi hyvin eikä oleellisia ongelmia löytynyt. Sovellusta jatkokehittäessä olisi voinut tehdä Visual studionin windows valikkoineen ja visuaalisineen menuineen, esimerkiksi päävalikot olisi voinut olla klikkattavia valikoita.

The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

**Features of the SDLC Waterfall Model**

1. **Sequential Approach**: The waterfall model involves a sequential approach to software development, where each phase of the project is completed before moving on to the next one.
2. **Document-Driven:**The waterfall model relies heavily on documentation to ensure that the project is well-defined and the project team is working towards a clear set of goals.
3. **Quality Control:** The waterfall model places a high emphasis on quality control and testing at each phase of the project, to ensure that the final product meets the requirements and expectations of the stakeholders.
4. **Rigorous Planning**: The waterfall model involves a rigorous planning process, where the project scope, timelines, and deliverables are carefully defined and monitored throughout the project lifecycle.

The Waterfall Model has six phases which are:

**1. Requirements:** The first phase involves gathering requirements from stakeholders and analyzing them to understand the scope and objectives of the project.

**2. Design:** Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.

**3. Development:** The Development phase include implementation involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.

**4. Testing:** In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.

**5. Deployment:** Once the software has been tested and approved, it is deployed to the production environment.

**6. Maintenance:** The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

Let us now learn about each of these phases in detail which include further phases.

### 1. Feasibility Study:

The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software.   
The feasibility study involves understanding the problem and then determining the various possible strategies to solve the problem. These different identified solutions are analyzed based on their benefits and drawbacks, The best solution is chosen and all the other phases are carried out as per this solution strategy.

### 2. [**Requirements Analysis**](https://www.geeksforgeeks.org/activities-involved-in-software-requirement-analysis/) and Specification:

The requirement analysis and specification phase aims to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.

* **Requirement gathering and analysis:** Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness (an incomplete requirement is one in which some parts of the actual requirements have been omitted) and inconsistencies (an inconsistent requirement is one in which some part of the requirement contradicts some other part).
* **Requirement specification:** These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between the development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.

### 3. Design:

The goal of this phase is to convert the requirements acquired in the SRS into a format that can be coded in a programming language. It includes high-level and detailed design as well as the overall software architecture. A [Software Design Document](https://www.geeksforgeeks.org/design-documentation-in-software-engineering/)is used to document all of this effort (SDD).

### **4. Coding and Unit Testing**:

In the coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The unit testing phase aims to check whether each module is working properly or not.

### 5. Integration and System testing:

Integration of different modules is undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over several steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.   
System testing consists of three different kinds of testing activities as described below.

* **Alpha testing:** Alpha testing is the system testing performed by the development team.
* **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.
* **Acceptance testing:** After the software has been delivered, the customer performs acceptance testing to determine whether to accept the delivered software or reject it.

### 6. Maintenance:

Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is 60% of the total effort spent to develop a full software. There are three types of maintenance.

* **Corrective Maintenance:** This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
* **Perfective Maintenance:** This type of maintenance is carried out to enhance the functionalities of the system based on the customer’s request.
* **Adaptive Maintenance:** Adaptive maintenance is usually required for porting the software to work in a new environment such as working on a new computer platform or with a new operating system.
* **Properly Documented:** Processes, actions, and results are very well documented.

<https://www.actitime.com/project-management/what-is-waterfall-model>

Valikkokohtien tekeminen oli melko helppoa, mutta minun piti viettää aikaa miettien, kuinka rakentaa sovelluksen runko ja koota joitakin valikon kohtia yhdessä. Tein kolme valikkokohtaa: työntekijöiden lisäys ja palkkatietojen muutos, palkka- ja satunnaiset menotiedot sekä tulevat palkkapäivät. Ensimmäisten valikkokohtien sisällä olevat valikkokohdat olivat hieman vaikeampia, mutta koodin sisennykset auttoivat. Jos valikon kohdat ovat yksinkertaisia eivätkä monimutkaisia, asiakas hyötyy siitä. Työntekijöiden ja heidän tietojensa lisääminen ja poistaminen on hieman haastavampaa. Ratkaisu oli tehdä niistä luettelo. Se, mistä palkka koostuu, oli vaativampaa, kuten ennakonpidätyksen, työeläkemaksun, työttömyysvakuutusmaksun, sairausvakuutusmaksun palkan ja nettopalkan valinta, joten käytin kolmansien osapuolien nettisivuja ja loin kuvitteellisia prosenttilukuja ja tein niiden perusteella matemaattisia kaavoja. koodissa asiakkaiden palkkatietoihin. Tulevien palkkatietojen koodaus oli tätä helpompaa. Vaikein koodausongelma oli saada JSON-tiedosto toimimaan ja muistamaan tiedot. Verkkofoorumit auttavat ja edellinen Ohjelmoinnin perusteet -kurssi antoi tähän opastusta.