|  |  |  |
| --- | --- | --- |
| קבוצה 20 | | |
| תאריך הגשה: **12/05/2020** | |  |
| שם מלא | ת"ז | מייל |
| איילון בן סימון | 312162951 | aylon8@gmail.com |
| סער ויקטור | 312392822 | saarms5@gmail.com |
| שובל דוד | 305284945 | shovaldavid877@gmail.com |
| שירז קרוב | 313599987 | shirazkarov@gmail.com |
| גל טולדנו | 205886559 | gal1010100@gmail.com |

שאלות:

1 .תארו את תהליך הניתוח שביצעתם למרכיב: "הפקת דירוגים" באמצעות המערכת האנליטית. פרטו

מה הם השאלות/פרטים שהתייחסתם אליהם. בתשובה התייחסו לקשרים ולמעברים 1 .ממודל -Use

case למודל תהליכי המיוצג בעזרת Diagram Activity ,2 .ממודל תהליכים למימושו בתוכנה.

2 .בהרצאה הוגדרה Reusability כתכונה של תוצר של תהליך הפיתוח אשר משקפת את היכולת

לבצע reuse בהקשר לתוצר זה. בהתאם להגדרה זו, תארו יישום של 3 התכונות המאפשרות לכם

לשלב במערכת MyFuel שאתם מפתחים קטעי קוד ומרכיבים אחרים שלא אתם כתבתם או תכננתם.

תארו בדיוק )ובהתייחסות ספציפית( ובפירוט את התכונות המאפשרות Reuse של אותם מרכיבים

אשר בחרתם לשלב במערכת שלכם, תוך התייחסות בדוגמאות ספציפיות )לא 'עקרוניות' או

'כלליות'( לדרישות הפונקציונליות של המערכת שתכננתם )התייחסות ספציפית בהקשר זה =

התייחסות למרכיבים ספציפיים מתוך התיאור המילולי הראשוני של פעולת המערכת ששאתם

מפתחים מהתחלת הסמסטר. לא כולל תהליך זיהוי משתמש(.

אם יש מי מ-3 התכונות הנ"ל אשר לא באה לידי ביטוי ב-reuse שביצעתם - הסבירו את הסיבה

לכך.

3 .א. הערכה כללית:

1 .מהם היתרונות של מודל UML כעזר לתהליך התכנון?

) i)הסבירו איך מתקבלים )מתממשים( היתרונות שציינתם.

) ii)ציינו דוגמה אחת קונקרטית )לא כללית ולא Login )מתוך תהליך הניתוח והתכן

שאתם בצעתם לשימוש מועיל ב-UML תוך תיאור והתייחסות ספציפית למרכיבים של

מערכת "MyFuel "שתכננתם ומידלתם.

2 .ציינו קשיים הנובעים מחסרונות של UML שנתקלתם בהם. גם כאן התייחסו ספציפית

לתהליך שבצעתם לפיתוח מערכת זו.

ב. ניתוח ודיון:

בהתאם לניסיון שרכשתם במהלך העבודה על מטלה זו, תארו אפשרויות לשינויים ושיפורים

במתודולוגית UML אשר נותנים מענה לחסרונות שנתקלתם בהם במהלך ה-design שביצעתם

בפרויקט שלכם. הסבירו את תשובתכם תוך תיאור דוגמה ספציפית )כולל שמות של רכיבים, לא

כולל Login )מתוך עבודתכם.

תשובות:

1. תהליך הניתוח שבצענו למרכיב: "הפקת דירוגים" באמצעות המערכת האנליטית מנה מספר שאלות שהיינו צריכים לענות עליהם:

* איזה בעל תפקיד מבצע את הפעולה של הפקת הדירוגים?
* האם יש בעלי תפקידים נוספים שקשורים בעקיפין לאותה פעולה?
* מהו רצף האירועים/פעולות שיש לבצע עד לרגע "הפקת הדירוגים?
* אילו תנאים מקדימים נדרש לבצע עד לרגע בו נוכל להפיק דירוגים?
* האם נדרש שיהיה מסך ייעודי שבו ניתן לראות את הדירוגים במערכת?

בעזרת ה- use case diagram ניתן לראות איזה בעלי תפקידים קשורים לתהליך מסוים וכך נדע איזה הרשאות לתת לכל בעל תפקיד במערכת ואילו באמצעות ה- activity diagram ניתן לראות רצף של אירועים, כלומר אילו מתודות מתרחשות ובאיזה סדר. במקרה שלנו ("הפקת דירוגים") ניתן לומר על פי ה-use case diagram שהשחקן העיקרי הוא המערכת עצמה כלומר זה תהליך פנימי שמתבצע בתוך המערכת (ע"י המערכת האנליטית) שלא דורש שבעל תפקיד מסוים ייזום את התהליך, אלא קורה אוטומטית. כמו כן, ניתן להבין שישנו בעל תפקיד נוסף שקשור לתהליך אך הוא אינו משפיע עליו, הוא (עובד מחלקת השיווק) יכול לצפות בדירוגים ולא להשפיע עליהם בשום אופן לצורך ניהול מבצעי קידום מכירות אך לא ניתן לראות כיצד הנ"ל מתבצע ואילו מתודות לוקחת חלק בתהליך הנ"ל. לעומת זאת, ע"י ה-activity diagram ניתן לראות באופן ברור את ה-flow של הפעולות השונות שנדרש לבצע על מנת שנציג השיווק יוכל להפיק דירוגים כלומר נדע מהם התנאים המקדימים עבור התהליך ומהו רצף הפעולות שצריכות להתבצע על מנת שהתהליך יושלם. במעבר ממודל תהליכים למימושו בתכנה נוכל לראות איזה רכיבים קשורים לביצוע התהליך בשלמותו. נדע איזה מתודות יצטרכו להתבצע ובאיזה סדר. כמו כן, נוכל לקבוע איזה פרטים הנמצאים ב- database התהליך הזה דורש ואיזה פרטים ישתנו כתוצאה מביצוע התהליך.

1. שימוש ב-Reuse מסייע למפתחי תוכנה להיעזר בקוד קיים ללא צורך במימוש הקוד עם אותה פונקציונאליות. נוכל לבצע שימוש חוזר בקטע קוד אשר מקיים את שלושת התכונות הבאות: זמינות גמישות והבנתיות. לדוגמא בפרויקט שלנו, במטלה הנוכחית השתמשנו ב-Reuse עבור מנגנון ה- OCSF: השתמשנו במחלקה OCSF כדי לא לממש בעצמנו עיצוב של שרת לקוח (Client-Server).

* זמינות: משתמשי שרתי לקוח יכולים להיכנס ישירות למערכת למרות המיקום או הטכנולוגיה של המעבדים.
* גמישות: הקוד ה"מושתל" גמיש בהיבט שיש לנו היכולת לשנות קטעי קוד מסוימים בו בקלות ולהוסיף קטעי קוד על פי משתמשים שונים ודרישות מערכת שונות.
* הבנתיות: הקוד ש"הושתל" באב טיפוס הוא קוד מובן ובעל תיעוד מיטבי אשר מסביר את מהות הפונקציונאליות, אשר עברנו עליו ווידאנו שהוא עובד בצורה הרצויה עבור האב טיפוס שלנו.

ברגע שהוספנו את מחלקת ה OCSF, יצרנו 2 ממשקים היכולים לתקשר אחד עם השני ולהעביר בין אחד לשני מידע, למשל אובייקטים, מסרים ועוד. באופן ספציפי, בעת מימוש האב טיפוס, בפועל יצרנו חבילה של שרת וחבילה של לקוח. חבילת השרת יכולה לתקשר עם חבילת הלקוח ולקבל ממנו מידע רצוי. כמו כן, ניתן לבצע עדכונים במידע הנמצא בחבילת השרת (כלומר לבצע שינויים במסד הנתונים) ורצף האירועים נראה כך: לקוח מבקש מידע מהשרת 🡨 שרת מתחבר למסד הנתונים 🡨 מבצע שאילתא התואמת את בקשת הלקוח 🡨 החזרת תוכן השאילתא ללקוח.

3. א. 1.

יתרונותיו של מודל ה UML כעזר לתהליך התכנון הם:

* ניתן להציג בצורה גרפית את המחלקות השונות והקשרים ביניהן דבר שעשוי לעזור הן ל"לקוח" והן ל"חברה" שנדרשת לממש את התוכנה המיועדת.
* באמצעות יצירה של דיאגרמות ברורות וקלות להבנה הנ"ל יוכל לשמש בעת הצגה והסבר של סקיצת הפרויקט, כאשר ייצוג ע"י דיאגרמות ולא ע"י כתיבה יחסוך ללקוח את הצורך בלקרוא כמויות רבות של מלל (תמונה אחת שווה אלף מילים).
* מודל ה-UML מאפשר את האפשרות לצפות את תהליך הפיתוח ולתכנן בהתאם את מימוש הקוד, דבר שיקצר/יקל ויסייע בזמן הפיתוח אשר כידוע דורש זמן רב.
* על ידי הדיאגרמות Activity ו- Sequence ניתן לראות את רצף האירועים השונים שמתרחשים במערכת, כלומר, אילו מתודות מתרחשות ובאיזה סדר ולאחר מכן הדיאגרמות הנ"ל עשויות לסייע במהלך מימוש הקוד.

למשל, במערכת שלנו, בעת שלב ביצוע ה- class diagram שמנו לב שאין צורך לעשות ישות של עובד ועוד הרבה ישויות שונות (בעלי תפקידים שונים במערכת) שיירשו ממנה, כי בעצם אין מידע שנדרש לשמור על כל עובד למעט תיאור תפקידו. בפועל, נרשום מחלקה אחת של עובד ששם ירוכזו כל המתודות (הרשאות) של כל העובדים, מה שיפשט את מבנה התכנה ויקצר לנו את זמן הפיתוח. בנוסף ב-DB במקום טבלה עם מידע עבור כל עובד בנפרד תהיה לנו טבלה אחת שתשמור את כל המידע עבור כל העובדים ובכך נחסוך תחזוקה של טבלאות מיותרות.

2.

אחד הקשיים הבולטים שעולים משימוש במודל ה-UML הוא שאין אפשרות למדל את ה-GUI, ועל כן עדיין אין לנו מספיק מידע לגבי כיצד המערכת תוצג למשתמשים, לדוגמא כיצד יראה מסך רישום לקוח ע"י נציג שיווק. קושי נוסף שעולה משימוש במודל ה-UML הוא הזמן שלוקח לנהל ולתחזק דיאגרמות UML. כדי לעבוד כראוי, יש לסנכרן את דיאגרמות UML עם קוד התוכנה, הדורש זמן להתקנה ותחזוקה, ומוסיף עבודה לפרויקט פיתוח התוכנה.

3. ב.

מודל ה- UML עזר להבנה הכללית של הפרויקט, הדרישות ואופי המערכת ככלל. לדוגמא, זה עזר משמעותית להבנה של תהליך התדלוק ואיך הוא אמור להתבצע ומהם השלבים הנדרשים לביצועו. כמו כן, כמו שהוזכר בסעיף קודם, מודל זה לא נותן מענה על כל הצרכים הנדרשים מהמערכת שלנו, למשל נושא ה – GUI. למשל, עבור מסך של רישום לקוח חדש, נציג השיווק צריך לקבל שגיאה אם יש שדות רלוונטיים שנשארו ריקים, מה שלא ניתן לראות אם לא עובדים בסנכרון גם על מסכי ה – GUI שצריכים להיות במערכת שלנו. בנוסף על מנת לנהל ולתחזק את דיאגרמות ה-UML בצורה יעילה ומועילה יש לעבוד ע"פ מתודולוגית "מפל המים" שבאמצעותה הדיאגרמות השונות ייבנו בצורה היררכית והדרגתית שלב אחרי שלב, דבר אשר יקל את ביצוע שלב ה-design. לדוגמה, על מנת לממש Sequence diagram עלינו לממש תחילה גרסה ראשונית של Class diagram , מתוך ה- Sequence diagram עולות לא מעט שאלות אשר באות לידי ביטוי גם בדיאגרמות שהתבצעו לפני, כמו גם מתודות אותם יש צורך להוסיף ל- Class diagram.