שיטות הנדסיות לפיתוח מע' תכנה, תרגיל 1:

קבוצה: 13

חברי הקבוצה: נתיב לנגל nativlengel11@gmail.com

נדב מאיר nadav491@gmail.com

תומר אבירם tomerav83@gmail.com

שאלה 1:

השתמשנו במודל ה use case בפיתוח המערכת AES שלנו, אנו חושבים כי היתרון העיקרי במודל זה הוא שהוא ידידותי, ולא עמוס, מאוד פשטני אך אם זאת, מצליח לתת את כמעת כל המידע הנדרש על המערכת, ובאופן כללי יעיל מאוד ורלוונטי כהיכרות ראשונה ותחילית עם נושא ה classes שאנו עתידים לעבוד בו בהמשך הפרויקט.

באופן כללי – המודל תורם להבנה ולתפיסה קצת יותר רלוונטית מבחינת התכנה – ישנם שחקנים שאפילו יכולים להיות זרים זה לזה, המבצעים פעולות המשפיעות באופן ישיר ובנוסף באופן עקיף על פעולות של שחקנים אחרים, והדבר היחידי המקשר ביניהם הוא המערכת.

היכולת לפרק ולהבדיל את סוגן של פעולות שונות, שהן חייבות להיעשות ומהוות חלק בלתי נפרד מהפעולה הראשית (include) , ואת אלו שהן אופציונליות (extend), בנוסף האילוצים שניתן להוסיף כדוגמת XOR על הדיאגרמה כמעיין הסבר בין שנים / יותר use case-ים, המציין באופן פשוט ומפורש כי רק אד מהמצבים יקרה, דבר המונע הבנה שונה של המערכת.

כדוגמאות ספציפיות מתוך מערכת AES –

1. Use case – "get statistical information" – המקשר בין שחקנים זרים – תלמיד המבצע מבחן, מורה שתלמידיו בצעו מבחנים, והתלמידים המבצעים את המבחנים, כל הנתונים הסטטיסטים נשמרים במערכת בשל כך שהשחקנים בצעו פעולות שהשפיעו על המערכת באופן עקיף.
2. Use case – "starting test" - המכיל בתוכו תתי פעולות שהן חובה (include), כדוגמת בחירת סוג המבחן, וכת רק פעולה המשכית אחת תבוצע בהתאם לבחירת סוג המבחן, וזאת בזכות אילוץ ה XOR.
3. Use case - "modify question" - שהוא בעצם פעולה חיונית, אך אופציונלית (extend) שהמורה יכול לבצע.

שאלה 2:

נתקלנו במעט קשיים, שאותן לא יכולנו להראות באמצעות מודל ה use case.

המרכזי שבהם הוא בעצם כל הדרכים בהם שחקן אחד תלוי באחר, ומחכה לתגובה ממנו, באופן כללי – הקשרים נראים מעט מסורבלים, ואפילו לא מובנים, מי השחקן שמחכה למי, מי עושה את הפעולה..

הקושי השני הוא בעצם כל הפעולות אותם צריכה המערכת לבצעה.

דוגמאות מפורטות מתוך מערכת AES -

1. דרישות פונקציונליות מבחינת תקשורת בין שחקנים-

Use case – "Change time limit – המקשר בין שחקנים זרים – מורה ומנהל, באופן כזה שבאם ירצה מורה לשנות זמן בחינה מסוימת, יצטרך לבקש את אישור המנהל לכך, דוגמה למראה קצת מבלבל בין שחקנים ו use case.

1. דרישות פונקציונליות מבחינת ביצועי מערכת –
   1. "המערכת תשמור מידע סטטיסטי"
   2. "המערכת לא תציג נתונים סטטיסטים לתלמיד"
   3. "המערכת תציג נתונים סטטיסטים למנהל באופן שונה, ועליה להיות גמישה"
   4. "המערכת מבצעת בדיקה אוטומטית של העתקה"
   5. "המערכת מבצעת בדיקה ומתן ציון למבחן"
   6. "המערכת מפיקה דוחות למנהל"
   7. "המערכת צריכה להיות גמישה ולאפשר שינויים עתידיים"

כל הדרישות הפונקציונליות האלו, לא כללות כלל ב use case diagram, אלא בדיאגרמת הדרישות, וללא דיאגרמת הדרישות, לא נוכל כלל לבטא את כל הדרישות הפונקציונליות של המערכת, דבר שעלול לגרור מעיין התעלמות מכל החלקים האלו.

שאלה 3:

הבעיות אותן תיארנו הן –

1. תקשורת בין שחקנים
2. דרישות פונקציונליות של ביצועי מערכת

הפתרונות אותם אנו מציעים –

1. הוספת אילוץ "await response" במובן בו נוכל לתאר את התקשורת בין השחקנים השונים, באופן מובן וברור יותר, כאשר למשל לאילוץ זה יהיה חץ שיראה בדיוק איזה שחקן מחכה לתגובת השחק האחר, ואיזה use case הוא הרלוונטי.
2. להוסיף מעיין "internal use-cases" שיאפיינו את כל דרישות המערכת הפונקציונליות, ואותם נוכל לקשר ל use cases הקיימים של השחקנים השונים, באמצעות הוספת אילוץ "sys performance" היקשר בין ה use case הקיים ל internal use cases.

תרומת הגישות כמענה לבעיות –

ראשית – אנו בעצם מצמצמים מעברים ויצירה של דיאגרמת נוספות לכדי דיאגרמה אחת, אמנם קצת גדולה יותר, אך ברורה בהרבה ופחות מסורבלת, הן מבחינת הבנת המתכנת/מפתח והן מבחינת מעצבי המערכת.

הדבר גורר באופן ישיר – הפחתה משמעותית של אי הבנות או "הבנה מרובה" של מרכיבי המערכת השונים – **דבר קריטי בעבודת צוות בפרויקט.**

בנוסף, נוכל לפשט ביטויים וצורות הצגה של דברים באופן קל יותר וברור יותר על ידי הוספת "אילוצי תקשורת" כשהדבר חייב להיות מתואר ובלתי נמנע במערכת.

דוגמאות –

1. Use case – "Change time limit – נוכל לתאר את הפעולה הזו באופן יותר ידידותי להבנה, מבלי להיכנס לפרטי ה use case, ולמנוע הבנתו מרובות של אנשי הצוות השונים.
2. "המערכת תשמור מידע סטטיסטי" – נוכל להוסיף את ה internal use-case הזה ל use case של השחקן "מנהל" ובכך נוכל לבטא גם את פעולות המערכת, ובאופן דומה גם לכל שאר הדרישות הפונקציונליות של ביצועי המערכת ותמיכת המערכת שציינו –

"המערכת לא תציג נתונים סטטיסטים לתלמיד"

"המערכת תציג נתונים סטטיסטים למנהל באופן שונה, ועליה להיות גמישה"

"המערכת מבצעת בדיקה אוטומטית של העתקה"

"המערכת מבצעת בדיקה ומתן ציון למבחן"

"המערכת מפיקה דוחות למנהל"

"המערכת צריכה להיות גמישה ולאפשר שינויים עתידיים"

ובאופן המתואר, נוכל לייעל, לשפר ולענות על הבעיות שציינו