נכתב ע"י: אביה צ'ריקר :)

הנדסת תוכנה שיעור 4

תיכון:

שאלת השיעור: איך? - ארכיטקטורה.

הנדרש: נצטרך להבין יותר איך ליישם את הדרישות בפועל.

הפלט: מסמך מפרט תיכון SDS

נתחיל בשאלות כלליות:

- מה זה תיכון/עיצוב?
  תוכנית עם יותר או פחות פרטים (לא מפרטים כמה היא תהיה מפורטת או לא). תוכנית שגורמת לנו לחשוב רגע לפני שכותבים את התוכנה עצמה.
- תיכון ואיכות:
  ככול שמזהים באגים מוקדם יותר – העלות של תיקונם קטנה יותר.

למה צריך את מסמך מפרט התיכון? SDS

נענה בכמה שלבים:

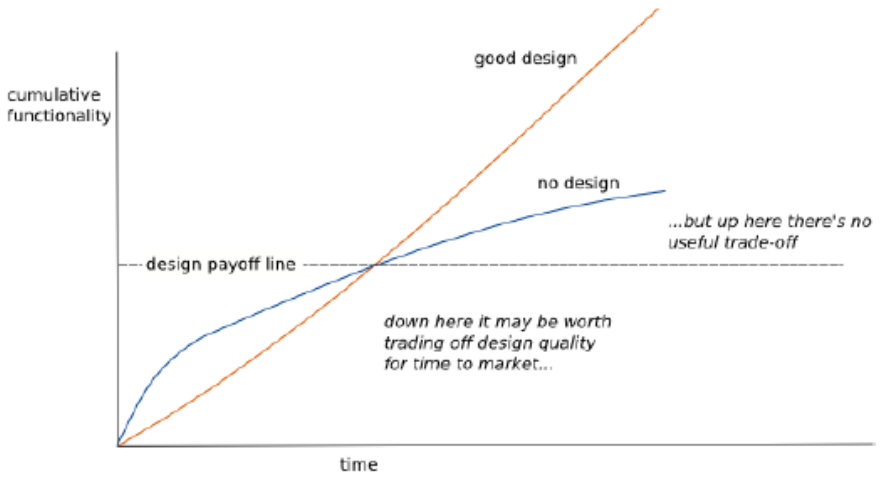
יש קונפליקט בין תכנון מראש כך שנהיה מוכנים להכל לבין העובדה שאף פעם לא נוכל לדעת באמת את כל מה שצפוי לקרות.

שאלה: איך עוברים מהבעיה/דרישות (הסיבה לשמה אנו צריכים לייצר את המוצר) לפתרון/קוד (המוצר עצמו)? איך מגשרים בין השניים?

תשובה: תיכון וארכיטקטורת תוכנה – נכתוב ונתאר את הממשקים ופעולות האפליקציה, זה יוצר בעצם מסגרת כללית לבניה ופיתוח התוכנה.

נכתב ע"י: אביה צ'ריקר :)

מה הדיאגרמה הבאה מראה?



הדיאגרמה מראה שיכול להיות שבהתחלה מסמך התיכון אינו יעיל וניתן להסתדר בלעדיו (ואפילו יותר טוב) אך בהמשך ניתן לראות שעבודה עם מסמך מפרט תיכון מייעלת בהרבה את העבודה כיוון שבלעדיו ייתכן שנעשה דברים מיותרים, ויתכן שהיו דברים שהייתי אמור לעשות ופספסתי.

ארכיטקטורה

ארכיטקטורה מתארת את המבנה העיקרי של המערכת כך שהיא תתאים לצורכי הלקוח תוך כדי עמידה באילוצי לקוח ותפקיד

מה היא כוללת בפנים?

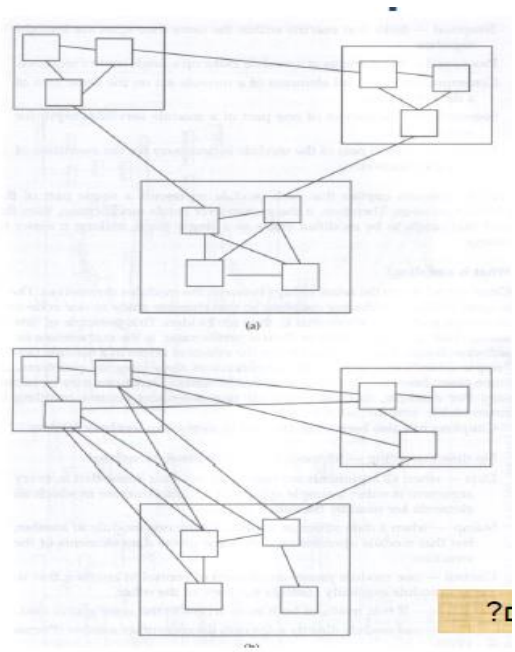
1. המרכיבים העיקריים וההתנהגות שלהם
2. הקשרים בין המרכיבים האלו

איך יודעים שהארכיטקטורה טובה ?

1. חלוקה לשכבות ברורות המייצגות כל אחת הפשטה של המערכת ומספקת ממשק ברור.
2. הפרדה בין הממשק והמימוש של כל שכבה.
3. פשטות: שימוש בהפשטות* ומנגנונים מקובלים.

נכתב ע"י: אביה צ'ריקר :)

מצגת 25, איזה תיכון יותר טוב?



התיכון הראשון.

הסיבה: אנו רוצים צימוד נמוך ולכידות גבוהה.

בין הרכיבים הגדולים יש מעט קשרים יחסית ז"א שהוא מקבל שירותים מאחרים בצורה פשוטה וברורה, (רמת הצמידות נמוכה) , אך בתוך הרכיבים עצמם רואים שכולם מחוברים אחד לשני זה אומר שכולם עובדים על אותה משימה (לכידות)

במה ארכיטקטורה עוזרת?

1. הבנת המערכת – תאור הקשרים בין מרכיבים
2. שימוש חוזר – לאור החלוקה הכללית לרכיבים, זיהוי הזדמנויות
3. טיפול בדרישות לא-פונקציונליות (אילוצים)
4. מענה לסיכונים
5. מימוש – חלוקה למשימות (במיוחד בצוותים גדולים) וכך נעבור מדרישות למימוש
6. ניהול – עוזרת להבין את כמות העבודה ולעקוב אחרי התקדמות
7. תקשורת – מייצרת הבנה ואוצר מילים, "תמונה אחת שווה אלף מילים"
8. אמורה לאפשר שינוי!

כשכותבים ארכיטקטורה כותבים את המערכת מכל מיני היבטים

1. מודל השכבות :
   הרבה פעמים כדאי לכתוב מערכת עפ"י השכבות
   לכל שכבה מותר לתת שירותים רק לשכבה מעליה והיא מקבלת שירותים רק מהשכבה מתחתיה.
2. השאר לא עברנו עליהם בשיעור.

נכתב ע"י: אביה צ'ריקר :)

מהו קוד טוב?

1. כל הבדיקות עוברות
2. אין כפילויות בקוד DRY (דונט רפיט יורסלפ)
3. ברור – מבטא את כוונת המתכנת
4. קטן – מינימום של מחלקות ומתודות

נשים לב שזה מסודר לפי החשיבות כאשר 1 חשוב יותר ו-4 הכי פחות חשוב

UML

שפה סטנדרטית שמקובלת בעולם ההנדסת תוכנה

האם צריך את זה או לא?

לפעמים הציורים של UML מסייע לנו להבין מה קורה במערכת.

Deployment Diagram - דיאגרמה של החומרות