Azrieli
College of Engineering
Jerusalem

עזריאלי
מכללה אקדמית להנדסה
ירושלים

Henry Ford: “If I'd talked to my customers, they would have asked for faster horses.”

Berard : “Walking on water and developing software from a specification are easy if both are frozen”

F. P. Brooks, "The Mythical Man-Month”: “The hardest single part of building a software system is deciding precisely what to build”

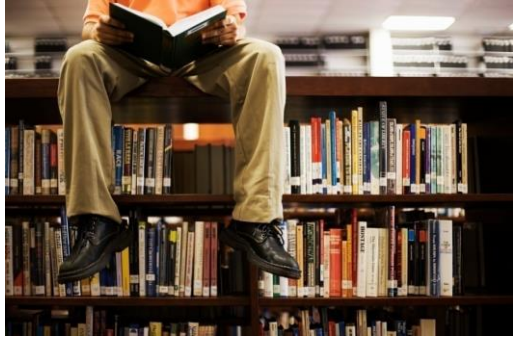
הנדסת תוכנה

3. דרישות

השבוע

- דרישות
  - משחק
  - מפרט דרישות תוכנה (SRS)
  - תרחישי שימוש
  - כלי מידול UML
  - סיפורי משתמש
  - מפרטים מורצים
- סדנת דרישות
- סקר אתחול פרויקט (שתי קבוצות בכיתה + השאר offline)
- פרויקט (מעבדה) – כתיבת מפרט דרישות (כולל אב טיפוס)
- בתרגיל
  - המשך בניית אב טיפוס – הגדרות ותבניות

מקורות



- Pressman, Requirements 5-6
- Cockburn Writing Effective Use Cases,
- Amber, Introduction to User Stories
- More:
  - Pragmatic Programmer, p. 202-208
  - Survival Guide, Ch. 8: Requirements Development
  - Adzic, Bridging the Communication Gap, Specification by Example and Agile Acceptance Testing
  - Online tutorial: http://www.cragsystems.co.uk/SFRWUC/index.htm
  - Spolsky, "Painless Functional Specifications"

איפה אנחנו בפרויקט (בקורס)?

- למה?
  בעיה (פלט: הצעת פרויקט\חזון\SOW)
- מי?
  צוות (Inception, אתחול\תכנון פרויקט)
- מה?
  דרישות (SRS)
- איך?
  תיכון (ארכיטקטורה) (SDS)
- מתי?
  תכנון וניהול – (ZFR)
- הלאה
  (איטרציות, Code)

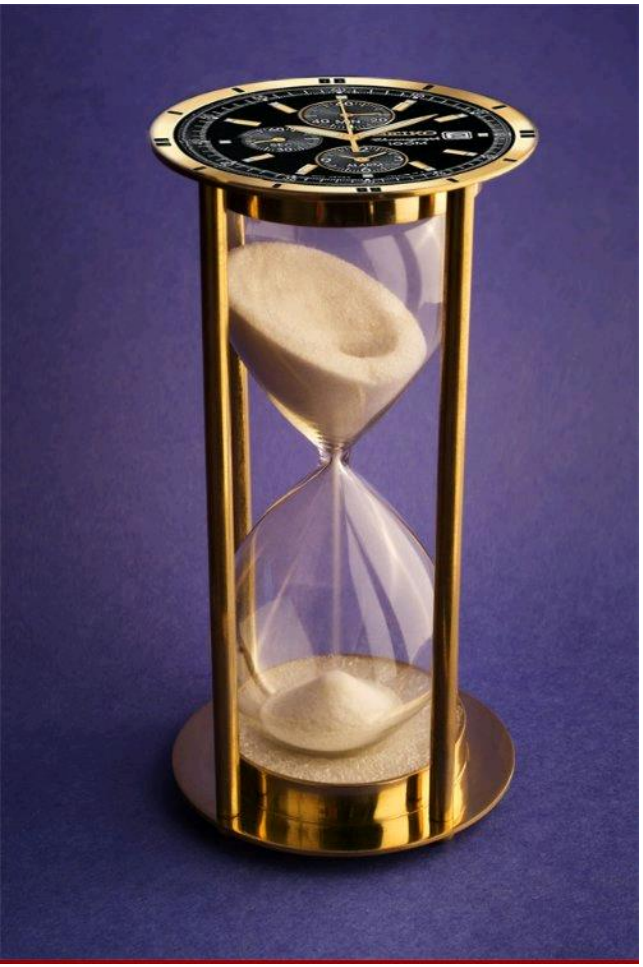
פרויקט – שלבים

- אבני דרך (עם תשלום), כל שבוע-שבועיים
  - Vision/SOW/SDP
  - SRS
  - SDS
  - גרסת 0 (ZFR)
  - סבבים \ ספרינטים
  - שחרור מתמשך \ גרסאות (למשל בטא)
  - שחרור סופי

Plan&Doc(sprint 0)

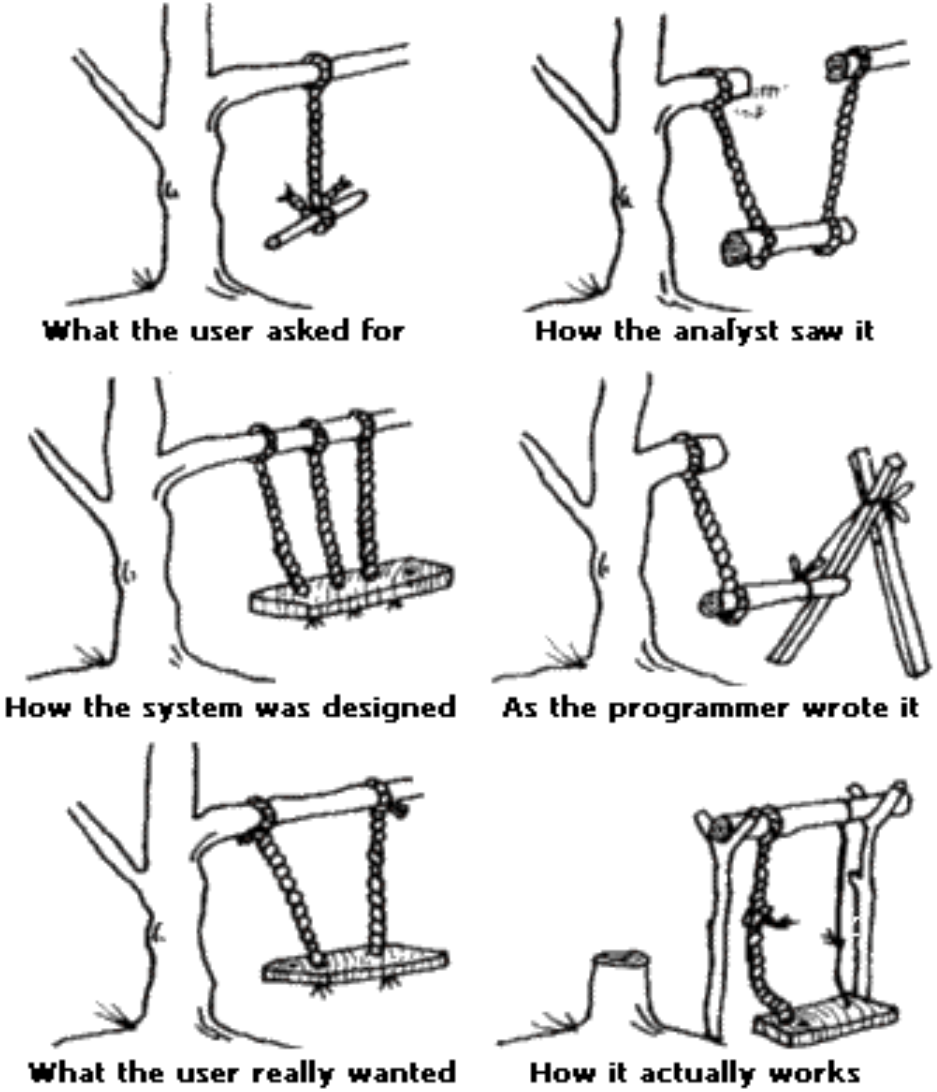
Agile / Iterative

"אז מה עושים עכשיו\בינתיים?"



- מחקר טכנולוגי ועסקי
  - נסיונות (spikes) להנמכת סיכונים (דיווח כחלק ממשימת מה-SRS)
- אב טיפוס (גם שיווקי)
- הכנת תשתיות פיתוח (במקביל למשימות הבאות)
- דרישות: איסוף וניתוח

Pragmatic Programmer Tip:
Don’t Gather Requirements - Dig for Them
Requirements rarely lie on the surface. They’re buried deep beneath layers of assumptions, misconceptions, and politics.



http://youtu.be/watch?v=OfgfnZZdMlI

דרישות

תהליך: פעילויות לדוגמא

- **דרישות**
  - **תת פעילויות, למשל: איסוף, ניתוח (Analysis)**
- **תיכון (Design)**
- **מימוש**
- בדיקות\אימות\שילוב
- תיעוד
- הטמעה\תמיכה\אחזקה

- פעילויות תומכות
  - ניהול הפיתוח
  - הבטחת איכות
  - סקרים
  - ניהול תצורה
- כלים

לכל פעילות בד"כ יש תוצרים

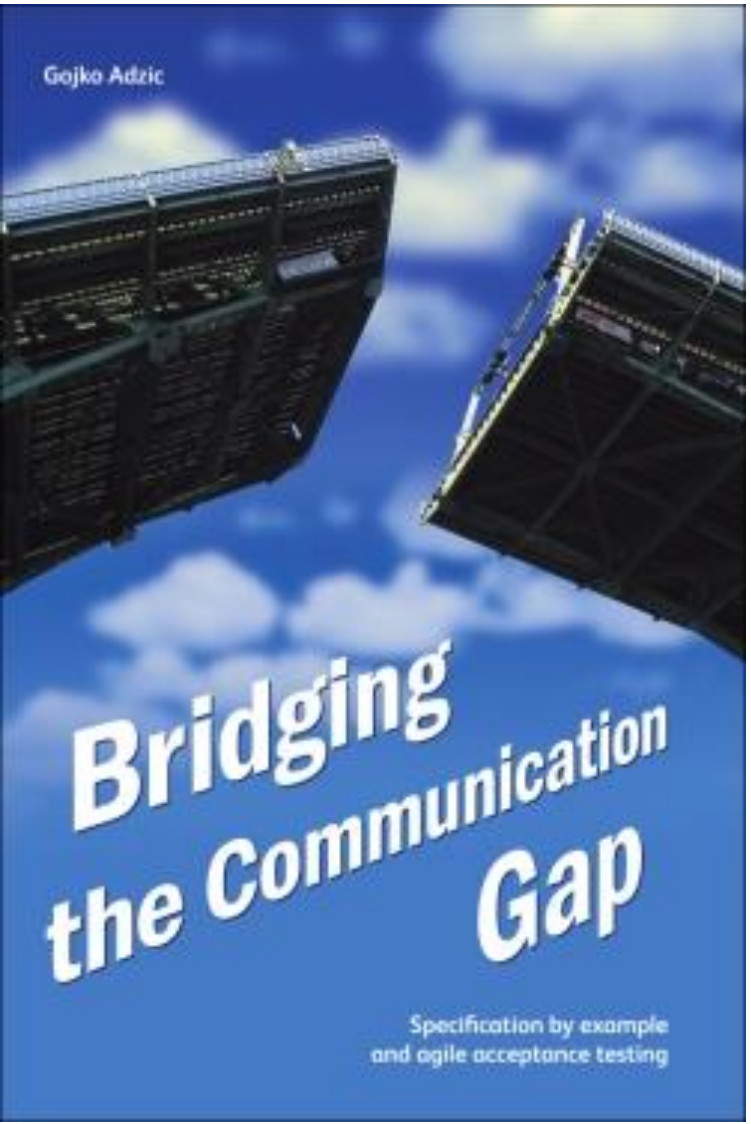
ראשי פרקים

- כמה דברים מעניינים על דרישות
- מהן דרישות?
- איך אוספים דרישות?
- איך מגדירים דרישות?
- איך בודקים דרישות?

סיפור משתמש

- תרגיל כתיבת סיפור (45 שניות)...
- הגיבור: את\אתה – איך הגעת לכאן הבוקר?
  - כל מה שעשית מאז שהתעוררת עד שהגעת לכיתה
- דיון

פער התקשורת



- צדדים מעורבים
  - לקוחות \משתמשים
  - מפתחים\מהנדסים
- בד"כ הלקוחות לא מבינים בתוכנה והמפתחים לא מבינים את הבעיה (העסקית)
- מה קורה אם אחד מהצדדים יותר דומיננטי?
- כיצד מעבירים את הצרכים והידע אל מי שאמור לפתח?

והפתרון... (ניסיון I)

מפרט דרישות תוכנה

Software Requirement Specification

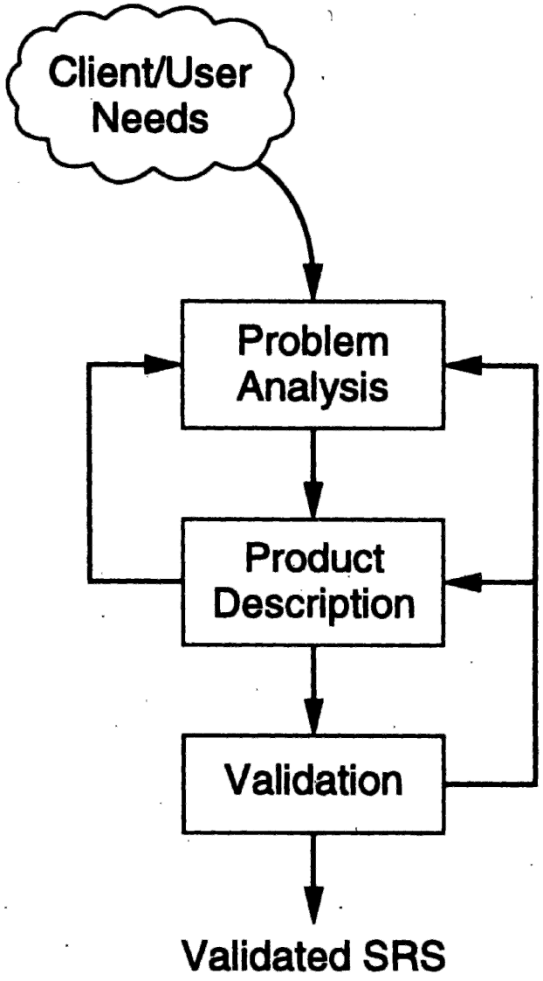
דרישות תוכנה

- דרישות מגדירות ומפרטות, מה לבנות
  - את ה"מה" ולא את ה"למה" (SOW\חזון\הצעה) וה"איך"
  - התרכזות בבעיה ולא בפתרון
- למה צריך דרישות?
-
  - להבין בדיוק מה נדרש מהתוכנה
  - לתקשר עם כל המעורבים לצורך הבנה מדויקת
  - לבקר את התהליך, כדי לוודא שהמערכת מממשת את המפרט (כולל שינויים)

Specification ? מפרט

- Lamport: a specification is a contract between user and implementer such that neither must talk to the other
- Brooks: clients do not and can not know their needs well enough to write such a contract
- Cockburn: A specification is notification that certain design decisions have been taken and the design space has been reduced. Each new specification should necessarily satisfy the previous; it may well happen that direct implementation is for many stages not practical.

תהליך הדרישות



1. איסוף
2. ניתוח
3. פירוט
4. אימות

שני סוגי דרישות עיקריים

- פונקציונליות \צרכים – השירות\ההתנהגות שהמערכת מספקת
- לא-פונקציונליות\אילוצי איכות – כל השאר
  - ביצועים (Atwood: Performance is a Feature)
  - אמינות
  - פרטיות, אבטחה
  - תעוד, קלות שימוש
  - קלות הרחבה
  - סביבת הפעלה, ממשקים
  - עלות, זמנים
  - תקנים, חוקים
  - (כלליו\איך\ ilities ...\לא בהכרח למימוש – אז מי דואג להם?)

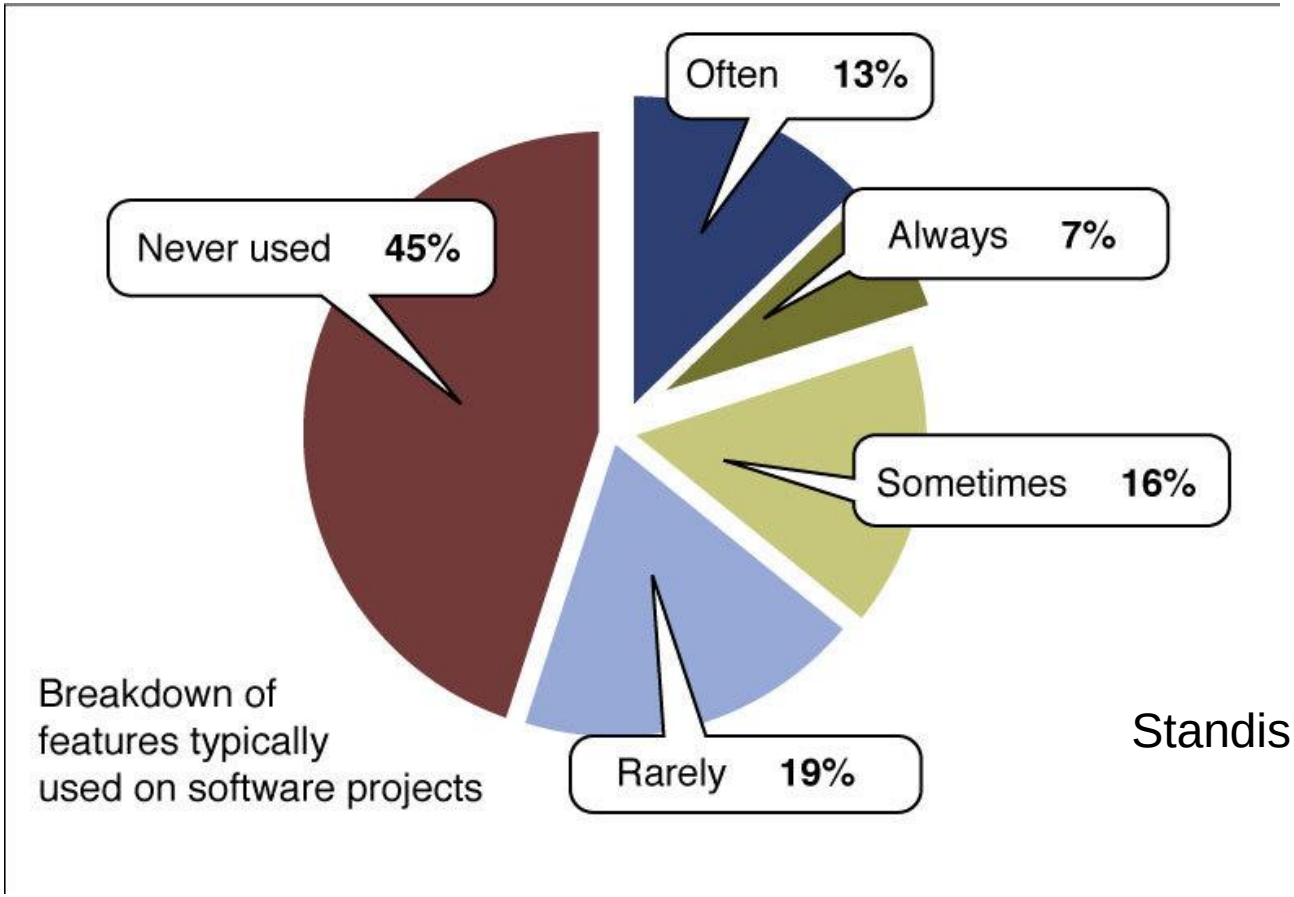
אבטחה ואג'ייל

- בפרויקט חדש ברשת הזקוק לאבטחה משמעותית הועלו מספר טענות כנגד פיתוח בשיטות אג'ייל, מהי הטענה החלשה ביותר לדעתך?

1. אבטחה היא דרישה לא-פונקציונלית ובלי מפרט דרישות ייתכן שהיא תתפספס
2. באג'ייל מתחשבים יותר מדי בלקוחות והם לא תמיד מבינים באבטחה
3. אבטחה מצריכה תכנון מראש ושיטתיות מה שחסר באג'ייל
4. אג'ייל שיטה חדשה מדי עבור פרויקט שמצריך אבטחה

הגדרה

- מילון: דבר נצרך עבור הקיום או ההתרחשות של דבר אחר
- האם?



Standish group 2002

דרישה?

- Kent Beck: “Software development has been steered wrong by the term ‘requirement,’ defined in the dictionary as something that is mandatory or obligatory. The word carries a connotation of absolutism and permanence, inhibitors for embracing change. And the word ‘requirement’ is just plain wrong.
- “Out of the thousands of pages used to describe requirements, if you deliver the right 5, 10, or 20 percent, you will likely realize all of the business benefit envisioned for the whole system. So what were the other 80 percent? Not requirements—they weren’t mandatory or obligatory.”
- שיטת MoSCoW – בפרויקט?
- אג'ייל: דרישה => מאפיין

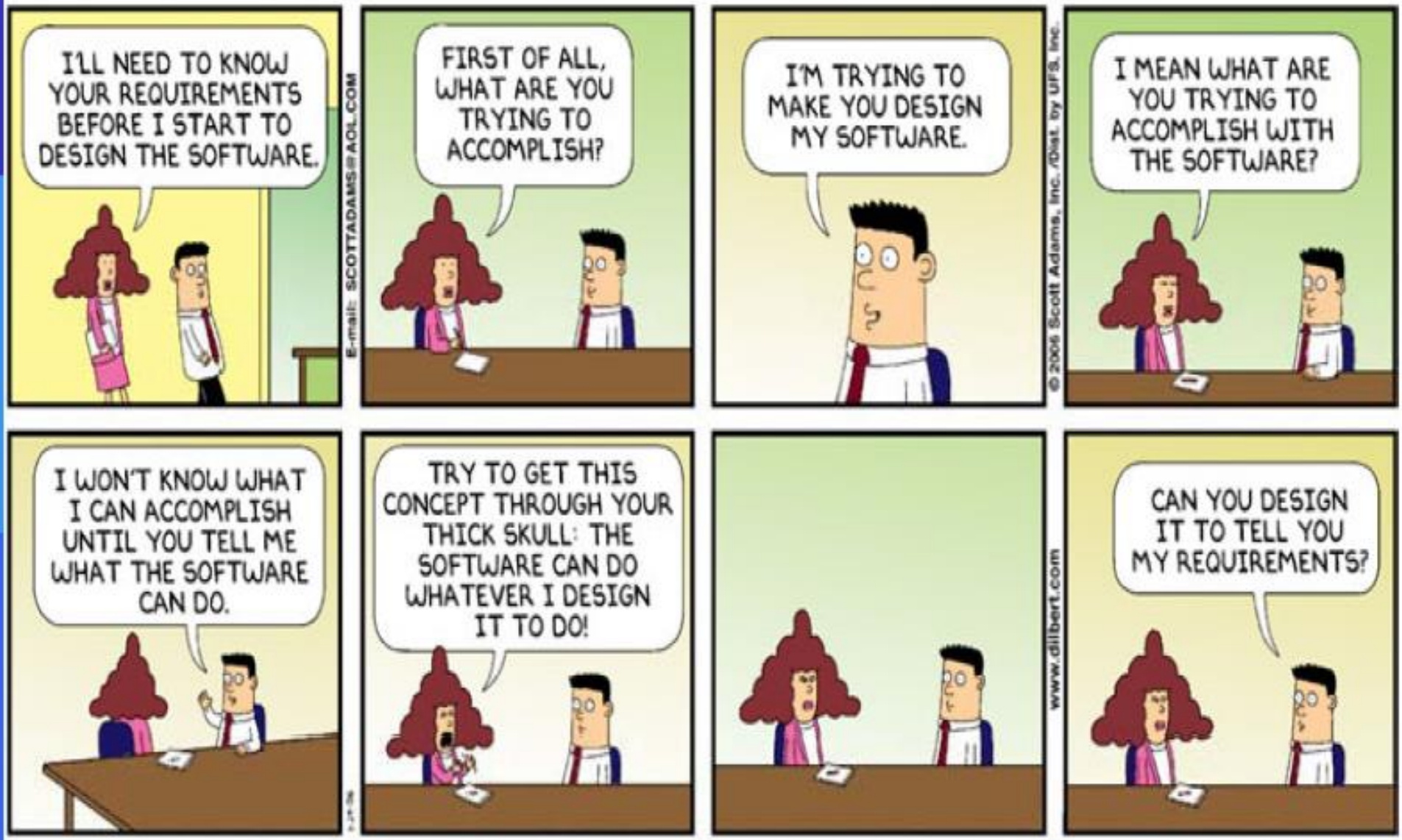
מהי דרישה טובה? (IEEE830)

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked
- Verifiable
- Modifiable
- Traceable

איך אוספים דרישות?

ראשית:

- מחקר "כאוס" של קבוצת Standish על יותר מ-8000 פרויקטים מצא שהסיבה העיקרית לכשלון פרויקט, היא חוסר מעורבות של המשתמש (סיבה שניה: חוסר ניהול).
- Facts & Fallacies in SE: שתי הסיבות העיקריות לפרויקטים מאחרים: הערכות שגויות (בהמשך) ודרישות לא ברורות
- "Easy access to end users is one of three critical success factors in rapid- development projects” (McConnell)


I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
E-mail: SCOTTADAMS@AOL.COM
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
© 2006 Scott Adams, Inc. /Dist. by UFS, Inc.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
www.dilbert.com
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

איך אפשר להגדיר דרישות?

- אז... אנחנו עובדים עם הלקוח להבין את הצרכים שלו, איך לוכדים את הדרישות האלו? מה האפשרויות?
- אב-טיפוס
- משחקי תפקידים, סיעור מוחות
- ראיונות, שאלונים, ביקור בית
- SRS – מפרט דרישות תוכנה
  - תרחישי שימוש Uses Cases
  - ביצועים, אילוצים, ...
  - ממשקים (משתמש וחיצוניים)
  - טבלת דרישות
- אג'ייל: סיפורי משתמש, סקיצות UI, מפרטים מורצים
- שיטות פורמליות (Z, FSM, Petri Nets)

"Documents are worthless, but documenting is everything". (Gause & Weinberg following Eisenhower Quote on planning)

תרחיש שימוש – Use Case

- מתאר דרך מסוימת להשתמש במערכת
- מייצג דו-שיח בין משתמש והמערכת מנקודת הראות של המשתמש -קופסה שחורה
- כלי ללכידת דרישות פונקציונליות
- פותחו ב- UML (ג'יקובסון, קוברן 90') פורמליות "רכה"
- “A specific way of using the system by using some part of functionality”, Jacobson

הגדרות

- שחקן Actor: מישהו שבא במגע עם המערכת
- בעל ענין Stakeholder: מישהו שיש לו ענין במערכת המפותחת (SuD)
- Use Case\תרחיש שימוש: חוזה על דרך פעולתה של המערכת
- שחקן ראשי: זה שמניע את התרחיש (UC)
- שם\מטרה: התוצאה הרצויה לשחקן הראשי ולבעלי הענין.
- הקף ורמה: הפרוט המופיע בתרחיש והיעד: ארגוני-מערכתי- תת-מערכתי

חלקי UC

- <u>טבלת שחקנים</u>
- <u>דיאגרמת תרחישים</u>, נתמכת בכלים שונים (UML):
  - מבט על חלק\כלל תרחישי המערכת
- <u>תרחיש שימוש</u> (UML)
- ובהמשך: סיפורי משתמש

בואו נבחן את הסוגים השונים:

טבלת שחקנים ובעלי עניין

- גם טבלה, של שחקנים ראשיים ומטרותיהם מהמערכת ז"א התרחישים שהם מניעים (מומלץ גם להבין את מטרות כל בעלי העניין)
- עבור מערכת השאלות:

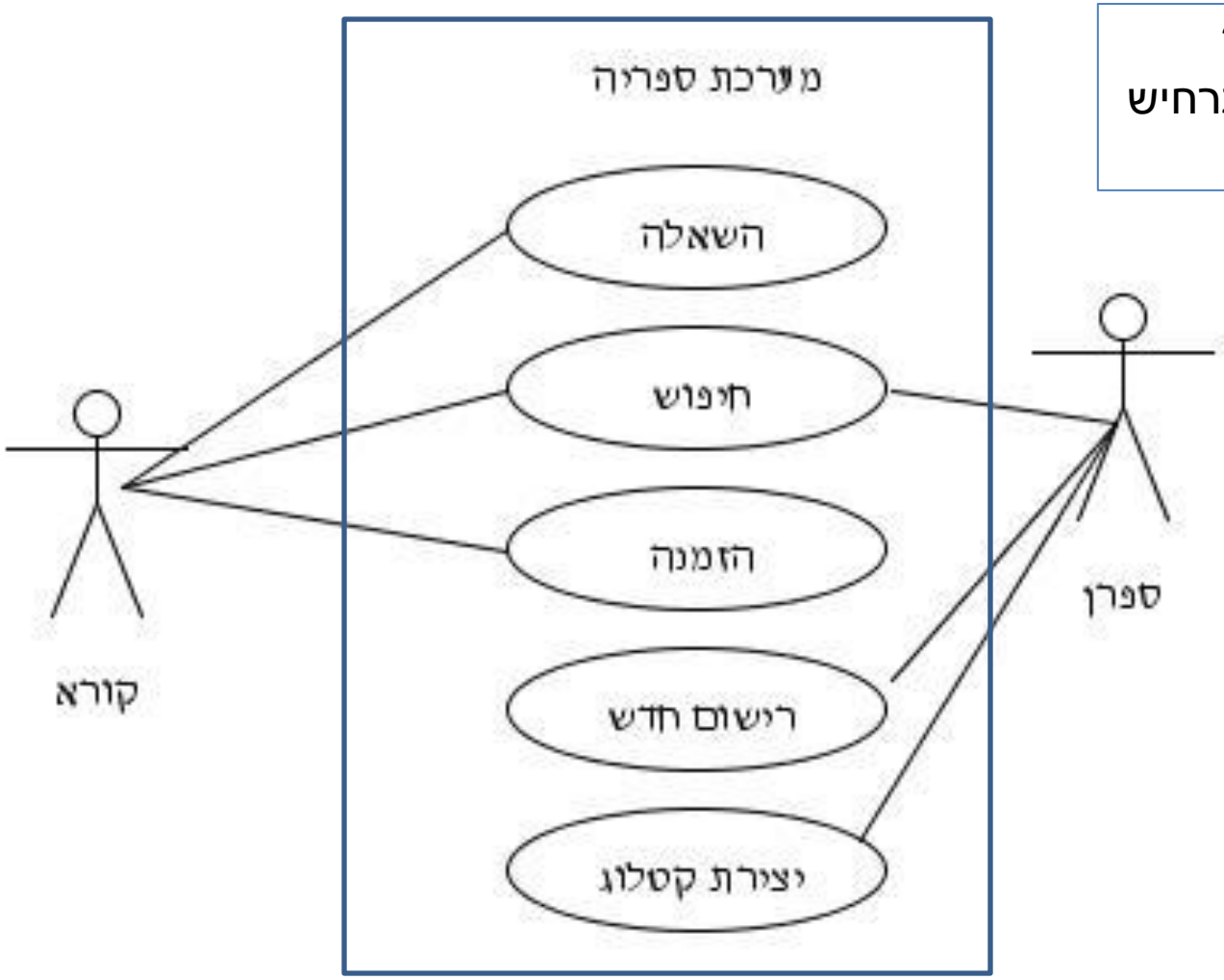
שחקן \ בעל-עניין	מטרות (ותרחישים)
קורא	חיפוש ספר
	השאלת ספר
	החזרת ספר
ספרן	חיפוש ספר
	בדיקת המצאות ספר
	בקשת ספר מספריה אחרת
המכללה	שירות לסטודנט

תרשימי (סיכום) תרחישים

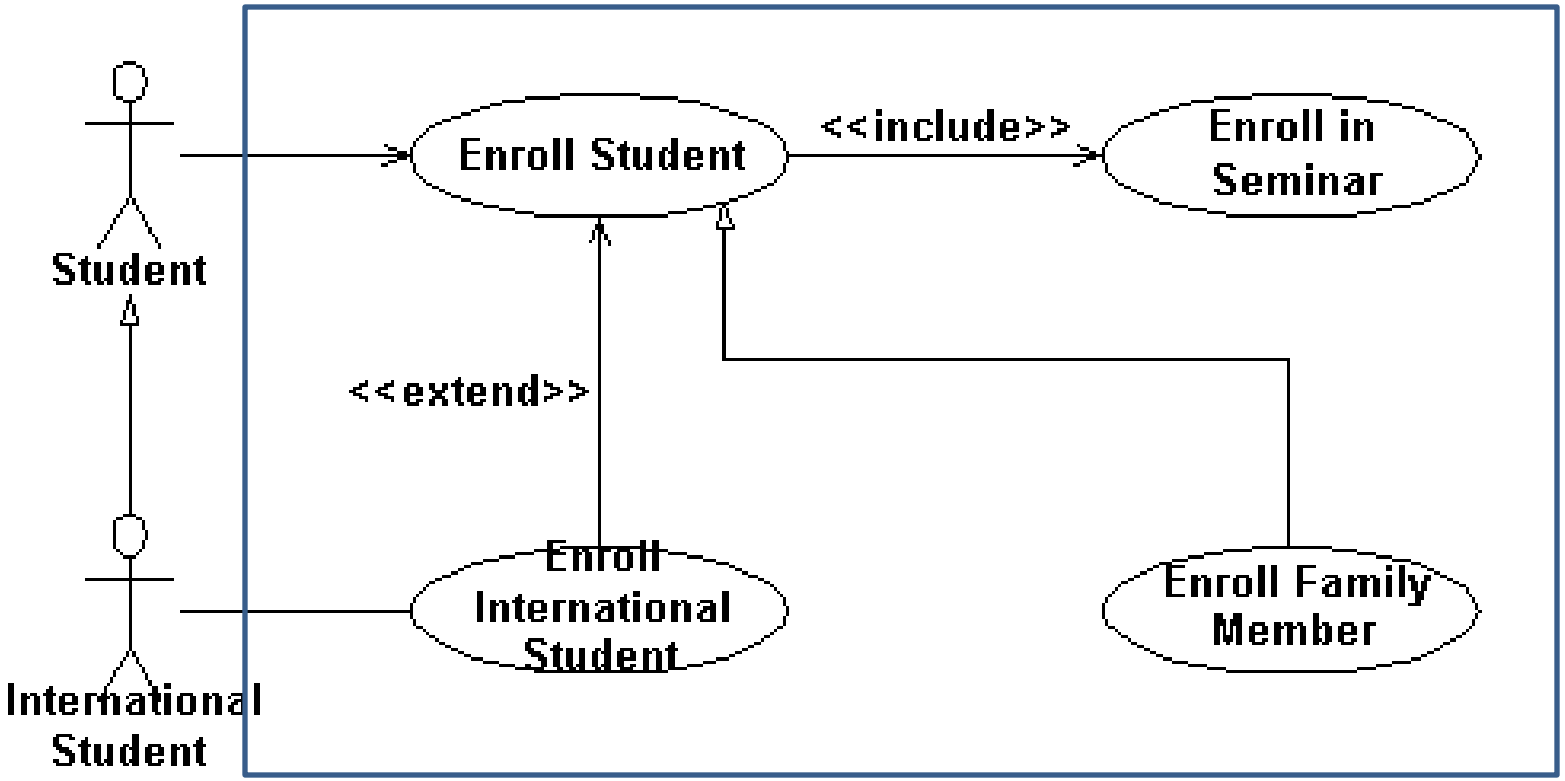
- דיאגרמה להצגת כלל\חלק מהתרחישים וקשרים שונים
  - שחקנים: בצורת אייקון בן-אדם, עם שמם (שם עצם)
    - עדיין יתכן שמדובר במערכות מחשב חיצוניות
  - תרחישים: בצורת אליפסה, עם שמם (פעולה – משהו שהשחקן רוצה להשיג)
  - קווי קישור, מקשרים את השחקנים עם התרחישים שהם משתמשים בהם (חץ עבור השחקן הראשי)
  - גבולות המערכת כמלבן מסביב לתרחישים
  - הקשרים בין הגורמים השונים
    - תרחישים יכולים גם להיות מחוברים לתרחישים שהם משתמשים או תלויים בהם

תרשים תרחישים

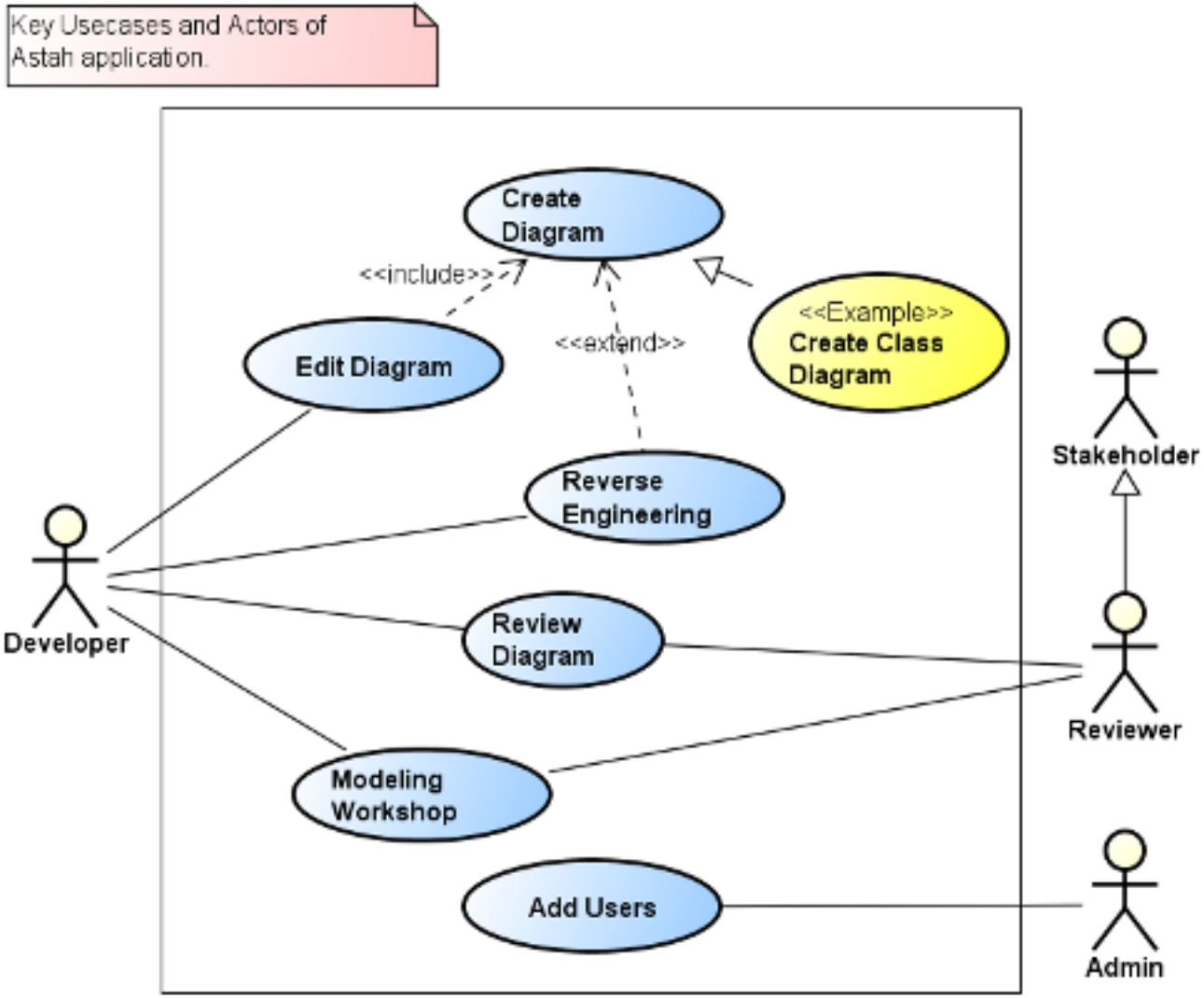
איפה צריך לשים חיצים?
מה צריך להוסיף עבור תרחיש השאלה בין ספרייתית?



תרשים תרחישים – עם קשרים



דוגמא נוספת


Key Usecases and Actors of Astah application.
Create Diagram
<<include>>
<<extend>>
<<Example>>
Create Class Diagram
Edit Diagram
Reverse Engineering
Review Diagram
Modeling Workshop
Add Users
Developer
Stakeholder
Reviewer
Admin

כלי UML

- IBM Rational Rose, Enterprise Architect, Rhapsody
- MS Visio, MS Visual Studio
- Free: ArgoUML / UMLet / StarUML
- Online: lucidchart, yUML, WebSequenceDiagrams, gliffy, draw.io
- Power Point… Napkin…

תרשים UC - יתרונות

- זיהוי שחקנים
- מבט כללי
- נקודת התחלה לפירוט הדרישות

- איך מפרטים הלאה?...

תרחיש לא-פורמלי

- נכתב כפסקה המתארת תרחיש\אינטראקציה מלא
  - עם נתונים ספציפיים
- דוגמא:

קורא מאבד ספר

הקורא מדווח לספרן שהוא איבד ספר. הספרן מדפיס את רשומת הספר ומבקש מהקורא לדבר עם מנהלת הספריה, שתקבע את גובה התשלום. המערכת תעודכן בנתוני הספר שאבד וכן כרטיס הקורא. מנהלת הספריה עשויה להורות על רכישת תחליף.

תרחיש שימוש (פורמלי) – דוגמא

שם	הזמנת ספר
שחקן ראשי	קורא
מטרה	קורא מעוניין לשריין ספר מתוך הקטלוג המקוון
הקף	מערכת הספריה
רמה	משתמש
בעלי עניין ואינטרסים	קורא – לשריין ספר בעל הספריה – שרות מורחב לרווחת הלקוחות
טריגר	הקורא נכנס למערכת
תנאי-קדם	הקורא עבר את מסך ההזדהות (login) ונחת בעמוד הבית
תנאי סיום מוצלח	הספר שמור עבור הקורא (האם זה תנאי מוצלח?)
תנאי כישלון	הספר אינו שמור

תרחיש שימוש -המשך

תרחיש הצלחה עיקרי	1. הקורא לוחץ בתפריט על הזמנת ספר 2. המערכת מציגה קטלוג עם מסך חיפוש 3. הקורא מזין את שם הספר 4. המערכת מציגה התאמות עם מיקומם 5. הקורא בוחר התאמה ובקשה לשמירה 6. המערכת מאשר את ההזמנה ומציגה את הקטלוג בחזרה
הרחבות (שגיאות)	2א. פג תוקף ה-login 2א.1. המערכת מחזירה את הקורא למסך הכניסה 2א.2. הקורא מתייאש או מנסה שוב 4א המערכת אינה מוצאת את הספר 5א.1. ...
תרחישים חלופיים	3. הקורא מזין מחבר או נושא

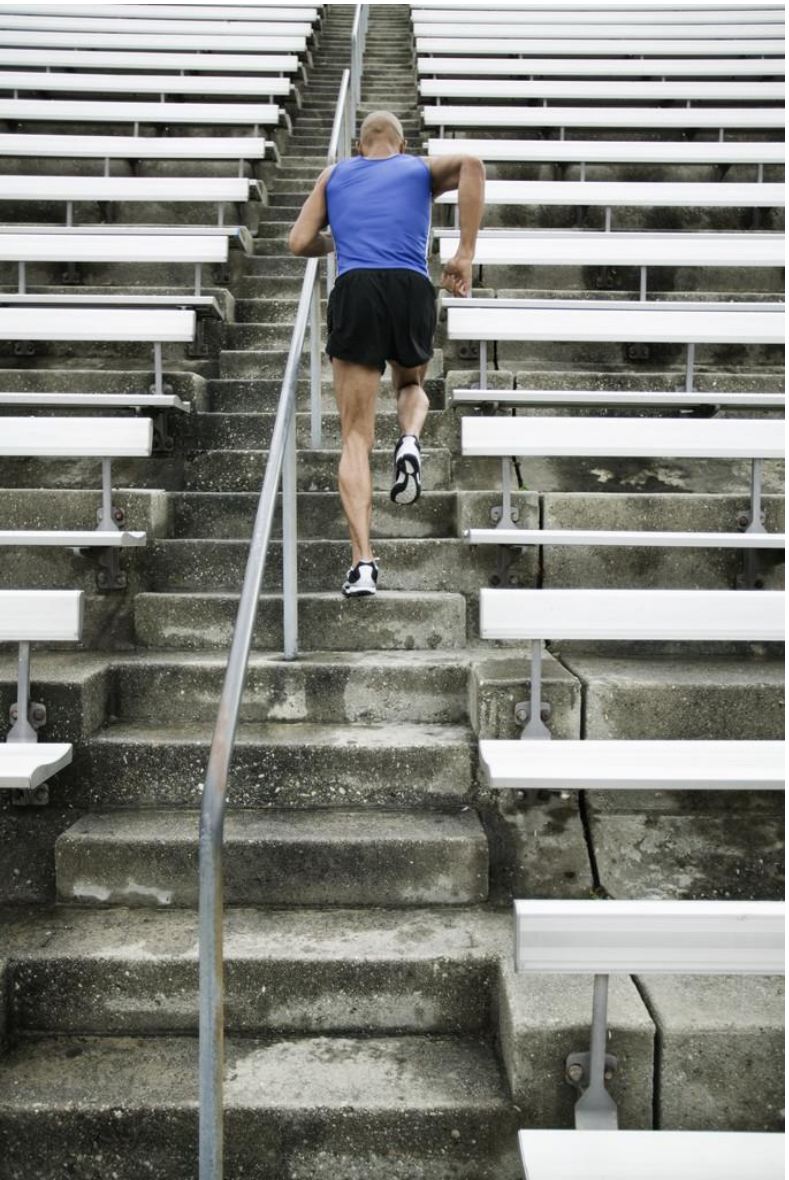
צעדים ליצירת תרחיש ביצוע

1. זיהוי שחקנים ומטרותיהם
   - א- אלו אנשים, מכונות ומערכות נוספות יהיו בקשר עם המערכת שלנו (שחקנים)
   - ב- מה כל שחקן צריך שהמערכת שלנו תבצע
   - ג- כדאי גם לפרט את המטרות של בעלי עניין אחרים
2. יצירת דיאגרמת תרחישים
3. פירוט לתרחישים פורמליים \ לא-פורמליים

סיכום ביניים: יתרונות לתרחישי שימוש

- מובן ללקוח אך עדיין פורמלי
- יצירת הבנה בין הלקוח והמפתחים בקשר לדרישות (תרחישי הצלחה)
- חושף את המפתחים לנושאים בעייתים (תרחישי הרחבה, חריגות)
- מאפשרים לתעדף מאפיינים ולתכנן בהתאם
- משמשים כקלט להמשך הפרויקט (הערכה, בדיקות, QA, ניהול)

תרגיל



- בואו נזהה שחקנים ומטרות עבור הפרויקטים שלכם
- תרשים תרחישים ראשוני

המשך תרחיש שימוש – כתיבת תרחיש הצלחה

- תרחיש ההצלחה העיקרי, הוא המסלול המועדף כשהכל הולך חלק
  - הקל ביותר לקריאה והבנה
  - כל השאר הם הסתעפויות וסיבוכים
- מתאר את מהלך העבודה של שחקן מהטריגר ועד לביצוע משימתו
  - צעדים ברורים וממוספרים

הרחבות כשלון

- בד"כ כמעט כל שלב יכול להיכשל
- מציינים זאת לאחר תרחיש ההצלחה
- אך מקשרים למספר הצעד הרלוונטי

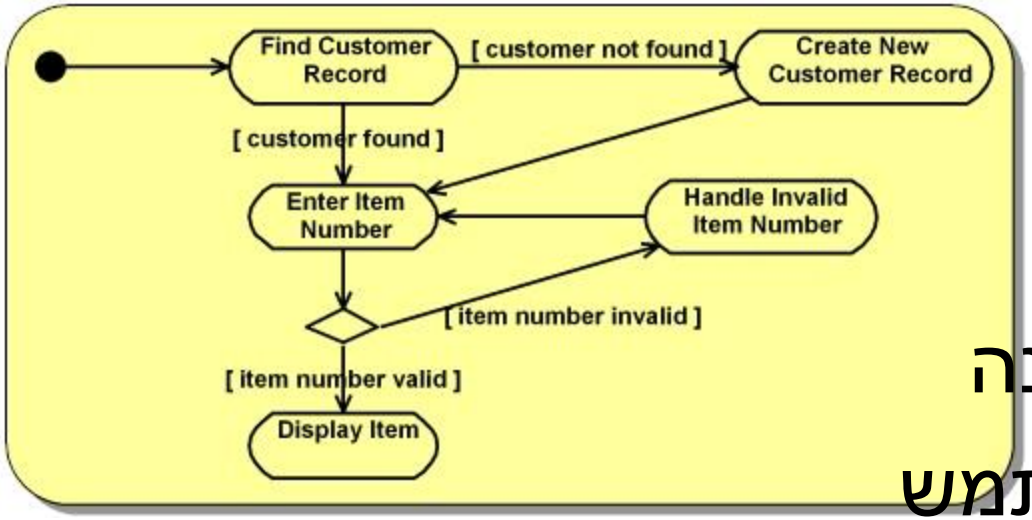
חלופות אחרות

- להרבה צעדים יכולה להיות התנהגות חלופית
- מקשרים גם כן למספר הצעד
- למשל:
  - 5'. חלופה ראשונה לצעד 5
  - 5". חלופה שניה לצעד 5

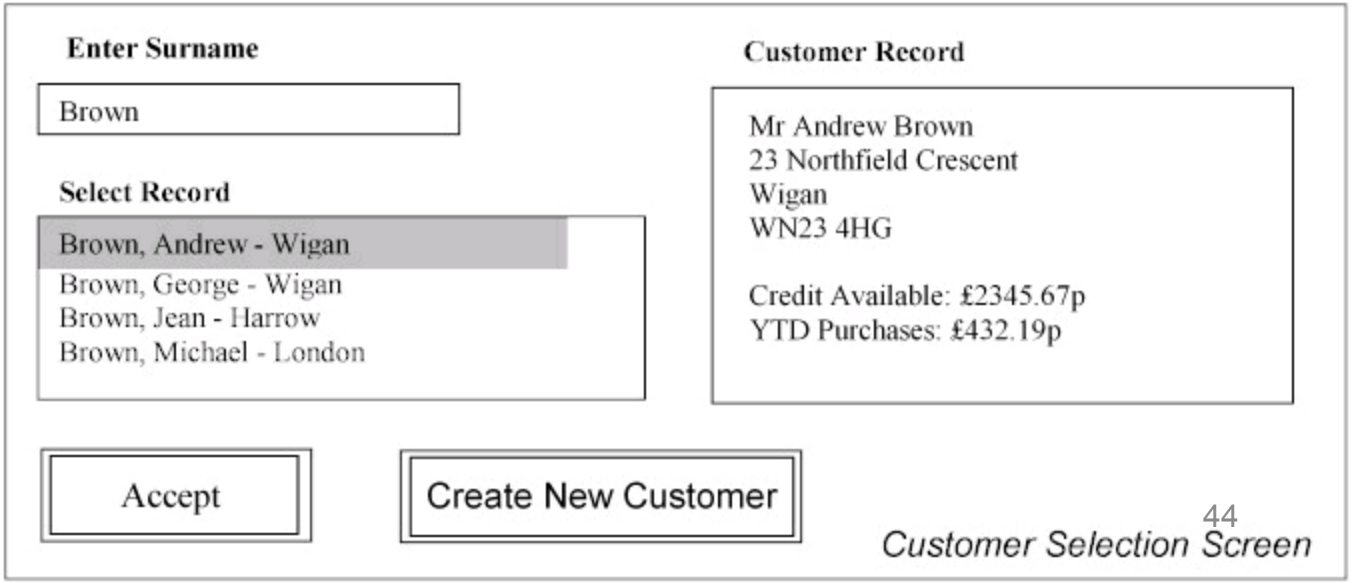
מאפייני תרחיש ביצוע טוב

- מתחיל בפניה של שחקן למערכת
- נגמר במענה לכל צרכי הבקשה
- מגדיר את יחסי הגומלין (בין השחקן והמערכת) שקושרים לבקשה
- נכתב מנקודת ראות של השחקן ולא המערכת
- מתרכז ביחסים ולא בפעולות פנימיות של המערכת
- אין תאור מפורט של GUI
- תרחיש הצלחה עיקרי בעל 3-9 שלבים
- נוח לקריאה (נכנס בעמוד)

מה עוד מכיל מפרט דרישות



- דיאגרמות פעילות
- מכונות מצבים
- ממשקי חומרה ותוכנה
- תרשימי ממשק משתמש



היה היו דרישות...

- מדרישות לרשימת משימות
- Alberto Savoia, **GTAC 2011: Opening Keynote Address - Test is Dead** http://www.youtube.com/watch?v=X1jWe5rOu3g&t=5m48s
- והיום...

תשובה 2 אג'ייל: סיפורי משתמשים

- הגדרת דרישות ברמה כללית בלבד
- כוללים מספיק מידע שיאפשר הערכה למימוש
- בד"כ קצרים מתרחישי שימוש
  - יכולים להיות כותרת של תרחיש
- תזכורת לשיחה עם הלקוח, דוגמאות:
  - הקורא יכול להשאיל ספר באופן מקוון
  - המערכת שולחת תזכורת במייל כאשר תאריך ההשאלה פג
- Cockburn: “promissory notes for future conversation”

Ron Jeffries: CCC

- Card
  - סיפורים נכתבים על כרטיסיות
  - מוסיפים עליהם הערכה, עדיפות ועוד
- Conversation
  - הפירוט מגיע בשיחה עם הלקוח
- Confirmation
  - בדיקות קבלה מאשרות שהסיפור מומש נכון

תבנית לסיפור (Connextra)

כותרת (עד חמש מילים)

- בתור ...
- עבור **מי** הסיפור
- אני מעוניין ....
- **מה** הוא\הם רוצה לעשות
- כך ש ...
- **למה** הוא רוצה לעשות זאת

לדוגמא - רכישת דיסק

- בתור חובבת מוסיקה
- אני מעוניינת לראות את הכותרים האחרונים
- כך שאוכל להזמין ולהנות מהדיסק של...

דוגמא (הרחבה לתבנית)

- הערות:
  - למשתמשים אכפת בעיקר משם היוצר
  - אבטחה: הצגת שם המשתמש במידה וביצע כניסה מאובטחת

- קריטריון קבלה:
  - מופיע שם היוצר עם רשימת דיסקים אחרונים
  - מופיע כפתור "הזמנה"

מה הבעיה העיקרית בסיפור הבא:

בתור משתמש אני מעוניין להינעל אחרי 3 כניסות שגויות

1. חסר החלק של "כך ש.."
2. זיהוי לא נכון של בעל העניין
3. משתמש בכלל לא מעוניין להינעל
4. כתוב בעברית

INVEST in User Stories

- [Bill Wake]:
  Independent, Negotiable, Valuable, Estimatable, Small, and Testable.
  - עצמאי, פתוח למשא ומתן, בעל ערך ללקוח, ניתן להערכה, קטן, ניתן לבדיקה
  - (עדיף מקצה לקצה)
  - (ר"ת אחרים S.M.A.R.T)

אנטי-דוגמאות

- "אוטמציה של שרת ה-build"
  - ללא ערך ללקוח
- "דיפלוימנט לשרת סטייג'ינג מחוץ לפיירוול"
  - פרטי מימוש ללא התוצאה, טרמינולוגיה שהלקוח אינו מכיר -> "פרסום דמו שהלקוח יכול להשתמש בו"
- "האתר עולה מהר"
  - חסר קריטריון ברור להערכה -> "החיפוש מסתיים תוך שניה"

אילו מהבאים הוא הכי פחות INVEST

1. המשתמש יכול לחפש סרט לפי כותרת
2. לאתר הסרטים צריכה להיות תגובתיות טובה
3. כשמוסיפים סרט למאגר ב-99% מהזמן הדף צריך להחזיר תשובה בפחות מ-3 שניות
4. כלקוח, אני רוצה לראות את רשימת הנמכרים ביותר ממוינת לפי מחיר, כך שאוכל לקנות קודם את הזולים

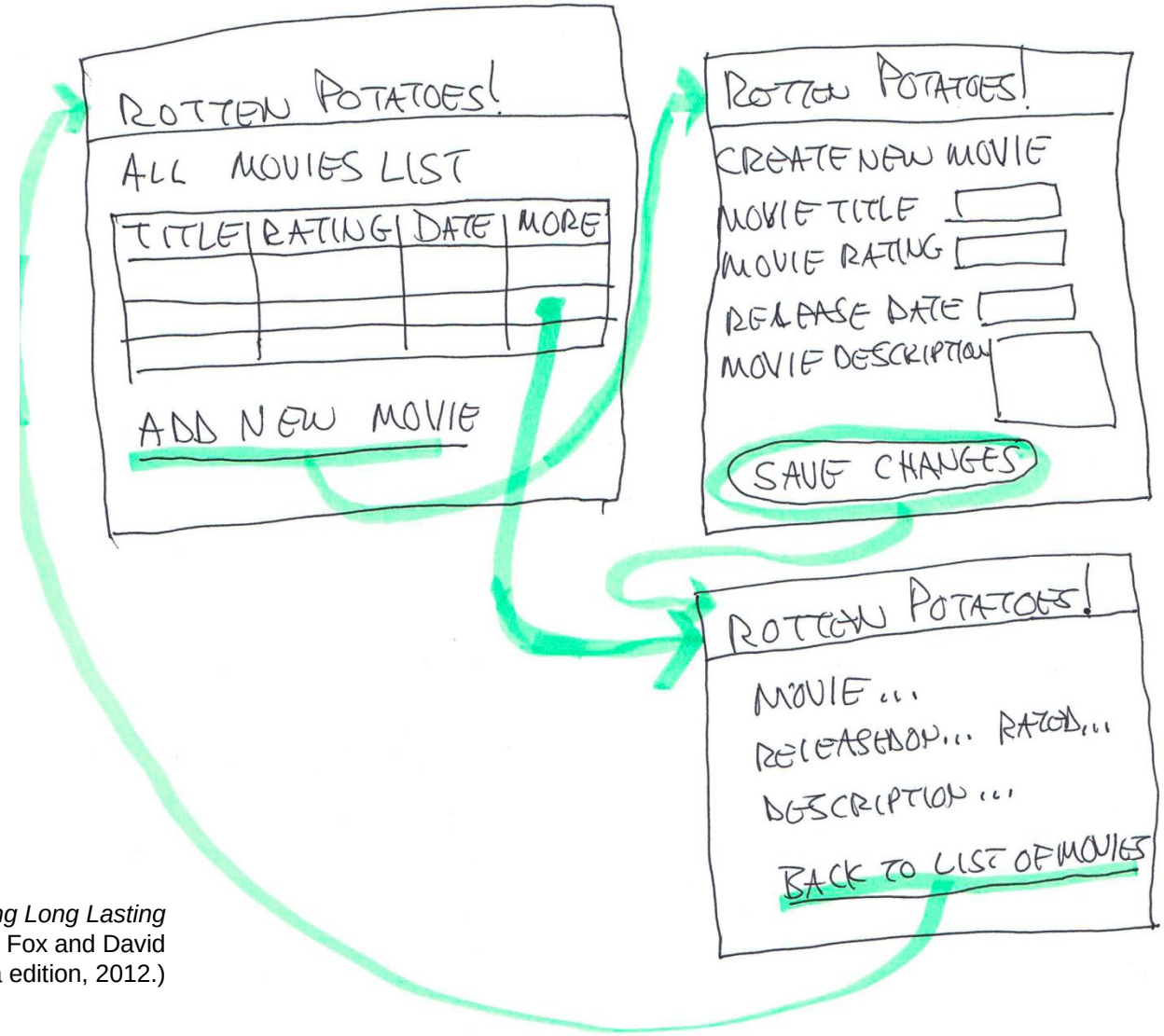
תרגיל

- נסחו 3-4 סיפורי משתמש לפרויקט שלכם
  - כותרות בלבד
  - בדקו שהם INVEST

סקיצות ו- Storyboard

- הדגמת שינויי ממשק בעקבות פעולות המשתמש
- תחום ממשק אדם-מכונה (HCI)
  - בדומה לתכנון סצנות בסרט
- עוד כלי להעצמת צד המשתמש
- כלים:
- Axure, Balasamiq, vs2012 power point storyboarding
- פרוטוטייפ
- עוד בהרצאת ux בהמשך

דוגמא ל- Storyboard



(Figure 4.4, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Alpha edition, 2012.)

תרגיל

- שרטטו מסך עיקרי ראשוני למוצר שלכם
- תכננו מעברים למסכים אחרים

האם הדרישות נכונות?

- מה יכול להשתבש?
-
  - חוסרים
  - סתירות
  - טעויות עובדתיות
  - דו-משמעות
- מה עושים?

כיצד מאמתים דרישות?

- סקר
- ניתוח
- אב-טיפוס
- יצירת מפרט בדיקות
- כלי טקטס (NASA Automated Requirement Measurement)
- הרצת הדרישות (בשיטות פורמליות)
- מפרטים מורצים (BDD, AUAT)..

~Demo: Cucumber

1: Describe behaviour in plain text

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I have entered 70 into the calculator
    When I press add
    Then the result should be 120 on the screen
```

2: Write a step definition in Ruby

```
Given /I have entered (.*) into the calculator/ do |n|
  calculator = Calculator.new
  calculator.push(n.to_i)
end
```

3: Run and watch it fail

```
$ cucumber features/addition.feature
Feature: Addition  # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers                          # features/additi
    Given I have entered 50 into the calculator      # features/step_d
      uninitialized constant Calculator (NameError)
      ./features/step_definitons/calculator_steps.rb:2:in `Given /
      features/addition.feature:7:in `Given I have entered 50 into
    And I have entered 70 into the calculator        # features/step_d
    When I press add                                 # features/additi
    Then the result should be 120 on the screen      # features/additi
```

4. Write code to make the step pass

```
class Calculator
  def push(n)
    @args ||= []
    @args << n
  end
end
```

5. Run again and see the step pass

```
$ cucumber features/addition.feature
Feature: Addition  # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers                          # features/additi
    Given I have entered 50 into the calculator      # features/step_d
    And I have entered 70 into the calculator        # features/step_d
    When I press add                                 # features/additi
    Then the result should be 120 on the screen      # features/additi
```

6. Repeat 2-5 until green like a cuke

```
$ cucumber features/addition.feature
Feature: Addition  # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers                          # features/additi
    Given I have entered 50 into the calculator      # features/step_d
    And I have entered 70 into the calculator        # features/step_d
    When I press add                                 # features/step_d
    Then the result should be 120 on the screen      # features/step_d
```

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I have entered 70 into the calculator
    When I press add
    Then the result should be 120 on the screen
```

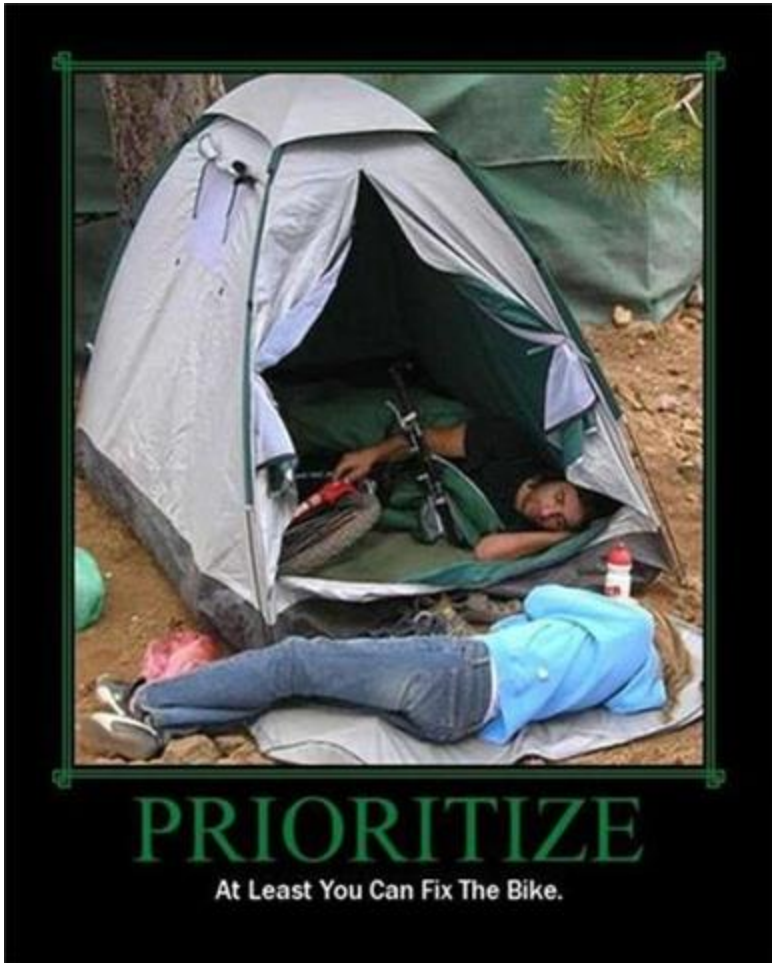
```
$ cucumber features/addition.feature
Feature: Addition  # features/addition.feature
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
  Scenario: Add two numbers                          # features/additi
    Given I have entered 50 into the calculator  # features/step_d
      uninitialized constant Calculator (NameError)
      ./features/step_definitons/calculator_steps.rb:2:in `Given /
      features/addition.feature:7:in `Given I have entered 50 into
    And I have entered 70 into the calculator    # features/step_d
    When I press add                             # features/additi
    Then the result should be 120 on the screen  # features/additi
```

מתרחישים לטבלת דרישות

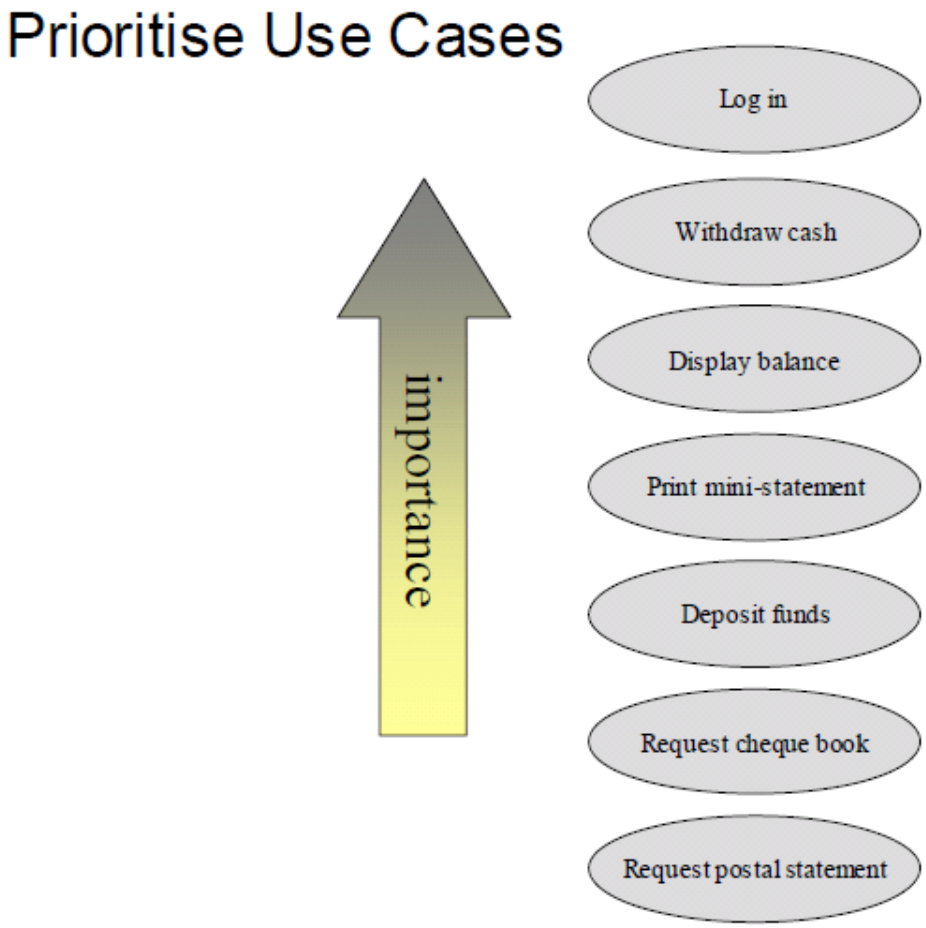
- תרחישים => (בעיקר) כלי ניתוח
- טבלת דרישות \ Backlog \ URD => כלי לתכנון הפיתוח
- עקיבות בין חלקי התהליך

עקיבות דרישות

- ציון המקור לכל דרישה
- מספור כל דרישה, כך שתוצרים בהמשך יוכלו להיתלות בה
- מה החשיבות בכך?

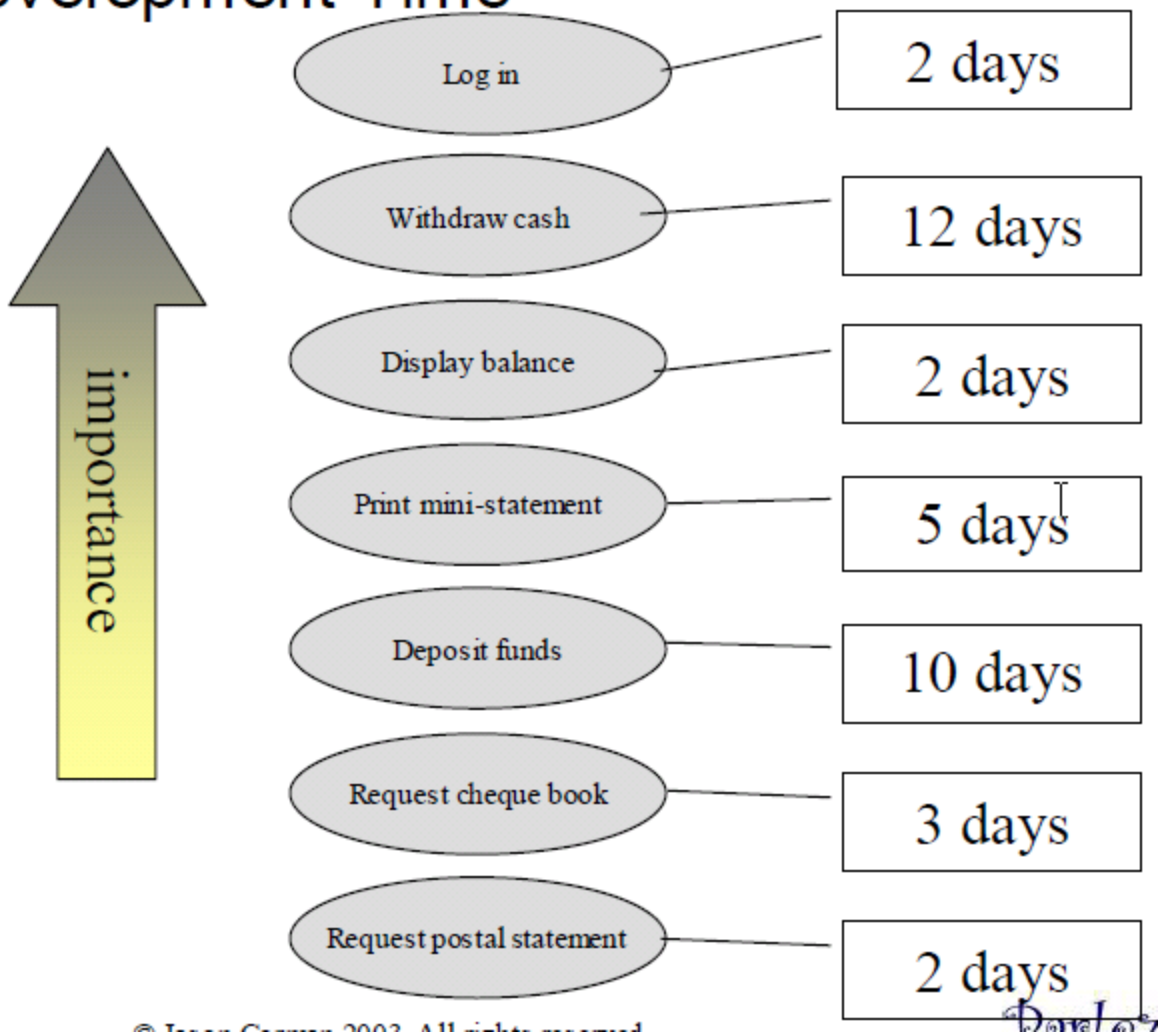

PRIORITIZE
At Least You Can Fix The Bike.

תיעדוף תרחישים



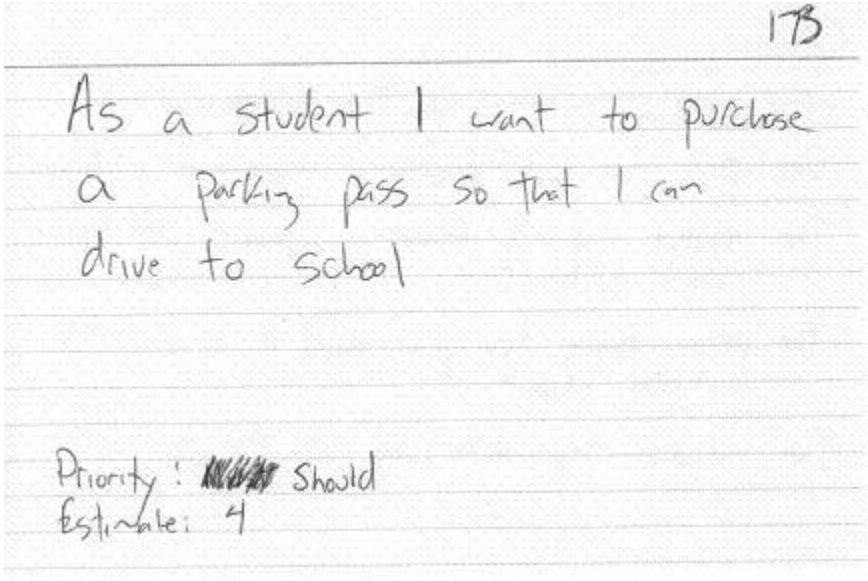
הערכה בעזרת תרחישים (בהמשך)

- Estimate Development Time



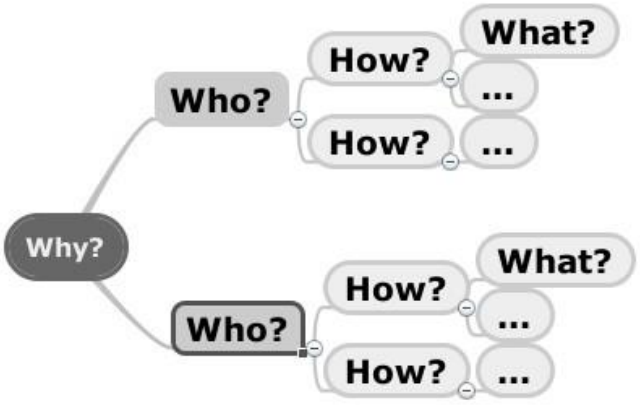
ניהול ותכנון עם דרישות

- כלים, למשל: Door , GitHub Issues, ...



שיטות נוספות

- Mind Map, Effect Map,
- User Story Mapping
- Feature-Driven Development
- Real Options
- Lean Startup
- מתי מתאימות?



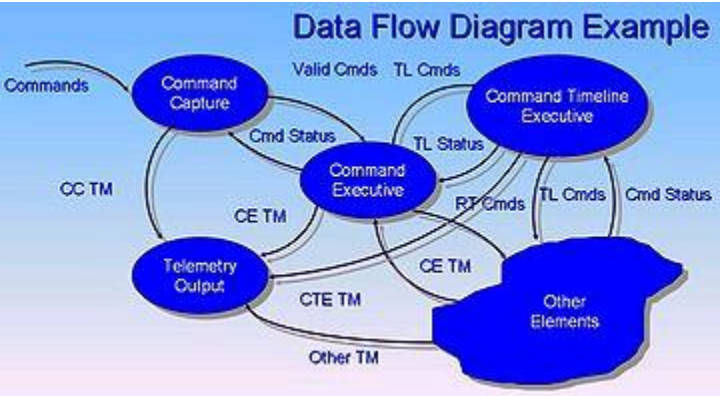
תרגיל

- עבודה על UC מפורט למוצר שלכם

התוצרים יכולים כבר לשמש למסמך ה-SRS שלכם

בפעם הבאה

- סקר מפרט דרישות (רישום ביומן הפגישות)
- דיברנו על ה"מה?"
- בדרך אל ה"איך?" וה"מתי?":
  - תיכון\ארכיטקטורה
  - בהמשך: הערכה ותכנון + כלי מרכזי: בקרת גרסאות



סיכום\דיון

- כמה צריך לפרט?
  - פשטות לעומת נכונות מפורטת
  - גראפי לעומת תאור מילולי וטבלאי מדויק
  - קצר ולענין לעומת מלא ומאוחר
- האיזון תלוי גם בלקוח ובטיב ביחסים איתו (או שיטת הפיתוח)
  - האם אחרי אג'ייל יש מקום ל-SRS?
  - תרחישי שימוש: אנשים הם גם חלק במערכת!
- פיתוח מפרט בסבבים (הורדת\דחיית מה שלא הכרחי, הפשטה)
- שיטות: סיפורי, משתמש, UC ככלי המרכזי ב-OO לנתוח דרישות
  - דיאגרמות נוספות Activity, State Machine
  - בעבר, למשל ב- Structured Analysis
    דיאגרמות אחרות: Context, ERD, DFD
- כלים: כלי UML
- הנדסת דרישות כמקצוע

עוד כמה הגיגים

- The most difficult part of requirements gathering is not the act of recording what the users want; it is the exploratory, development activity of helping users figure out what they want. McConnell
- Work with a User to Think Like a User – it's the best way to get insight on how the system is easily used. Pragmatic Programmer Tip
- I like to listen. I have learned a great deal from listening carefully. Most people never listen. Ernest Hemingway
- Embrace simplicity in your product and in your code. The value is in what gets used, not what gets built. – Kris Galle