

# הנדסת תוכנה

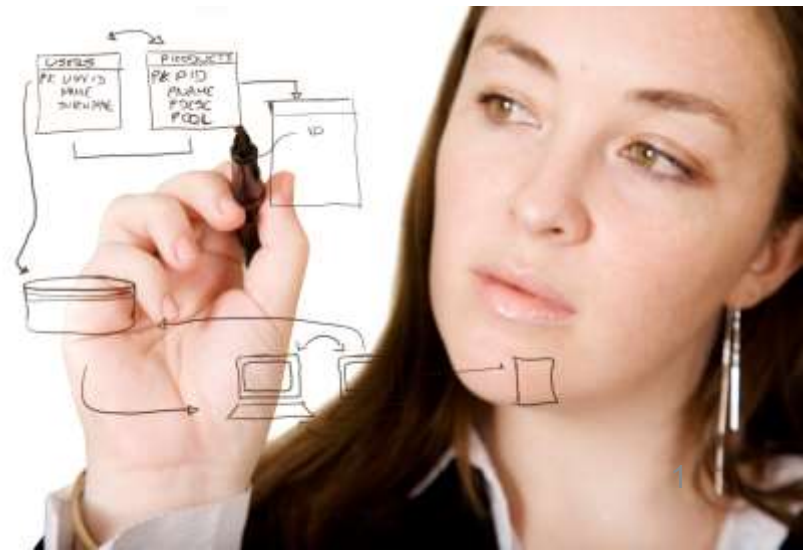
קורס 10014

סמסטר א' תשע"ד

## 1. מבוא

ד"ר ראובן יגל

[robi@post.jce.ac.il](mailto:robi@post.jce.ac.il)



# השבוע

- מבוא להנדסת תוכנה
- לוגיסטיקה – סילבוס \ אתר
  - ש.ב. 1 – רישום לקורס
- הפרויקט
  - מטרות
  - הדגמה
  - משימת פרויקט 1: רעיון
- תרגיל
  - הכרות עם סביבת העבודה בקורס

# מקורות להרצאה

- Pressman, chap. 1, “Software and Software Engineering”
- [ויקיפדיה](#) – הנדסת תוכנה\SE
- IEEE SWEBOK'04
- Laplante, “What every engineer should know about software engineering”
- Brooks, Mythical Man Month

# שאלות

- מהי "הנדסת תוכנה"?
- האם זו הנדסה? האם זה כמו תכנות?
- האם זה חשוב? מעניין?
- מהם ההבדלים בין תרגיל סטודנט לפרויקט תעשייתי?
- עוד....

# הנדסת תוכנה היא:

- לא רק תכנות
- ?

# מהי הנדסת תוכנה?

**“Software Engineering** is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software”

[IEEE SWEBOK’04], Wikipedia

Software engineering has accepted as its charter,  
"How to program if you cannot." -- *E. Dijkstra*

# 1968 NATO SE Conference

- “We undoubtedly produce software by backward techniques.” “We build systems like the Wright brothers built airplanes—build the whole thing, push it off the cliff, let it crash, and start over again.”

• מה מצבינו כיום?

*Software is becoming the foundation of modern civilization; software constitutes or will control the products, services, and infrastructure people will rely on for a wide variety of daily activities from the vital to the trivial. ... software is not sufficiently engineered at this time to fulfill the role of “foundation.”*

David Rice



# IEEE - SWEBOK

- מסמך\מיזם (2004), הגדרת עשרה תחומי ידע בהנדסת תוכנה וכן דיסציפלינות קרובות ([ויקיפדיה](#))
  - דרישות תוכנה
  - תכנון תוכנה
  - בניית תוכנה (תכנות מחשבים)
  - בדיקות תוכנה
  - תחזוקת תוכנה
  - ניהול תצורת תוכנה
  - ניהול הנדסת תוכנה
  - תהליכי הנדסת תוכנה (תהליכי פיתוח תוכנה)
  - כלים ושיטות בהנדסת תוכנה
  - איכות תוכנה
- הנדסת מחשבים
  - מדעי המחשב
  - ניהול
  - מתמטיקה
  - ניהול איכות
  - הנדסת אנוש
  - הנדסת מערכות

# תשובה מקוצרת שלנו

- אוסף תהליכים, שיטות וכלים לפיתוח מוצר תוכנה בעל ערך ללקוח ויכולת התאמה למצבים שונים תוך שימוש מיטבי ואיכותי במשאבים, משלב הרעיון ועד לשלב הפרישה

# TIVI

- Glenn Vanderburg: "Software engineering is the **science and art** of designing and making, with **economy and elegance**, [...] systems so that they can readily adapt to the situations to which they may be subjected."  
[confreaks 2010](#) [Qcon 2012](#)
- “Working software that matters” – D. North

# Alt.: The One Thing Every Software Engineer Should Know

1. people understand what you're doing
2. people become interested in what you're doing
3. people get excited about what you're doing

<http://www.codinghorror.com/blog/archives/001177.html>

# האם זו הנדסה? מתכנת\ת או מהנדס\ת תוכנה?

- מתכנת -> מהנדס ([ויקיפדיה](#), [stack overflow](#))
- מהנדס: משמעות חוקית, אחריות

– חוקי חמורבי:

(9) בנאי הבונה בית ברשלנות, ואדם מת בעקבות כך -  
יהרג הבנאי, ואם נהרג הבן של בעל הבית בשל  
רשלנות הבנייה - יהרג בנו של הבנאי <sup>1,2</sup>.

– ["Bad coder to Jail"](#)

[Pragmatic Programmer Tip](#) #1:

**Care about Your Craft**

Why spend your life developing software  
unless you care about doing it well?

# רשיון?

- “ACM's position is that our state of knowledge and practice in software engineering is **too immature to warrant licensing**. Moreover, Council felt licensing would be ineffective in providing assurances about software quality and reliability.”
  - [SWEBOK Pulled out...](#)

# מהם ההבדלים בין תרגיל כיתה לפרויקט תוכנה תעשייתי?

- פרויקט (שלבים ויכולות שונים)
- עבודת צוות
- בעיות ותהליכים מסובכים
- דרישות סותרות ומשתנות
- שינויים לאחר ותוך כדי הפיתוח
- עלות תקלות
  - כלכלית
  - בטיחותית
- ראייה מערכתית
- לקוח! צריך לעזור לו!

# מהי בעצם תוכנה?

- דיברנו קצת על הנדסה
- תוכנה - מוצר\שירות הכולל:
  - תוכניות מחשב להרצה (קוד)
  - מסמכים מלווים
  - נתונים (+תצורה, סקריפטים וכו')
  - במטרה שמישהו יהיה מרוצה:
- לרוב מיוצר ע"י קבוצה עבור אחרים
- נראה בהמשך



# תוכנה – פנים שונות

- מוצר
  - ניצול חומרת המחשב \ תקשורת
  - עיבוד מידע
- תשתית לפיתוח מוצרים
  - כלים (למשל office, graphics)
  - בקרת תוכניות אחרות (מערכת הפעלה)
  - תקשורת למחשבים אחרים
  - כלים לבניית תוכנה
  - שרותים
- כיום קטגוריות רבות ושונות

# מהי תוכנה טובה (איכותית [ISO/IEC 9126](#))?

- פונקציונליות
  - תקפות (The right product)
  - נכונות (The product right)
  - ...
- אמינות, שימושיות, יעילות, ניידות
- תחזוקתיות / **גמישות לשינויים**
  - הוספת תכונות
  - תיקון באג
  - אופטימיזציה
  - שיפור מבנה

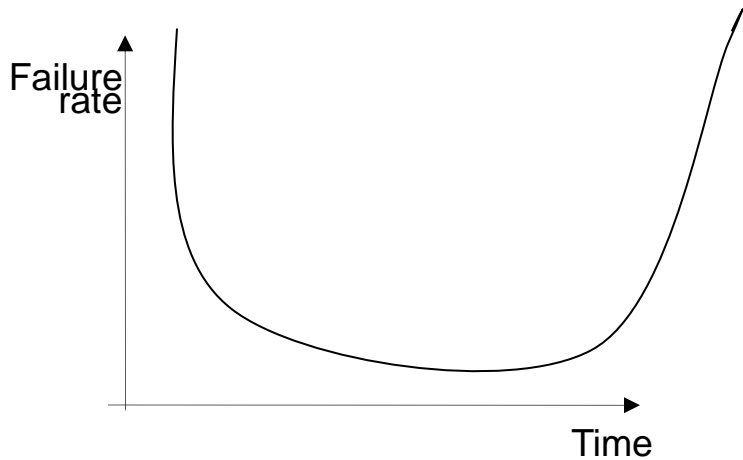
Good software is software that is designed to easily survive **change**.

[Stephen Walther](#)

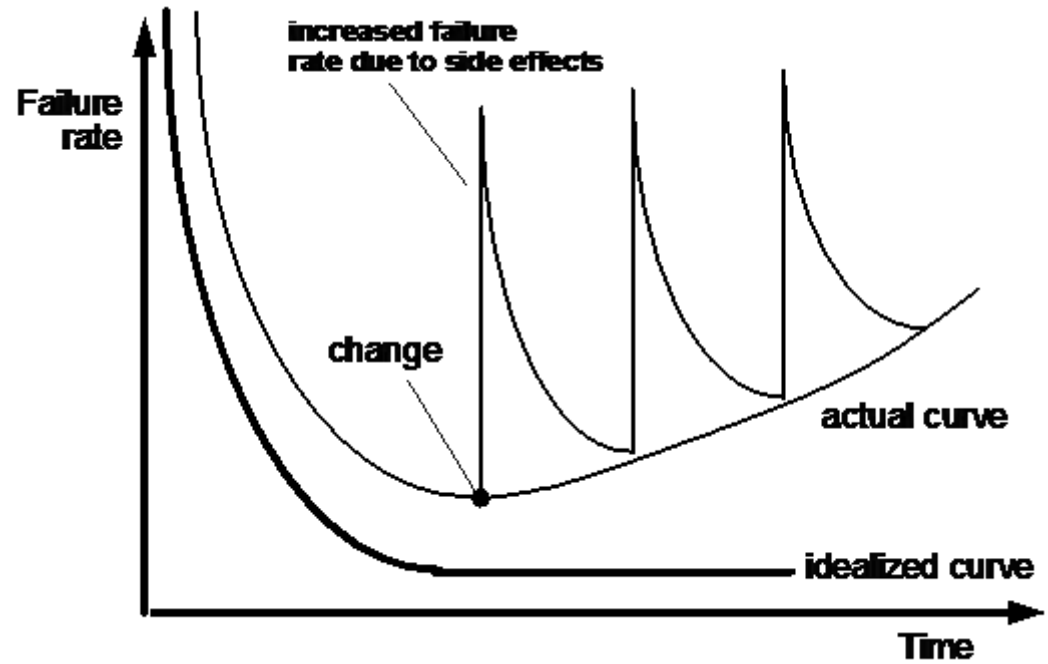
# מה מיוחד (קשה) בתוכנה?

- תוכנה נבנית (מהונדסת?)
  - כמעט כל פרויקט עוסק במשהו שלא היה קיים מעולם (לפחות בארגון, עבור הלקוחות, טכנולוגיה מסוימת ובד"כ בשילוב)
- תוכנה לא מתבלה (אך כן מתקלקלת)
- בד"כ מסובכת (א"א לבדוק את כל המצבים)
- מצד שני: השפעה עצומה על החברה האנושית
- Bjarne Stroustrup: "our civilization runs on software"  
Marc Andreessen, [2011](#): "software is eating the world"
- ובכ"ז תחום חדש יחסית

# יחודיות התוכנה



Failure for Hardware



Pressman, p. 37-38

# מכונות

• 1943: מספר המחשבים = 1

• 2010: < 500 מיליארד

• 2050 ?

## Technology Cycles Have Tended to Last Ten Years

Mainframe  
Computing  
1960s



Mini  
Computing  
1970s



Personal  
Computing  
1980s



Desktop Internet  
Computing  
1990s



Mobile Internet  
Computing  
2000s



Wearable /  
Everywhere  
Computing  
2014+



Others?

# מהו פרויקט?

- מאמץ זמני (עם התחלה וסיום מוגדרים) ליצירת מוצר או שרות ייחודי, השונה מכל מוצר/ שרות אחר

PMBOK

- מאפיינים עיקריים
  - חד פעמי – אין שני פרויקטים זהים
  - תחום בזמן – נקודת התחלה וסיום מוגדרות
  - הפרויקט לא כולל את שלבי התפעול השוטפים

# מה מיוחד\קשה בפרויקט תוכנה?

- Brooks: סיבוכיות מובנית מול אקראית  
“No Silver Bullet”

– סיבוכיות

– תאימות (לכל הדרישות)

– גמישות לשינויים

– חוסר נראות (סינדרום 90% לסיום)

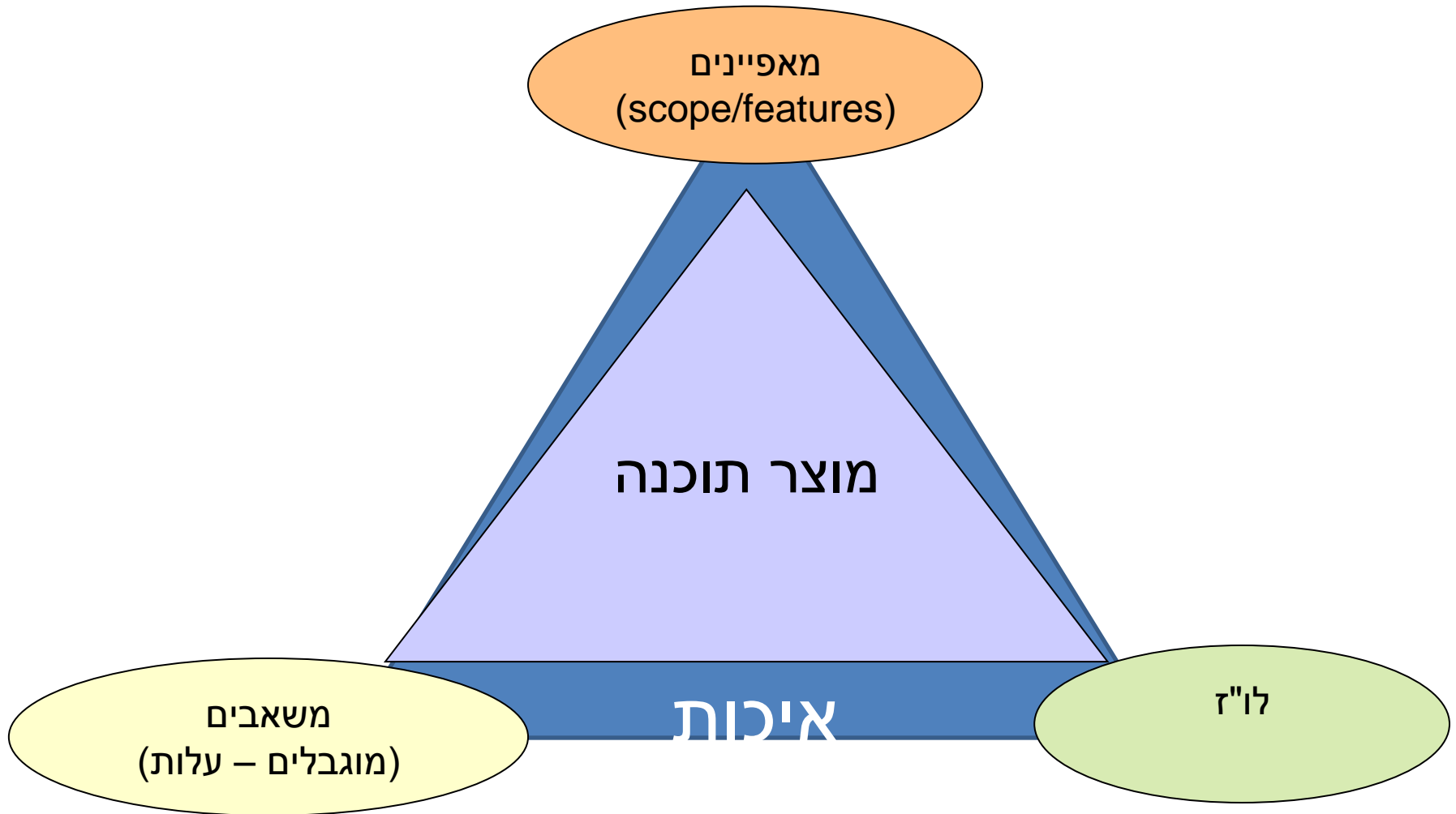
- בד"כ: עבודה אינטלקטואלית,  
עבודת צוות, רב תחומי

- רכישת ידע

Why software projects are not routine work



# אילוצי פרויקט (תוכנה)





# מה אחריותי כמנהל פרויקט\מוצר?

- תכנון והערכה
- מדידה ובקרה
- ניהול סיכונים
- תקשורת, תיאום והנהגה, סילוק מכשולים, שחרור הצוות מכל השאר, ...
- האם רק המנהל אחראי על כך?

# ניהול גרוע של פרויקט גורם ל:

- בניית המוצר הלא-נכון
- בניית מוצר בעל איכות נמוכה
- איחור
- ביטול
- ע(ו)בדים 80 שעות בשבוע
- ...

# עוד קצת הסטוריה

- משבר התוכנה – החל משנות ה-60

– פרויקטים בד"כ מאחרים ועולים יותר מהמתוכנן

– אחוז גבוה של כשלונות

– חוות דעת ציבורית גרועה

- משבר קבוע? היכן הבעיה?

- ועידת נאטו 1968

- התגובה: דיסיפלינה: מודלים,

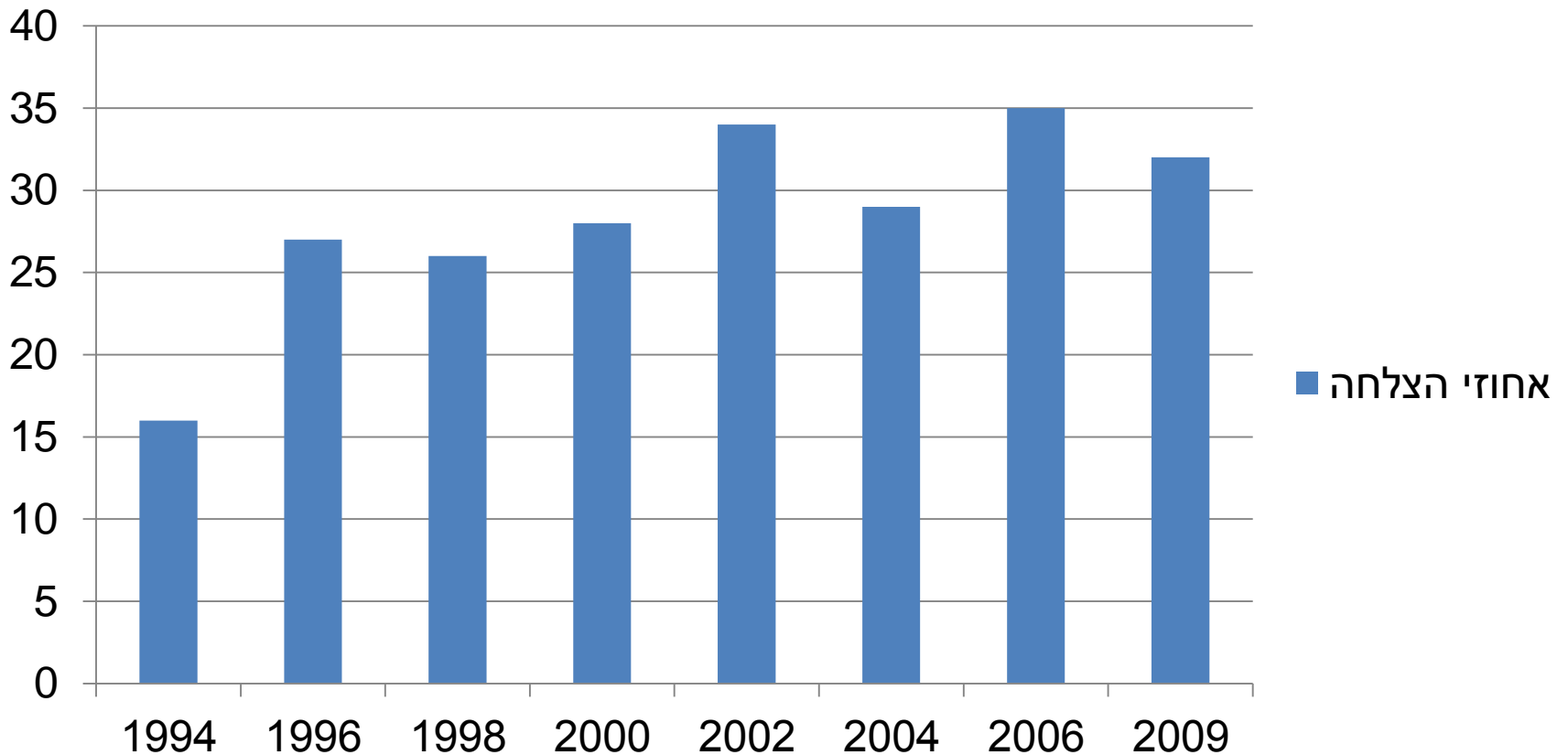
תהליכים, תקנים, שפה משותפת

- לאחרונה: קבוצת SEMAT

Wikipedia: "The software crisis has been fading from view, because it is psychologically extremely difficult to remain in crisis mode for a protracted period (more than 20 years)"  
-See also: David Notkin: [A Software Crisis? Please, sir, may I have some more?](#)

# Chaos Report (debatable...)

## אחוזי הצלחה של פרוייקטי תוכנה



# מדוע פרויקטי תוכנה נכשלים לעיתים כל כך קרובות? \*



- יעדי הפרויקט לא מציאותיים או לא ברורים
- אומדנים לא מדויקים של המשאבים הנדרשים
- דרישות מערכת אינן מוגדרות היטב
- דיווח לקוי לגבי מצב הפרויקט
- סיכונים לא מנוהלים
- תקשורת לקויה בין הלקוח, המפתחים והמשתמשים
- שימוש בטכנולוגיה לא בשלה
- אי-יכולת לנהל את מורכבות הפרויקט
- פרקטיקות פיתוח מרושלות
- ניהול לקוי של הפרויקט
- פוליטיקה של בעלי עניין
- לחצים מסחריים

בסיום הקורס נחזור לרשימה  
זו ונבדוק האם רכשנו כלים  
מתאימים להתמודד עם  
בעיות אלה.

\* [Charette, R. N., Why Software Fails?, IEEE Spectrum, Vol. 42, Issue 9, Sept. 2005](#)

1

2

3

4

# מה מתוך הבאים כנראה אינו מהווה סיבה לאחוז הכישלון הגבוה בפרויקטי תוכנה?

1. מחסור במהנדסי תוכנה טובים
2. פיתוח תוכנה בשיטות הנדסיות לא מתאימות
3. חשיבותה של התוכנה לחברה האנושית
4. חוסר במעמד חוקי (רשיון) של מהנדס

# מה אפשר להספיק בסמסטר אחד?

- סילבוס...

- אתר ויומן הקורס

<https://github.com/jce-il/se-class/wiki>



# על הקורס

למידה באמצעות התנסות.

השתתפות, שותפות ומעורבות חיוניים להצלחה!  
קרוב לעולם האמיתי, ככל האפשר

**הסטרט-אפ הראשון שלכם ? Next Facebook**



# לא הכל נכנס, למשל:

[Martin](#): “School can teach the theory of computer programming. But school does not, and cannot teach the discipline, practice, and skill of being a craftsman.”

[Knuth Law](#): “Premature optimization is the root of all evil”  
[C2 Wiki](#): “Make It Work Make It Right Make It Fast”

- מבוא לתכנות+
- צוות וניהול
- תיכון מונחה עצמים
- – UML ומידול בכלל
- – Design Patterns
- מדידות וביצועים
- דיבוג
- אימות פורמלי
- שיוק \ סטרטאפים
- ...

# פרויקט הדגמה

- מרעיון למוצר
- מטרות להדגמה
- התרשמות מכמה רעיונות מרכזיים של הקורס
- הכוונה ראשונית לפרויקט\לרעיון שלכם
- מה אפשר להספיק בשעה?
  - מינימום קוד... (ובלי להבין הרבה...)
  - מקסימום התרשמות מתהליכים, שיטות וכלים

# פרויקט הדגמה – רעיון \ הצעה

- למה? (הבעיה)

לאחרונה בוטלה האפשרות להעלות קבצים למאגר קוד github ללא היכרות עם git – מקשה על מי שלא מכיר (עדיין) את הפרוטוקול ([Goodbye, Uploads](#))

- מה? (פתרון אפשרי)

מערכת שתפשט למשתמש את ההעלאה

- איך?

אפליקציה ברשת (למה?)

- שם הפרויקט (זמני): GitUploads



# GitUploads – איך מתחילים?

[Atwood](#): “a basic sketch of a homepage design is the first thing you should work on in any webapp, because it serves as the essential starting design document and vision statement.”

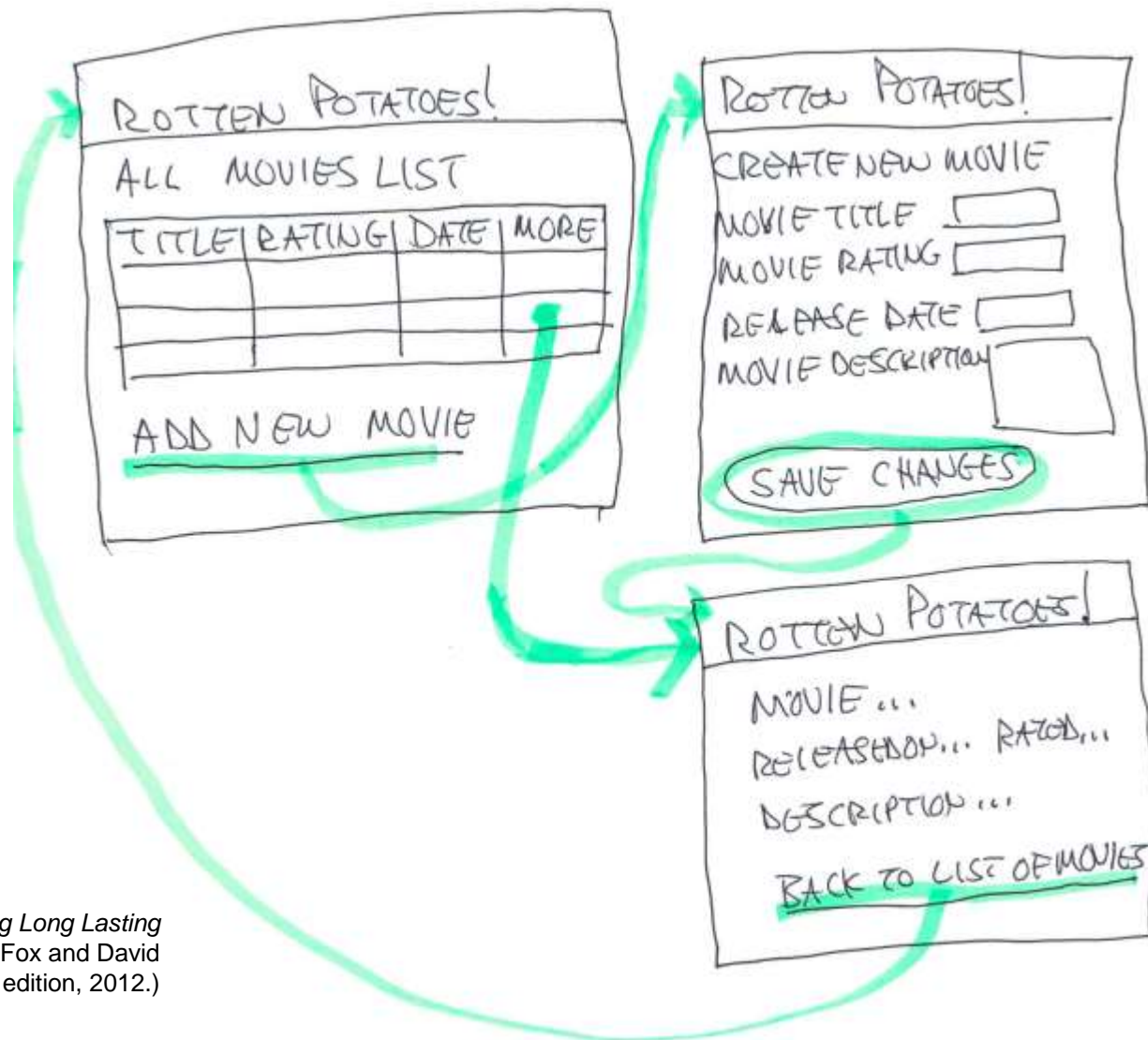
- יכולות ודרישות (מה)
  - זיהוי בעלי עניין ומשתמשים
  - תרשימי ממשק משתמש
  - איסוף וניתוח
- תיכון ראשוני \ ארכיטקטורה (איך)
- תהליך (כיצד)
  - תכנון הפרויקט
  - מימוש
  - משוב והתקדמות

# GitUploads – דרישות \ סיפורי

## משתמשים

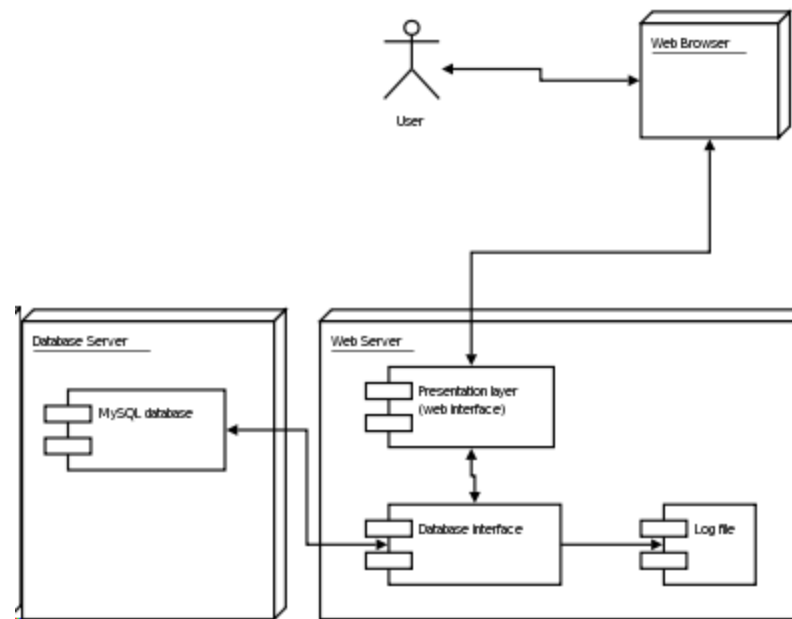
- **בתור** סטודנט שעדיין לא מכיר git **כדי** להעלות קבצים למאגר הקוד של הקורס כחלק ממשימה 1  
**אני רוצה** העלאה נוחה למאגר הקורס  
(בעלי עניין: סטודנט, צוות הקורס, github?, העלי באתר)
- **בתור** משתמש במאגרים שאינו מתכנת **כדי** שאוכל לעזור עם קבצים נוספים לפרויקט שאני חבר בו  
**אני רוצה** גישה להעלאת קבצים באמצעות ממשק דפדפן
- **מחיקה? ניהול? ...**

# GitUploads – תרשימי ממשק משתמש



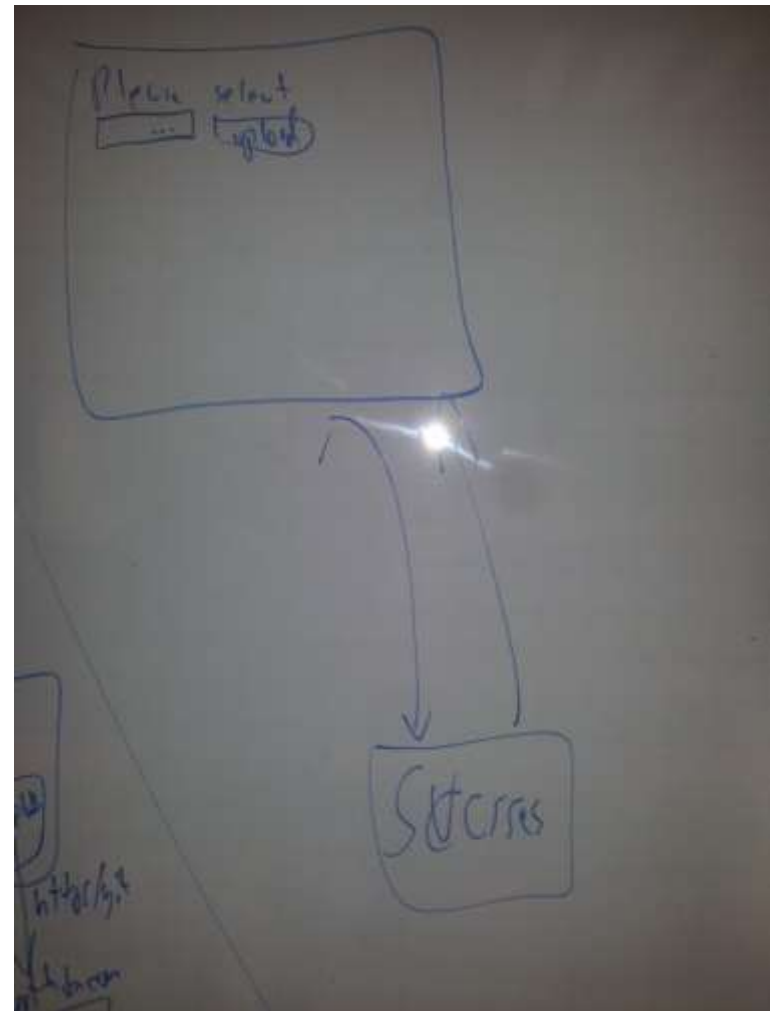
(Figure 4.4, *Engineering Long Lasting Software* by Armando Fox and David Patterson, Alpha edition, 2012.)

# GitUploads – תיכון ראשוני (UML)



[http://en.wikipedia.org/wiki/Applications\\_of\\_UML](http://en.wikipedia.org/wiki/Applications_of_UML)





# GitUploads – תכנון פרויקט

- צוות?
- משימות? הערכות? חלוקה ותיעדוף?
- הזדמנויות? סיכונים?
- אילו מסמכים? שיתוף קוד? הפצה?
- אתר ניהול פרויקט, למשל – [github.com](https://github.com),  
[PivotalTracker.com](https://pivotaltracker.com)
- הרצאות ומשימות 1+2

# GitUploads - תהליך ומשוב

- אז מאיפה מתחילים? MVP
- מי קובע מה עושים?
- לפי אילו שלבים לעבוד?
- לכל שלב יעדים \ קלטים \ פלטים
- איך נדע שסיימנו?
- הרצאות 3-5

# GitUploads - ניתוח תרחיש ומימוש

- בחירת סיפור בכל הערך הגבוה ביותר: נניח 1 לעיל (איך מחליטים? מי קובע?)
- מאפיין: העלאת קובץ
- תרחיש:

– **בהינתן** שאני גולש לאפליקציה

– **כאשר** אני בוחר קובץ ומיקום במאגר

– **אזי** המאגר מתעדכן עם הקובץ

- יכול לשמש לאימות ה**מימוש** (מסמטר קודם, התנצלות...)

<https://gist.github.com/robi-y/5081544>

# GitUploads - תשתיות

- עזרה מסביבת פיתוח, תוספים וספריות (RAD),  
למשל MS Visual Studio / Eclipse
- בקרת גרסאות (git)
- שימוש בתבניות ידועות (MVC)
- מונחה בדיקות (כלי בדיקות יחידה MsTest/Moq)
- הפצה מתמשכת (לענן, למשל [azure](#), Heroku, Google AppEngine)
- הרצאות 5-8

# GitUploads – עקרונות פיתוח

- שימוש ברכיבים קיימים
  - ספייק (מחקר \ שיטוט...)
  - מנהל חבילות\תלויות (למשל maven, nuget)
- תיכון מפורט
  - כללי קידוד, למשל Don't Repeat Yourself (partial view)
  - עקרונות OOD, למשל Dependency Inversion Principle, Single Responsibility
  - תבניות תיכון, למשל [Repository Pattern](#)
  - בדיקות יחידה ו- TDD (**Red-Green-Refactor**)
- עוד: UX חווית משתמש (למשל עדכון עם ajax), משוב (למשל UserVoice)
- הרצאות 9-14

# GitUploads - Review

- תהליכים: תכנון, דרישות, תיכון, בדיקות, פיתוח איטרטיבי ואנכי, משוב, ...
- שיטות: פיתוח מונחה בדיקות, תיכון מונחה עצמים, חווית משתמש, ...
- כלים: ניהול פרויקט, בקרת גרסאות, כלי תיכון, סביבות פיתוח, בדיקות, הפצה, פורומים\חיפוש, ...
- אנשים: צוות, ניהול, סקרים, הצגה, ...
- המטרה: תוכנה איכותית
- **האם אפשר בלי כל זה?**



# GitUploads - רטרוספקטיבה

- מה הלך טוב?
- מה פחות וניתן לשפר?
- מה נשנה לפעם הבאה?



# GitUploads – כלי הנדסת תוכנה

- Compiler, IDE+, Version Control, Package, UML
- Testing, Building, Integration, Deployment (Cloud)
- And more: Code Review, UserVoice, A/B testing
- Web: Google, stackoverflow,...

# בקרת גרסאות הינה:

1. תהליך לפיתוח תוכנה
2. עזר לשיתוף פעולה בין מפתחים
3. כלי לגיבוי קבצי קוד
4. תשובות 2 ו-3 נכונות

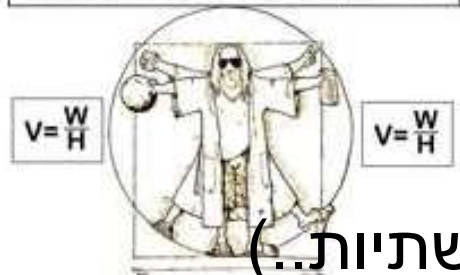
# הפרויקט - מטרות

- נסיון ישיר עם חומר הקורס
- אתגרים טכניים בשל גודל הפרויקט
- אתגריים חברתיים במסגרת מאמץ קבוצתי
- הזדמנות להתנסות בסביבה חדשה, למשל: התקנים ניידים, Web, .net, Ruby, APIs, קוד פתוח, רשתות חברתיות, דרייברים, תוסף דפדפן, [html5](#) ....  
(בכל זאת מומלץ RAD)
- הזדמנות עסקית (זכויות יוצרים!)
- **המלצה: לא לפתוח הרבה חזיתות!**
- **מה בכל זאת שונה מהתעשייה?**



# פרויקט קבוצתי – מהיכן מתחילים?

Dude's Law: Value = Why / How



- כל זוג מגישים **רעיון** \הצעה לפרויקט

- בעיה (למה?)

- הצגה פונקציונלית (מה?)

- פתרון \ תיכון ראשוני (איך?) (מחשבים \ תשתיות...)

- ייתרונות, ישימות וסיכונים (האם?)

- מסמך ומצגת

- בהרצאה הבאה:

- מצגות

- <- צוותים

- פירוט בויקי+מסמך

# מאיפה אקח רעיון?



nhprman  
@nhprman



Follow

"All startups should be thinking: what frustrates me and how do I make it better?"  
Sir Richard Branson

← Reply ↺ Retweet ★ Favorite

5:01 PM - 9 Aug 12 - Embed this Tweet

Eric Raymond , The Cathedral and the Bazaar:  
**"Every good work of software starts by scratching a developer's personal itch."** [link](#)

Ed Catmull, Pixar: "If you give a good idea to a mediocre group, they'll screw it up. **If you give a mediocre idea to a good group, they'll fix it.** Or they'll throw it away and come up with something else. "

<http://www.youtube.com/watch?v=k2h2lvhzMDc>



• בוער לי

• חיפוש וסיעור מוחות

• משהו מעניין



# התרשמות מקורסים קודמים

- 2012/3 – <https://github.com/jce-il/se-class/wiki/PastProjects>
  - Example: [Picture Barcode Generator](#)
- 2011E - [https://bitbucket.org/robi\\_y/se11b/wiki/Projects](https://bitbucket.org/robi_y/se11b/wiki/Projects)
  - Example: [Car Pool](#)
- 2011 - [https://bitbucket.org/robi\\_y/se11a/wiki/teams](https://bitbucket.org/robi_y/se11a/wiki/teams)
- 2010 - [https://bitbucket.org/robi\\_y/se11a/wiki/teams](https://bitbucket.org/robi_y/se11a/wiki/teams)
- 2010E - <http://sejce2008.codeplex.com/> (down right at the main page)
- 2009 - <http://my.jce.ac.il/~robi/jce2008/jce2008-fogbugz/jce2008-old.fogbugz.com/default68ca.html> <http://jproject-2/fogbugz2008> (במכללה)
- 2008 -
  - CrazyLinks - <http://sites.google.com/site/crazylinksproject/Home>, ThinkFast - <http://sites.google.com/site/thinkfastgame/>, FastRegister - <http://groups.google.com/group/fast-register?hl=en>, GoogleIt - <http://sites.google.com/site/searchitproject/>, SharePaint – <http://www.freewebs.com/sharepaint>, ReadItUp – <http://readitup.hopto.org/>

**DONE IS  
BETTER  
THAN  
PERFECT**

# עוד רעיונות

- Open Source Projects (github, bitbucket, source forge, codeproject, codeplex, ....), AppStores
- Project pages of a similar course [1](#) [2](#) [3](#)
- [startupisrael.com](http://startupisrael.com), [Start-ups Israel news](#) ([student example](#)), [ynet news](#)
- <http://startupweekend.org>
- מרכזי חדשנות: [טכניון](#), [בין-תחומי](#)
- [google summer of code](#), [2012](#)
- Competitions: [MS imagine cup](#), [RailsRumble](#)

# סיעור מוחות - קישורים

- Ries, [The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses](#)
- Savoia, [Pretotyping](#) ([book](#), [video](#))
- McConnel, GTAC 2011: [Closing Keynote - Secrets of World Class Software Organizations](#), Video 2011
- Copeland, [Innovation at Google](#), Video 2010
- [The Myth of Innovation](#)
- [Graham: Ideas for Startups](#)
- [The Naming Game: Things to consider when naming an open source project](#)



# נושאים נוספים למבוא

- [אתיקה למהנדס תוכנה](#)
- הנדסת תוכנה מול  
הנדסות אחרות \ מדעי המחשב \ ייצור
- הנדסת מערכות
- [מבחן ג'ואל](#) (או איך לבחון את מקום עבודה)
- חדשנות, שיווק
- איך להיות מהנדס טוב \ הכנה לתעשייה
- [כישלונות ידועים בתוכנה](#)



# ההרצאה היום עזרה לי להבין ש:

1. הצילו, איך בורחים מכאן?
2. מצטער, הרבה דברים עדיין לא ברורים לי
3. דווקא נחמד, מקווה שאעבור את זה איכשהו
4. אין, חייב כבר להתחיל לעבוד על הפרויקט שלי

# בפעם הבאה

- תיאוריה: תהליכים, מודלים ומתודולוגיות  
(הכנה: [wikipedia sdp](https://www.wikipedia.org/wiki/SDP) [פיתוח תוכנה זריז](#))
- משימות
  - רישום לפורום, ל-github וקבלת הרשאות במאגר הקורס
  - רעיון\הצעת פרויקט (בזוג) : מסמך ומצגת, רישום להצגה
- סקר רעיונות

# לסיכום

- דיון בהנדסת תוכנה, תוכנה היום, אתגרים
- מטרת הקורס: מתכנת  $\leq$  מהנדס תוכנה
- המיוחד בקורס
  - רב תחומי
  - הזדמנות לעבוד על רעיון שלכם
  - בד"כ אין תשובה אחת נכונה, מותר לטעות
  - כישורים "רכים" (יצירתיות, שיתוף פעולה)
  - לא קשה, אבל עבודה די רבה (ומהתחלה!)
- תוכן הקורס: תהליכים, דרישות, תיכון, בדיקות, מימוש, כלים ועוד.....
- שאלות \ הבהרות \ הצעות ?

## בהצלחה ובהנאה

