תוכנה 1 בשפת Java

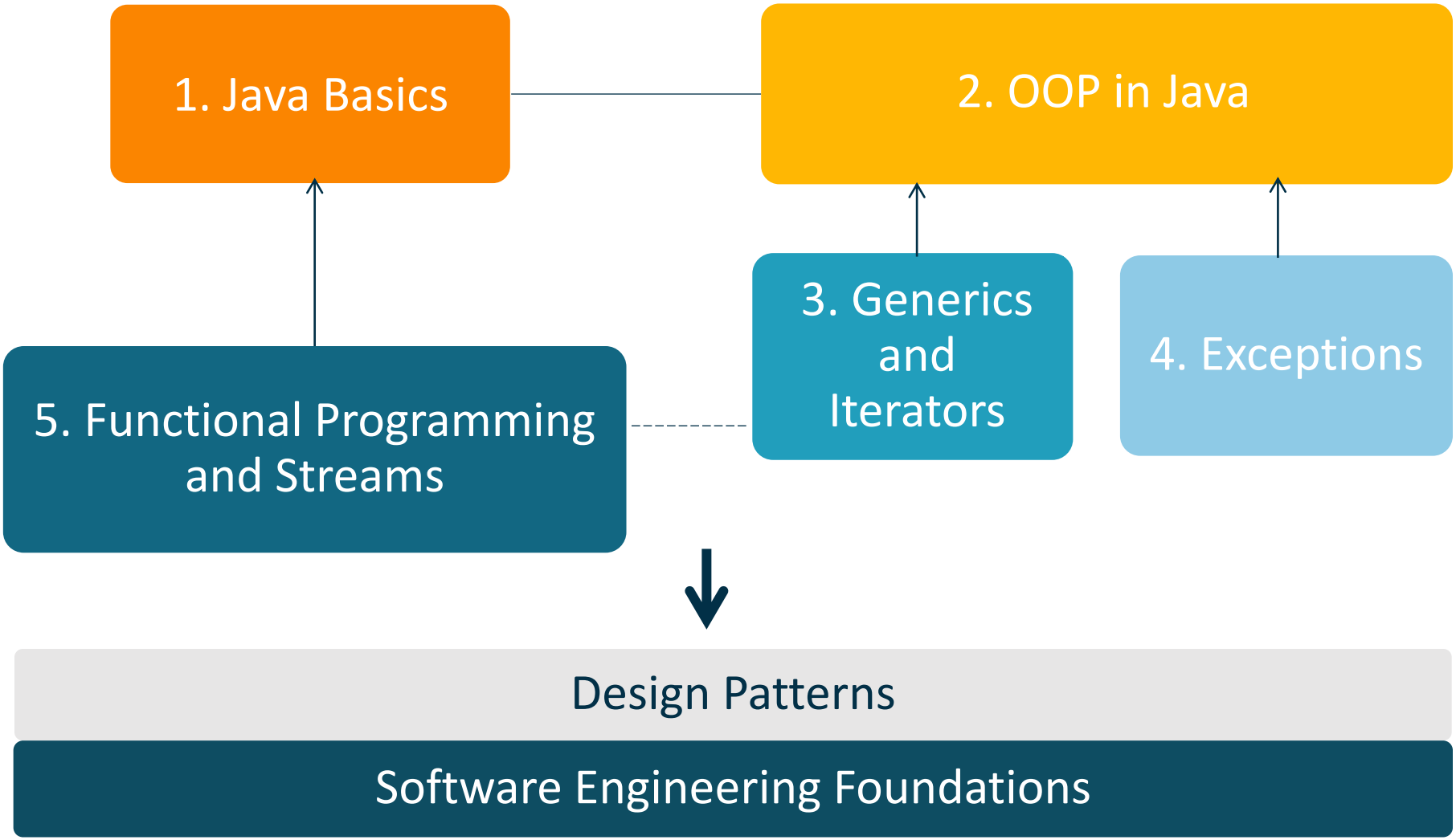
שיעור מספר 10: הנדסת תוכנה ותהליכי פיתוח תוכנה

מיכל קליינבורט

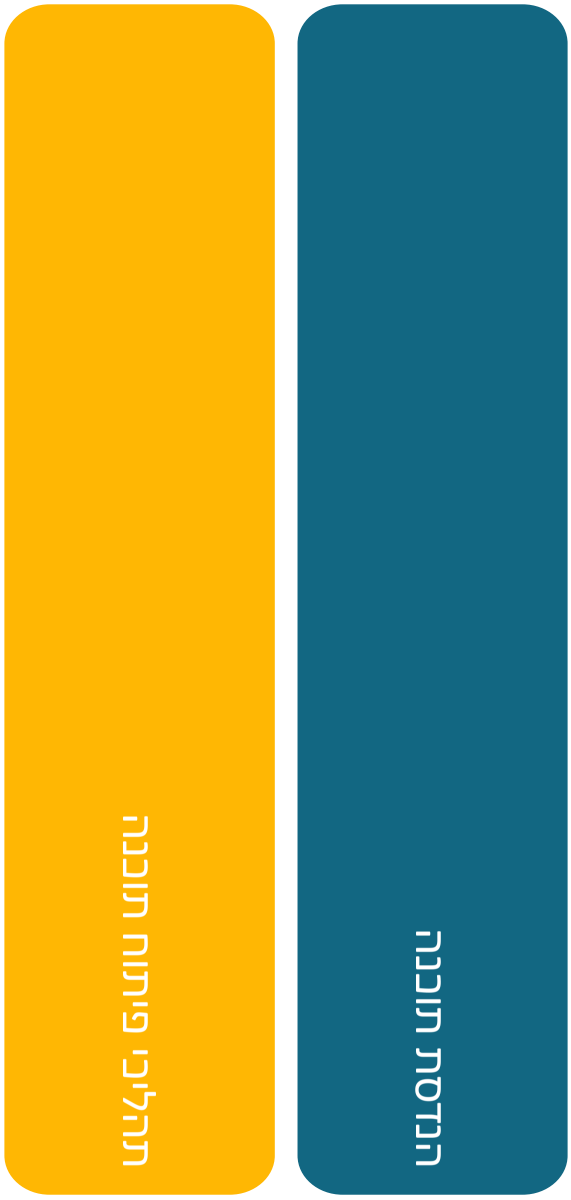
בית הספר למדעי המחשב

אוניברסיטת תל אביב

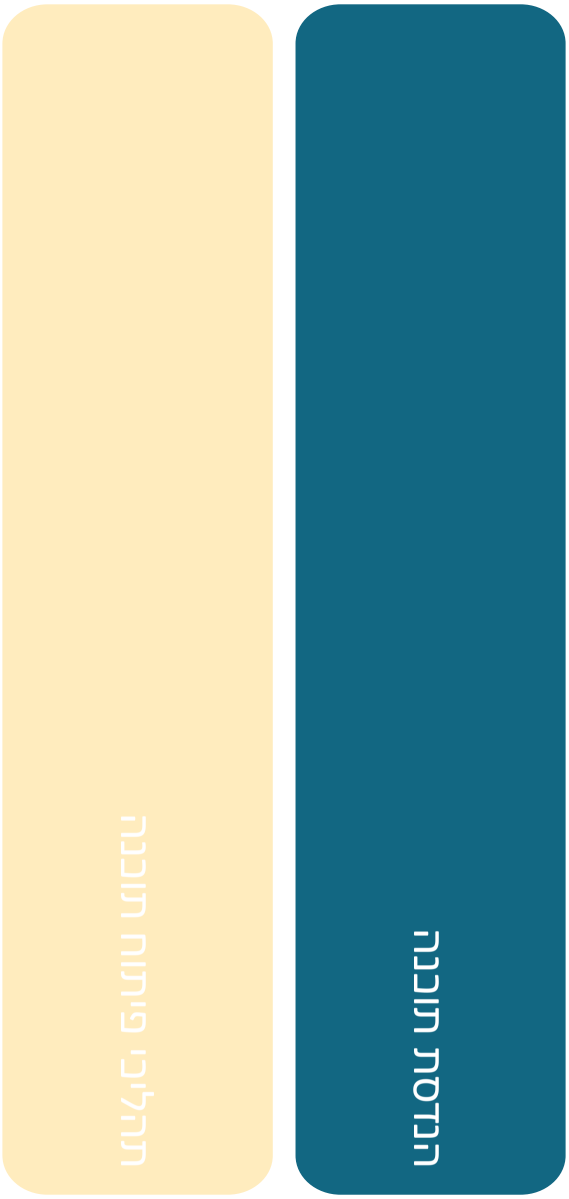
נושאי הקורס


1. Java Basics
2. OOP in Java
3. Generics and Iterators
4. Exceptions
5. Functional Programming and Streams
Design Patterns
Software Engineering Foundations

התוכנית להיום



התוכנית להיום



מה זו הנדסת תוכנה?

- תחום במדעי המחשב
- אוסף של מתודולוגיות, שיטות וכלים לבניה של תוכנה איכותית בצורה יעילה
- סוג של הנדסה?

- לפי ויקיפדיה:

> The term *engineering* is derived from the Latin *ingenium*, meaning "cleverness" and *ingeniare*, meaning "to contrive, devise".[3]

האם הנדסת תוכנה היא הנדסה?

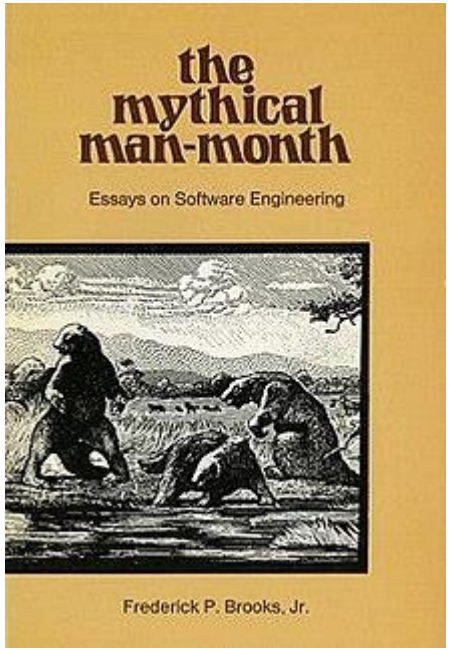
- מה שאנו מכנים "הנדסת תוכנה" הוא אוסף של כלים, טכנולוגיות ושיטות ניהול
- זה עדיין רחוק מתחומי הנדסה אחרים
- בדי להתקדם לשם צריך:
  - להגדיר את הידע הנחוץ למומחים, להפוך אותו לנגיש, לעודד שימוש חוזר (ספריות, design patterns), התמחויות מקצועיות, לשפר את החיבור בין מדע לשימושים מסחריים (העברה מתאוריה לפרקטיקה)

פרויקט בתוכנה כפרויקט הנדסי

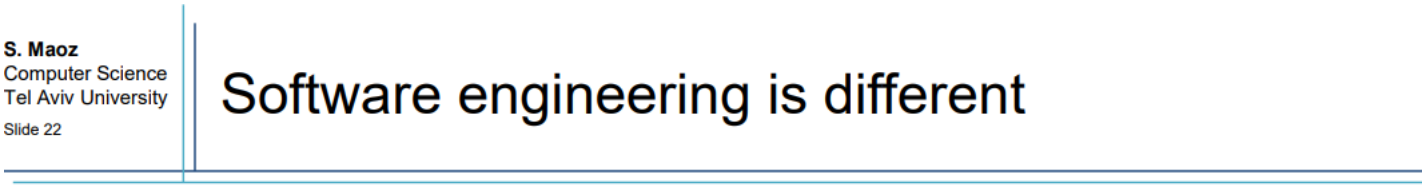
- פרויקט בתוכנה הוא פרויקט הנדסי מורכב שמערב גורמים רבים (תוכנה היא רק חלק ממנו):
  - לקוחות
  - משתמשים
  - ספקים שונים (צד שלישי, שרתים)
  - בודקי תוכנה
  - צוותי פיתוח
  - תקציב

The Mythical Man-Month

- אחד הספרים המוכרים שעוסקים בסוגיות של הנדסת תוכנה
  - ובפרט, בניהול פרויקט תוכנה
- נכתב ע"י Fred Brooks בשנת 1975
- כמה מהתובנות של המחבר:
  - הוספה של כח אדם לפרויקט תוכנה שמתעכב תגרום לדחיה נוספת בפרויקט
  - במערכת מורכבת יש מספר מינימלי כלשהו של באגים שלא ניתן לצמצם. כל נסיון להעלים שגיאות עלול ליצור שגיאות אחרות חדשות.



אז מה מייחד הנדסת תוכנה מהנדסה אחרת?



Software is different than other engineering disciplines

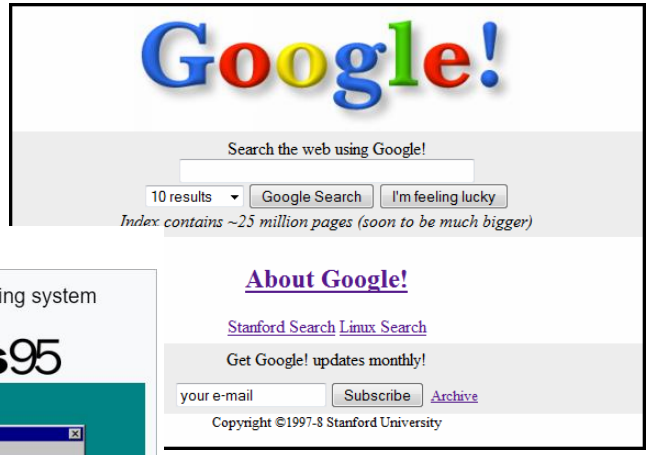
- No fundamental theory
- Ease of change
- Rapidly evolving technology
- Negligible manufacturing cost
- No borders

לקוח מהקורס "נושאים בהנדסת תוכנה" של פרופ' שחר מעוז

אז מה מייחד הנדסת תוכנה מהנדסה אחרת?

- תוכנה דורשת יכולת עדכון , גמישות

1995-1998



2021-2024

Credits:
https://en.wikipedia.org/wiki/Windows_11
https://en.wikipedia.org/wiki/Windows_95
https://www.pingdom.com/blog/googles-first-website-from-10-years-ago/

אז מה מייחד הנדסת תוכנה מהנדסה אחרת?

- הפרדיקטביליות (יכולת החיזוי) היא לא טובה
  - קשה לדעת כמה זמן ייקח פרויקט תוכנה, מתי יסתיים, כמה אנשים צריך. יש פערים עצומים בין מתכנתים שונים
- הרבה שאלות שנרצה לשאול על התוכנה שכתבנו הינן שאלות לא חשיבות (בעיית העצירה, למשל)

Programs vs. software products

S. Maoz
Computer Science
Tel Aviv University

Programs vs. software products

Programs	Software products
Typically small	Large
Author is sole user	Large number of users
Single author	Team of developers
No proper user interface	Well-designed interface
No proper documentation	Well documented
Ad hoc development	Systematic development

התוכנית להיום

הנדסת תוכנה

תהליכי פיתוח תוכנה

תהליך פיתוח תוכנה

- מתאר את הדרך מרעיון למוצר
- לחלק את המורכבות בצורה שיטתית
- פורמלי או לפחות semi-פורמלי
- ישנם כמה מודלים לתהליכי פיתוח:
  - Waterfall : מתאר מעבר לינארי משלב אחד לאחר, כמו במפל
  - Evolutionary prototyping: נתחיל מאפיון ראשוני ונפתח אותו בהתאם למשוב מהלקוח
  - Agile: משפחה של שיטות שבכולן השאיפה היא להיות גמיש ולקחת בחשבון שיהיה שינויים
  - ועוד...

שלבים מסורתיים בפיתוח תוכנה

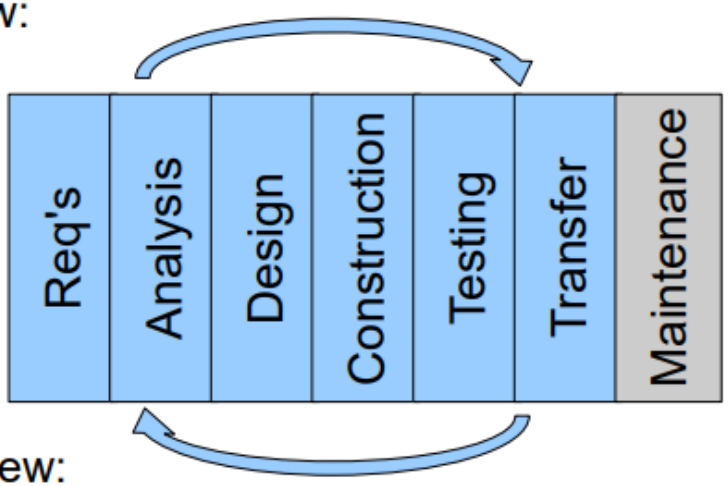
- **דרישות**: לברר מול הלקוח ולהבין איזו מערכת עלינו לבנות
- **Design**: תכנון מבנה התוכנה (בצורה שהיא high level או מופרטת יותר)
  - Architectural design, Interface design , Component design, Data structure and algorithm design
- **מימוש**: כתיבת הקוד ומימוש ה design
- **וריפיקציה ובדיקת הקוד**: עלינו לוודא שהקוד מתנהג כפי שתכננו
  - did we build the system right? Does the system actually implements the specification? did we build the right system? Did we build the system that the customers want?
- **Maintenance**: אחרי ה release, תיקון באגים שמצאו משתמשים, התאמה לשינויים בסביבה (מערכות הפעלה חדשות, ספריות חדשות) , הוספה של features. תחזוקת המערכת בשהיא מתפתחת.

Maintenance

S. Maoz
Computer Science
Tel Aviv University

Maintenance: textbook vs. reality

Textbook view:



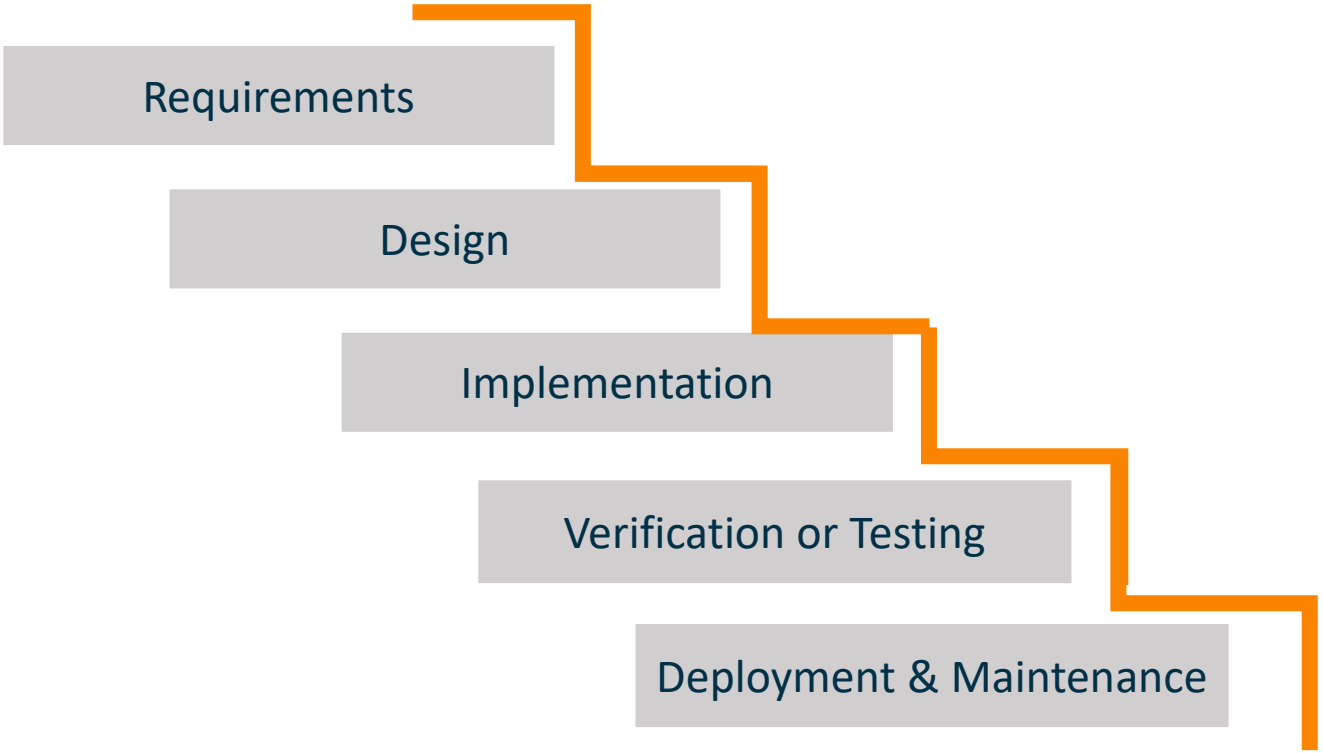
Real-world view:

Software lifecycle model

מתאר ביצד יש לארגן את שלבי הפיתוח מתחילת ועד סוף הפרויקט:

- לקבוע את סדר השלבים
- לקבוע מה הקריטריון לעבור לשלב הבא

מודל המפל Waterfall



הופיע לראשונה במאמר מ 1970 מאת Royce

הרעיון שעומד מאחורי מודל המפל

- קיימת הנחה מסורתית ובסיסית שעלות השינוי גדלה אקספוננציאלית כתלות בזמן
- אם זה נכון, כדאי לזהות שגיאות בשלבים מוקדמים ולכן להשקיע הרבה בתכנון מראש
- קל יותר לשנות דרישות ומודלים מאשר קוד, ולכן נשקיע בהם זמן ואנרגיה, ורק לאחר שיבדקו נעבור לשלב הפיתוח

מודל המפל: דיון

- מתי מודל זה עובד טוב?
  - כאשר יש הגדרה יציבה של המוצר
  - כאשר התחום מובר וידוע היטב והטכנולוגיות מוכרות ומובנות
- יתרונות עיקריים:
  - המודל עוזר למצור שגיאות בשלב מוקדם
  - קל לבנות צוות
- חסרונות עיקריים:
  - חוסר בגמישות
  - קשה ליישום כאשר הדרישות חלקיות או צפויות להשתנות, כאשר המפתחים לא מומחים בתחום, כאשר הטכנולוגיות חדשות ומתפתחות
- קיימים שיפורים למודל (למשל: לעבור על התהליך פעמיים)

מודל המפל: האם מתאים לעולם האמיתי?

לא ממש מתאים לרוב פרויקטי התוכנה בעולם האמיתי

- לא סביר שנעשה הכל נכון
- בדיקות מגיעות בשלב מאוחר ועלולות לחשוף בעיות בתכנון במקורי
- לקוחות לא תמיד יודעים מה הם רוצים: הדרישות והאילוצים עלולים להשתנות שוב ושוב
- קשה למפות את כל האפשרויות מראש
- פרדיקטביליות נמוכה יוצרת delay, נוצרים צווארי בקבוק
- הרבה פעמים התוכנה לא עושה את מה שנדרש ממנה: בשלב התכנון חשבנו משהו אחד ובפועל הלקוחות משתמשים אחרת. יש Waste: עשינו עבודה לשווא. זה יקר כי המהנדסים יקרים.

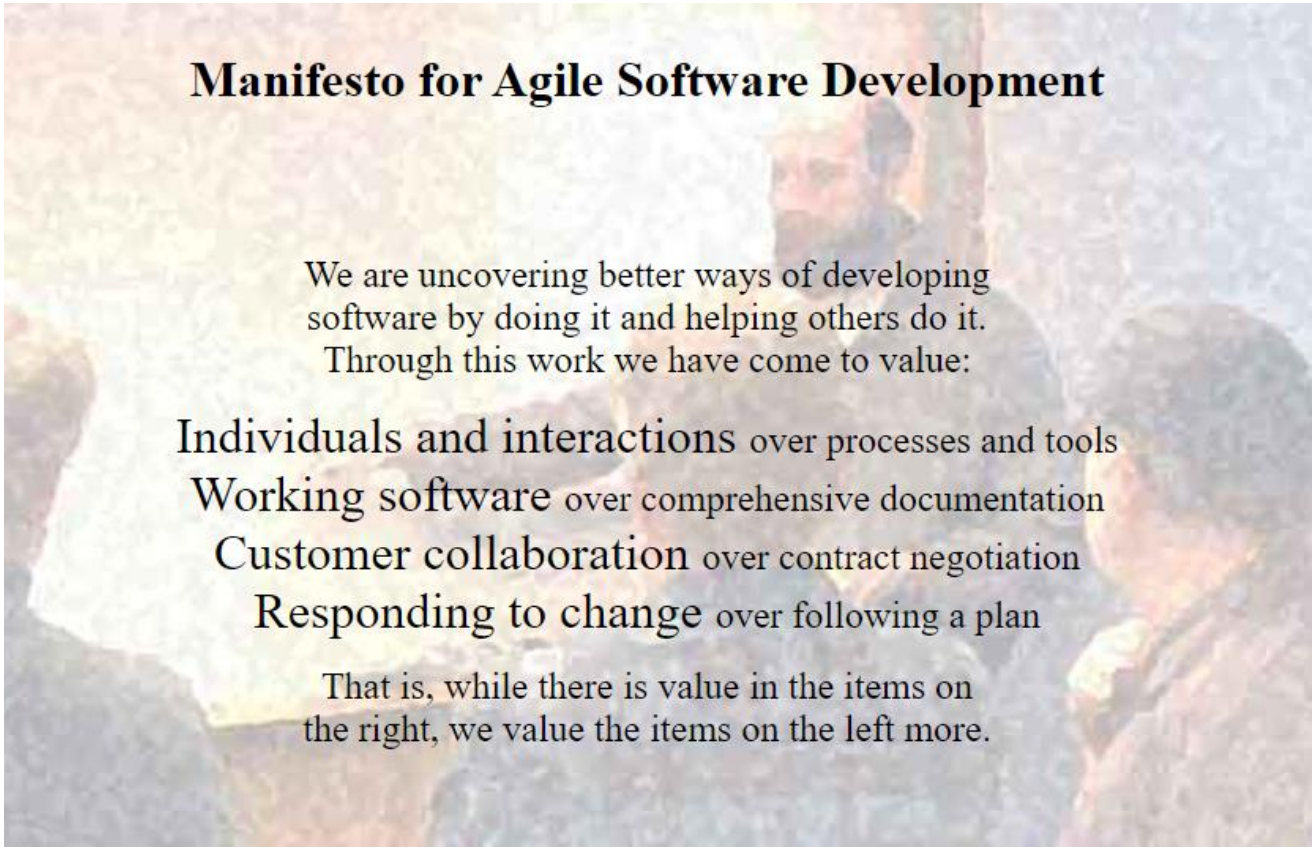
Agile Software Development

- גישה מודרנית לפיתוח תוכנה
  - הרבה השתנה בעשורים האחרונים: cloud computing, מחשבים נהיו מהירים וזולים, קיימות הרבה שפות תכנות מודרניות ונוחות, כלי פיתוח טובים, כלי ניהול גירסאות ועוד.
    - כלומר: **הרבה יותר קל לעשות שינויים מאשר בעבר**
- תכנון רב מראש עלול להיות מעמסה
- אם יש אי בהירות לגבי הדרישות אז עדיף לדחות
- רוצים לצמצם Waste

- מקורה של הגישה ב- 17 מהנדסי תוכנה

The Agile Manifesto (2001)

https://agilemanifesto.org/



The Agile Manifesto (2001)

https://agilemanifesto.org/

עקרונות

- Principles:
  - Focus on code rather than on design (to avoid unnecessary docs)
  - Value people over process (developers, customers)
  - Iterative approach (deliver working solution quickly, change quickly)
  - Customer involvement (feedback throughout the process)
  - Expecting that requirements will change
  - Mentality of simplicity (as simple as possible)

(slide taken from Prof. S.Maoz)

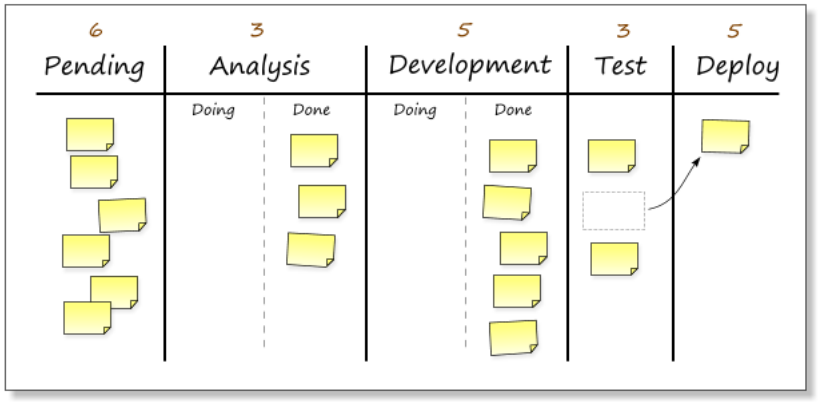
Test-Driven Development (TDD)

- חלק מגישת ה Agile
- שלושה שלבים עיקריים:
  - להתחיל מכתיבת טסטים שמקודדים את הדרישות. הם יכשלו.
  - לכתוב מספיק קוד כדי לגרום לטסטים הללו לעבור בהצלחה
  - לשפר את הקוד שכתבנו כדי להפוך אותו לאיכותי יותר (refactoring, יכול גם לערב שינוי דיזיין)
- אחר כך ממשיכים בתהליך עבור סט דרישות אחר

Kanban: a popular instance of Agile

יש גישות שונות שממממשות Agile , אבל זו אחת הפופולריות

- שיטה יפנית שהתפרסמה בעולם בזכות Toyota (Just-in-time production)
- הרעיון: מחלקים את העבודה לשלבים, משימה עוברת בין השלבים. בכל שלב יש מכסה מקסימלית של משימות שיכולות להיות באותו שלב בו זמנית.



- התמונה והסבר נוסף נמצאים ב: https://kanbanblog.com/explained/

Kanban: a popular instance of Agile

- ואם נוצר עומס בשלב מסוים?
  - יש חברות שדואגות לחזק את הצוותים העמוסים בתקופות העומס, וכך להגדיל את המכסה. אנשים עוברים מצוות לצוות , צוותים נהיים ורסטיליים.
- גישה זו מאפשרת למנהלים כל הזמן לדעת מה קורה
- הרבה חברות עובדות כך