# פיתוח מערכות תוכנה מבוססות Java

אוהד ברזילי

אוניברסיטת תל אביב

"אפילו מתכנתים יכולים להיות אנשים שלמים בעולם האמיתי. XP הוא הזדמנות לבחון את עצמך, להיות עצמך, להבין שאולי היית בסדר כל הזמן ופשוט הסתובבת עם האנשים הלא נכונים..."

Kent Beck, eXtreme Programming Explained

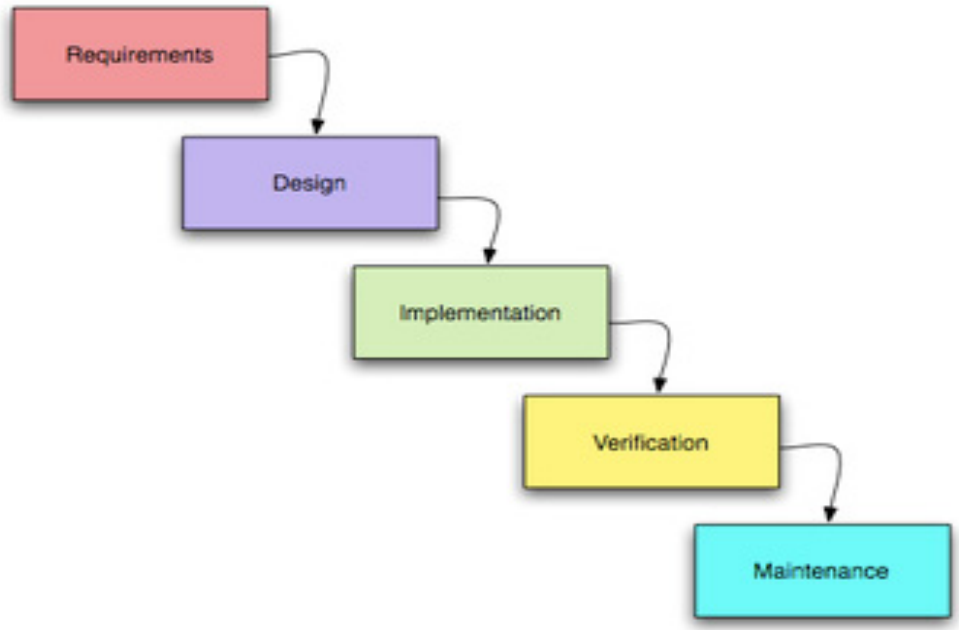
# מחזור החיים של תוכנה

- ניתוח דרישות (requirements analysis)
- תיכון (design)
- מימוש (Construction, implementation or coding)
- שילוב (integration)
- בדיקות וניפוי שגיאות (Testing and debugging aka: verification)
- בדיקות קבלה
- ייצור (production)
- הפצה והתקנה (deployment and installation)
- תחזוקה ושינויים (maintenance)

- התייחסות מיוחדת למקרה שמערכת התוכנה היא חלק ממערכת ממוחשבת הכוללת חומרה ותוכנה

# מודל המפל

- המודל המסורתי של מחזור חיים נקרא מודל מפל המים (waterfall model, Royce 1970) - כל שלב מתבצע לאחר שקודמו הסתיים (אך ניתן לחזור לשלב קודם לצורך תיקון).



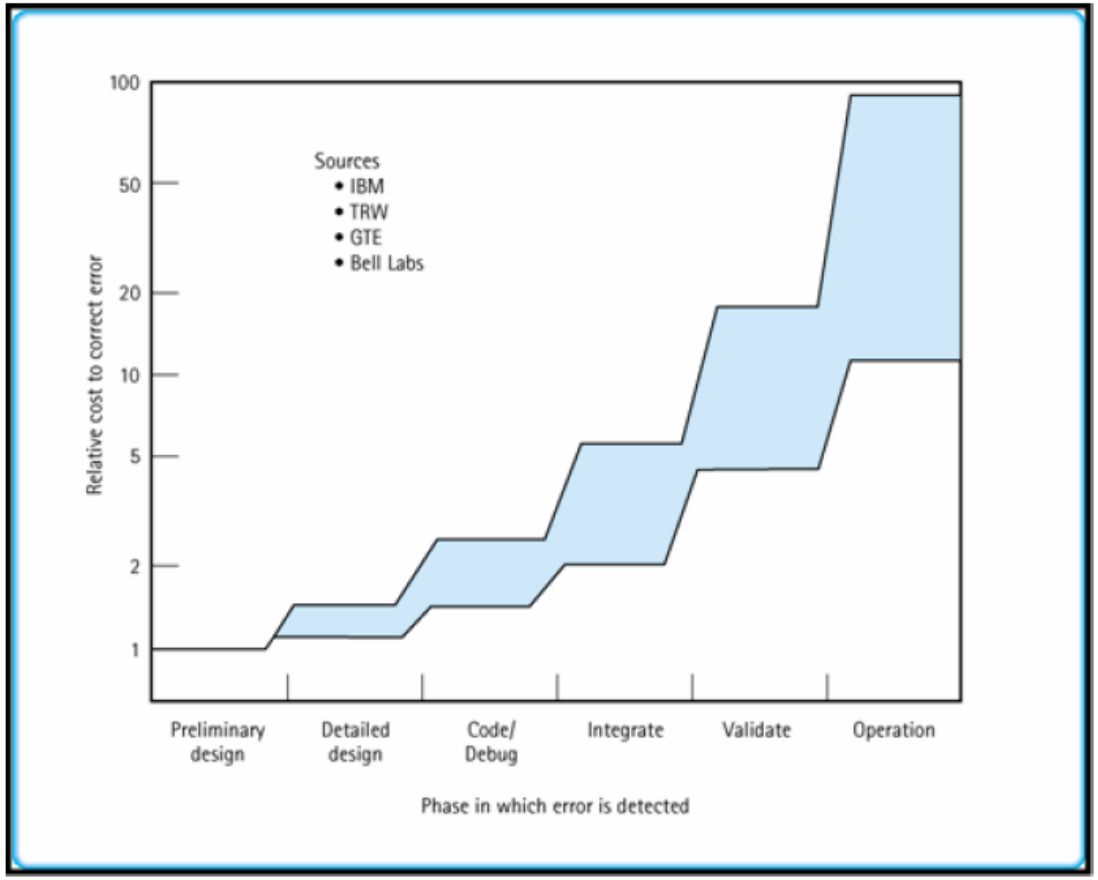
# מחירן של טעויות

- **ככל שטעות מתגלה מוקדם יותר, מחיר תיקונה קטן יותר**

- נניח שטעינו בניתוח הדרישות ושכחנו פעולה מסוימת שהתוכנה צריכה לבצע
  - אם נגלה את הטעות לפני המעבר לתיכון, המחיר יהיה מינימאלי, אולי עיכוב קטן בלוח הזמנים
  - אם נגלה בזמן התיכון, נצטרך אולי לזרוק חלק מהתיכון שלא יתאים לדרישות המתוקנות
  - אבל אם נגלה את הטעות רק בזמן בדיקות הקבלה, נצטרך אולי לזרוק חלקים גדולים מהתיכון ומהמימוש!

- עדיף לגלות טעויות מוקדם; לשם כך צריך לתכנן בקפדנות את תהליך הפיתוח הכולל, ולהשתדל להשתמש בשיטות שימזערו טעויות ואת הצורך לחזור אחורה לשלב קודם

# מחירן של טעויות



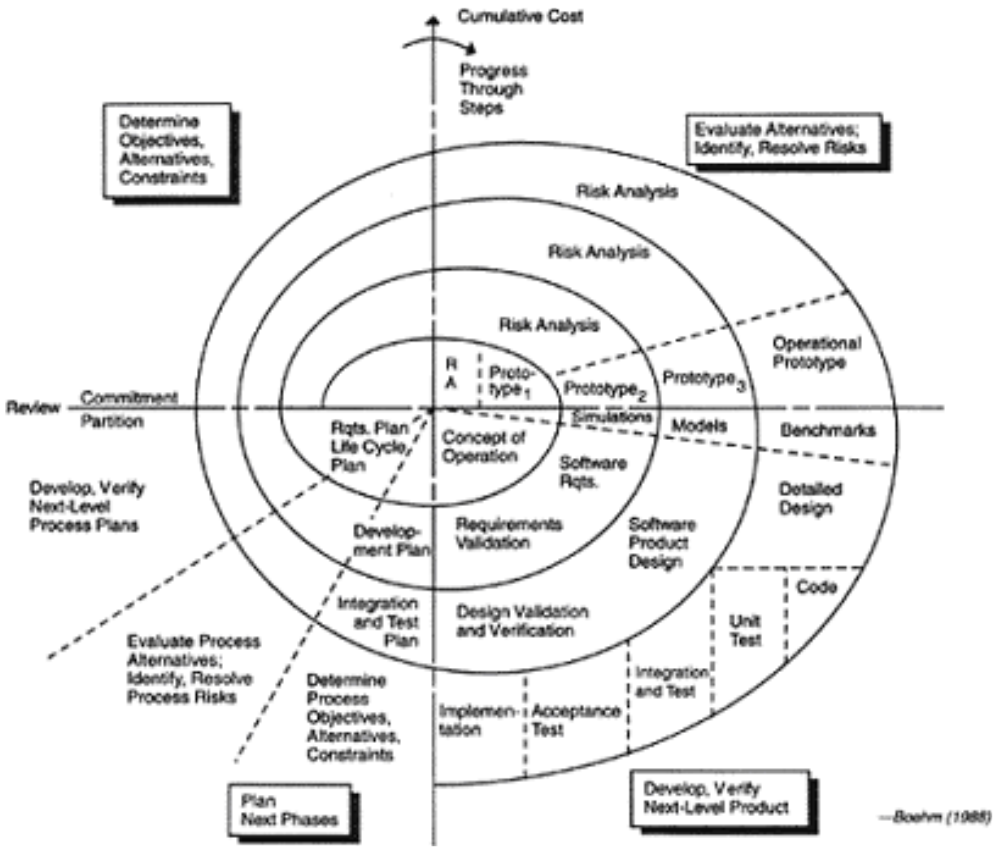
# שינוי

Everything in software changes. The requirements change. The design changes. The business changes. The technology changes. The team changes. The team members change.

The problem isn't change, because change is going to happen; the problem, rather, is our inability to cope with change.

Kent Beck, eXtreme Programming Explained

# מודל הספירלה




- מודל הספירלה ( spiral model) שהוצע מאוחר יותר (Barry Boehm, 1988) מפתח את המערכת באופן אבולוציוני.

- מתחילים מפיתוח מערכת מינימלית, ומבצעים את כל השלבים. לאחר סיום מעריכים את המוצר הנוכחי, מחליטים מה להוסיף, וחוזרים על כל השלבים

# מפל או ספירלה?

- מודל הספירלה מאפשר לראות מוצר חלקי ולהעריך אותו
- אבל מפל המים משקף את הרצוי:
  - רצוי לא לטעות
  - רצוי לדעת הכל מראש

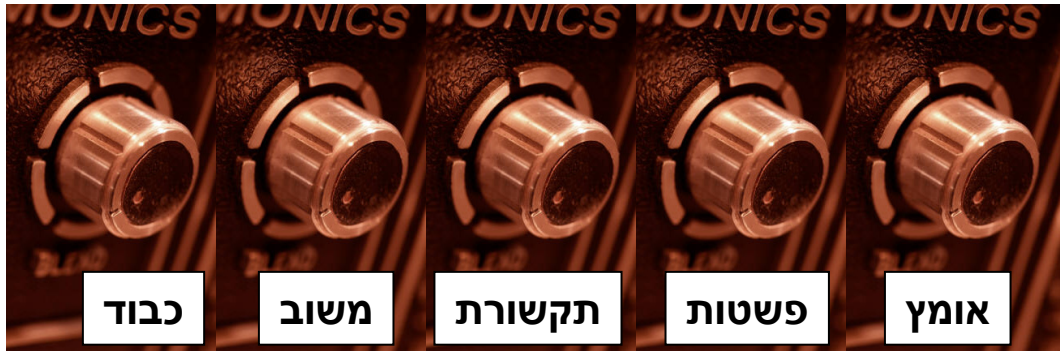
- קיימים גם מודלים אחרים לתהליך הפיתוח:
  - בשנים האחרונות עולה הפופולריות של משפחת המודלים הקלילה (agile)
  - הנציג הבולט של המשפחה הזו הוא e**X**treme **P**rogramming (תכנות קיצוני, XP)

# תכנות קיצוני (XP)

- מעצבי השיטה (Kent Beck, Ward Cunningham, Ron Jeffries, 1996) ניסחו 5 ערכים:
  - משוב (feedback)
  - פשטות (Simplicity)
  - תקשורת (Communication)
  - אומץ (Courage)
  - כבוד (Respect)

- מכיוון שהערכים הוכחו כטובים, נלקח כל אחד מהם לקיצוניות

# ערכים, עקרונות, מיומנויות



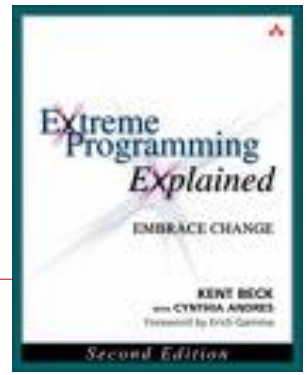

- ערכים אלו כלליים ויש לגזור מהם **מיומנויות**, דברים פשוטים שנוכל ליישם ברמה הטכנית
- חשוב להבין את **העקרונות** שלפיהם נגזרו המיומנויות כדי שנוכל להתאים את השיטה לצרכנו (להוסיף או להסיר מיומנויות)

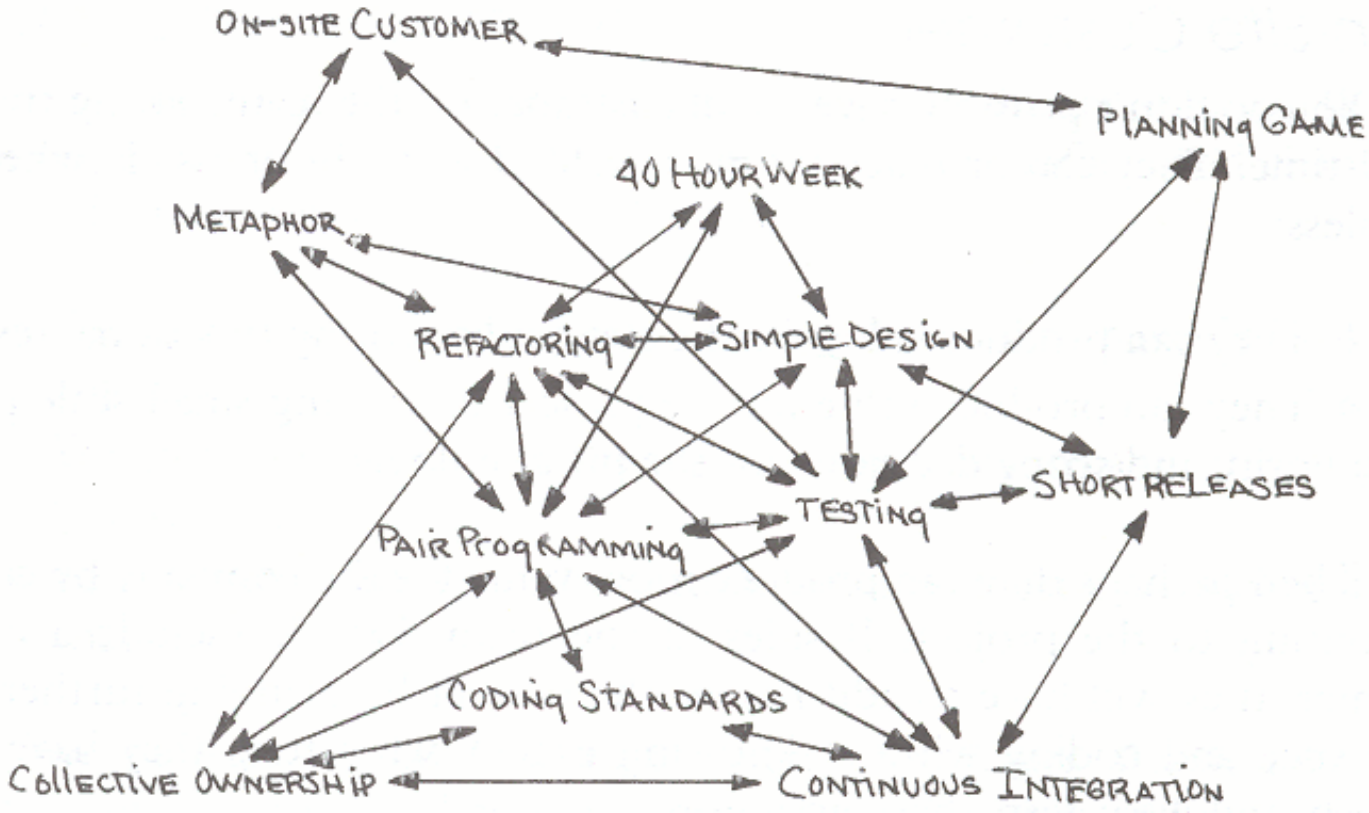
# העקרונות

- אנושיות (humanity)
- כלכלה (economics)
- רווח הדדי (mutual benefit)
- דמיון עצמי (Self Similarity)
- שיפור (Improvement)
- שוני, מגוון - (diversity)
- הרהור (reflection)
- זרימה (flow)
- הזדמנות (opportunity)
- יתירות (redundancy)
- כשלון (failure)
- איכות (quality)
- צעדי תינוק (baby steps)
- קבלת אחריות ( accepted responsibility)

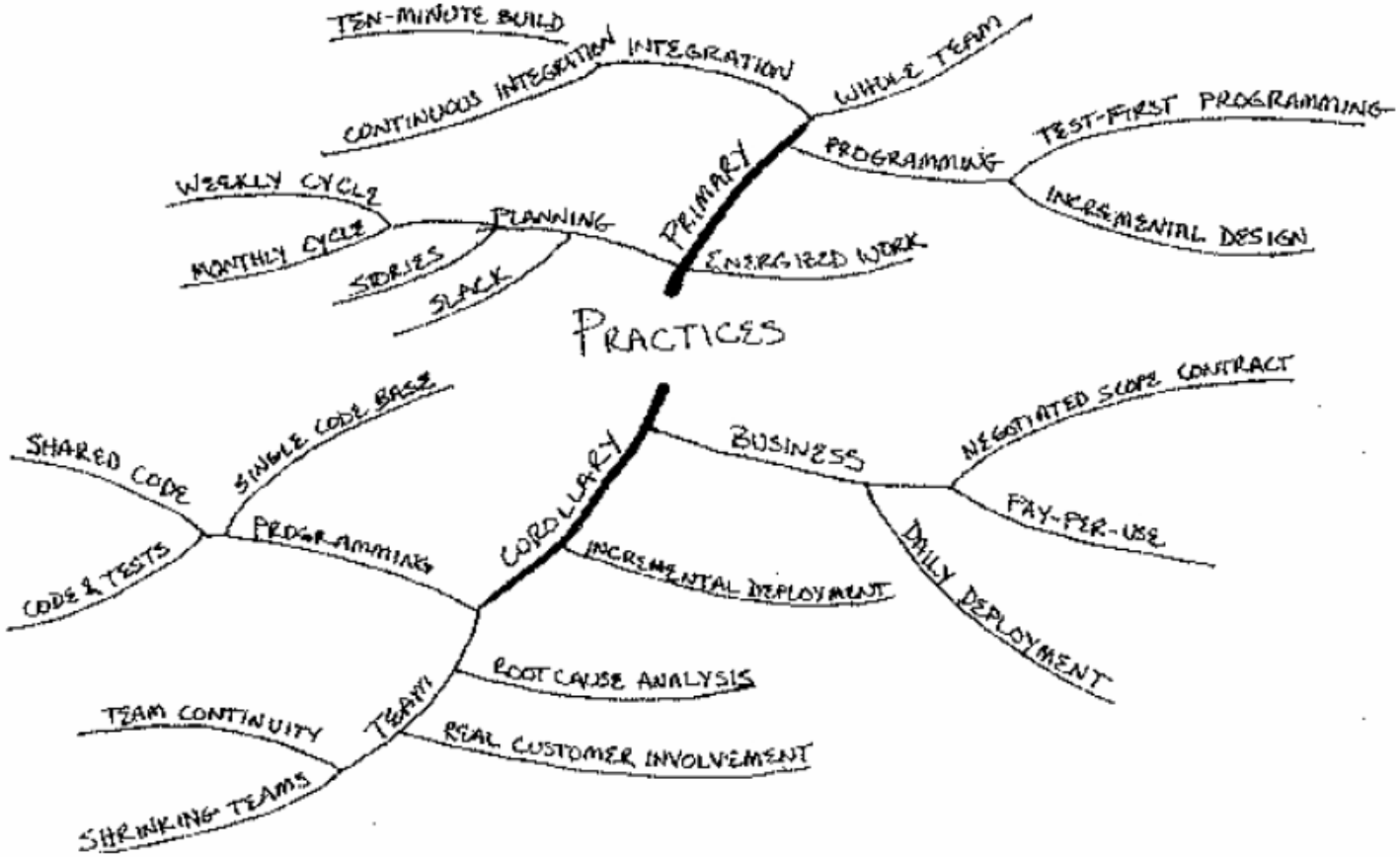
# אבולוציה של מיומנויות

- ניתן ללמוד על ההבשלה של גישת XP ע"י בחינת השינויים שעברו המיומנויות בין המהדורות של הספר [במצגת נפרדת]
  - ***Extreme Programming Explained***
    - **First Edition, Kent Beck, 2000**
    - **Second Edition, Kent Beck with Cynthia Andres 2005**






**Source:** Beck, K. (2000). *eXtreme Programming explained*, Addison Wesley.




**Source:** ***eXtreme Programming explained***
**Kent Beck with Cynthia Andres, Second Edition, 2005**

# The Rules and Practices of Extreme Programming

## Planning

- User stories are written.
- Release planning creates the schedule.
- Make frequent small releases.
- The Project Velocity is measured.
- The project is divided into iterations.
- Iteration planning starts each iteration.
- Move people around.
- A stand-up meeting starts each day.
- Fix XP when it breaks.

## Coding

- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.
- All production code is pair programmed.
- Only one pair integrates code at a time.
- Integrate often.
- Use collective code ownership.
- Leave optimization till last.
- No overtime.

## Designing

- Simplicity.
- Choose a system metaphor.
- Use CRC cards for design sessions.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.

## Testing

- All code must have unit tests.
- All code must pass all unit tests before it can be released.
- When a bug is found tests are created.
- Acceptance tests are run often and the score is published.