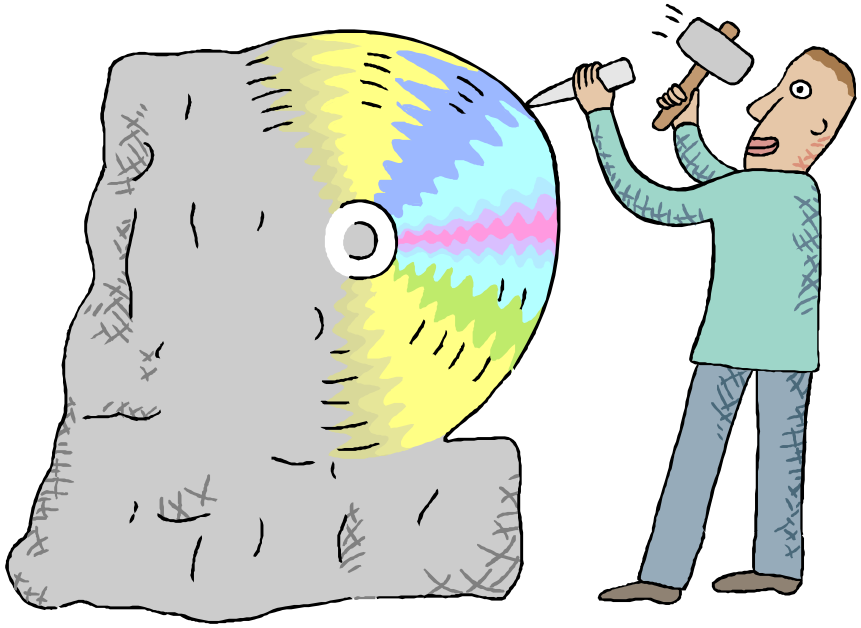
פרופ' עמיר תומר
amir@amirtomer.com

פרק 2

תהליך הפיתוח של מערכת עתירת תוכנה

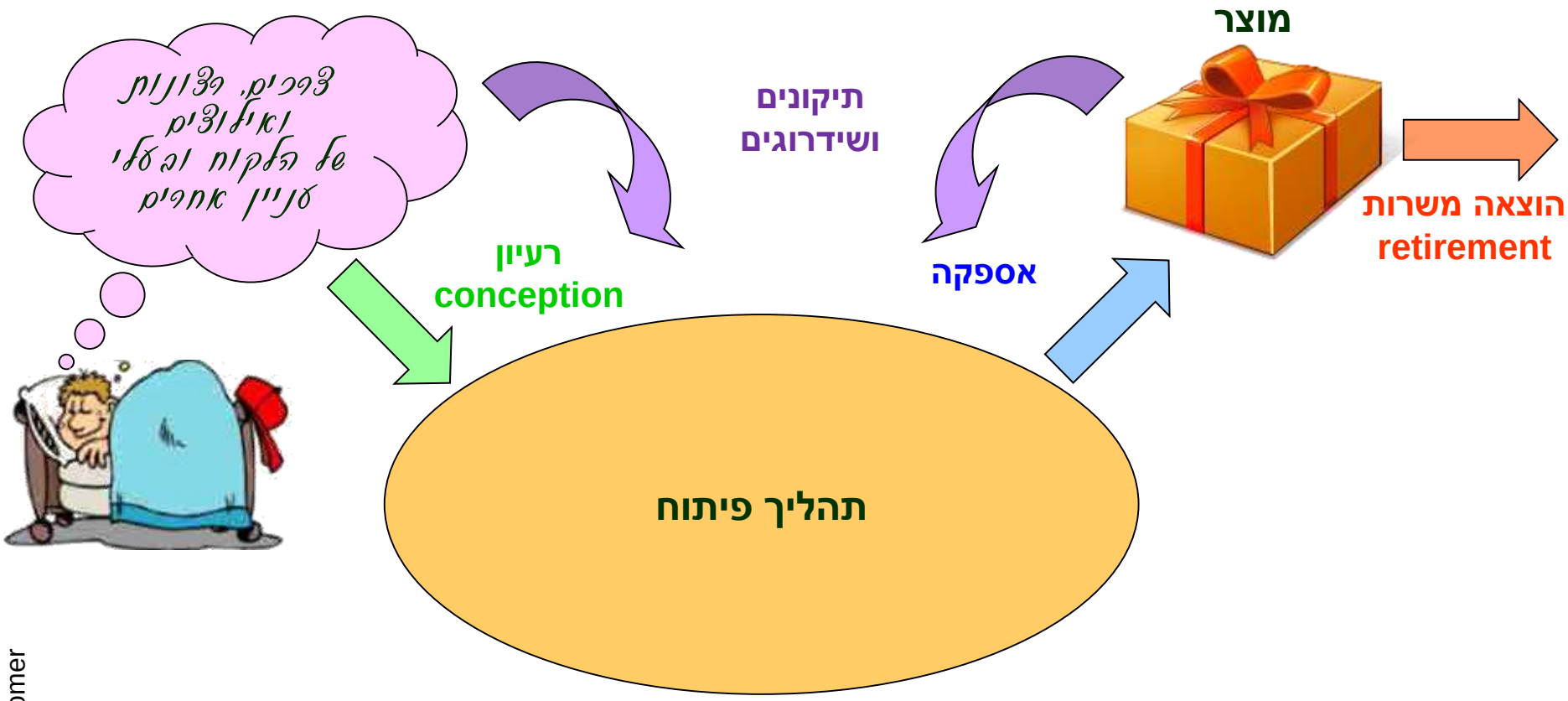
The Development Processes of a Software Intensive System



מוצר תוכנה = התוצר הסופי של תהליך פיתוח התוכנה

- Software Product: The set of computer programs, procedures, and possibly associated documentation and data [ISO/IEC 12207:2008]
- מוצר תוכנה מסופק באחת מ-3 צורות:
  - מוצר עצמאי
    - התוכנה מסופקת בנפרד (על גבי דיסק או בהורדה מהרשת), ומיועד להתקנה על מחשב סטנדרטי הקיים ברשות המשתמש, והכולל מערכת הפעלה ותוכנה תשתיתית.
      - לדוגמה: אנטי-וירוס
  - מוצר משוכּן/משובץ (embedded)
    - התוכנה מסופקת ביחד עם החומרה (היעודית), מערכת ההפעלה ותוכנת התשתית כמוצר שלם אחד.
      - לדוגמה: מכשיר טלפון סלולרי
  - תוכנה כשֵירות (SaaS = Software as a Service)
    - התוכנה עצמה אינה עוברת ללקוח ואינה נשארת ברשותו, אך הוא מקבל את השירותים הדרושים באמצעותה
      - לדוגמה: שרות חשבונות (billing)

מחזור החיים של מוצר תוכנה



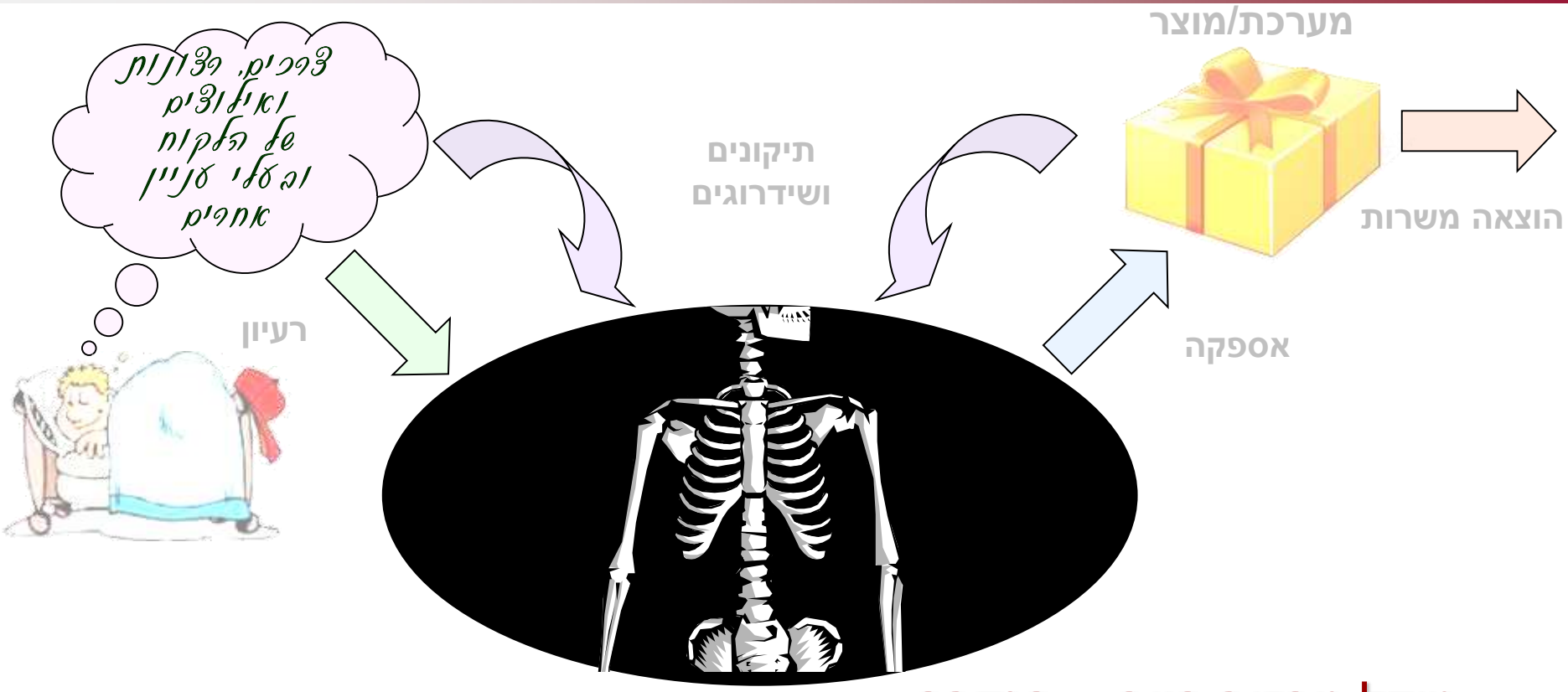
- מחזור חיים - הגדרה
  - *"אבולוציה של מערכת, מוצר, שרות, פרויקט או ישות אחרת מעשה-ידי-אדם מהרעיון (conception) ועד להוצאה משרות (retirement)"* [ISO/IEC 15288]

מהו תהליך פיתוח?

- תהליך פיתוח הוא אספקת מענה לבעיה / צורך
  - באמצעות מיפוי **מרחב בעיה** ל**מרחב פתרון**
- תהליך הפיתוח צריך להכיל את השלבים הבאים
  1. הבנת הבעיה / הצורך
  2. גיבוש תפיסת פתרון (מערכת)
     - תפיסת מבנה
     - תפיסת תפעול
  3. תיכון פתרון (מערכת -> תת-מערכות -> מכלולים -> מודולים)
     - תיכון מבני
     - תיכון פונקציונאלי/תפעולי
  4. מימוש הפתרון
     - בניית מרכיבים
     - שילוב
  5. בחינת הפתרון
     - תאימות לתיכון
     - מענה לבעיה / לצורך

- יישום ספציפי של תהליך פיתוח נקרא "מודל מחזור חיים"

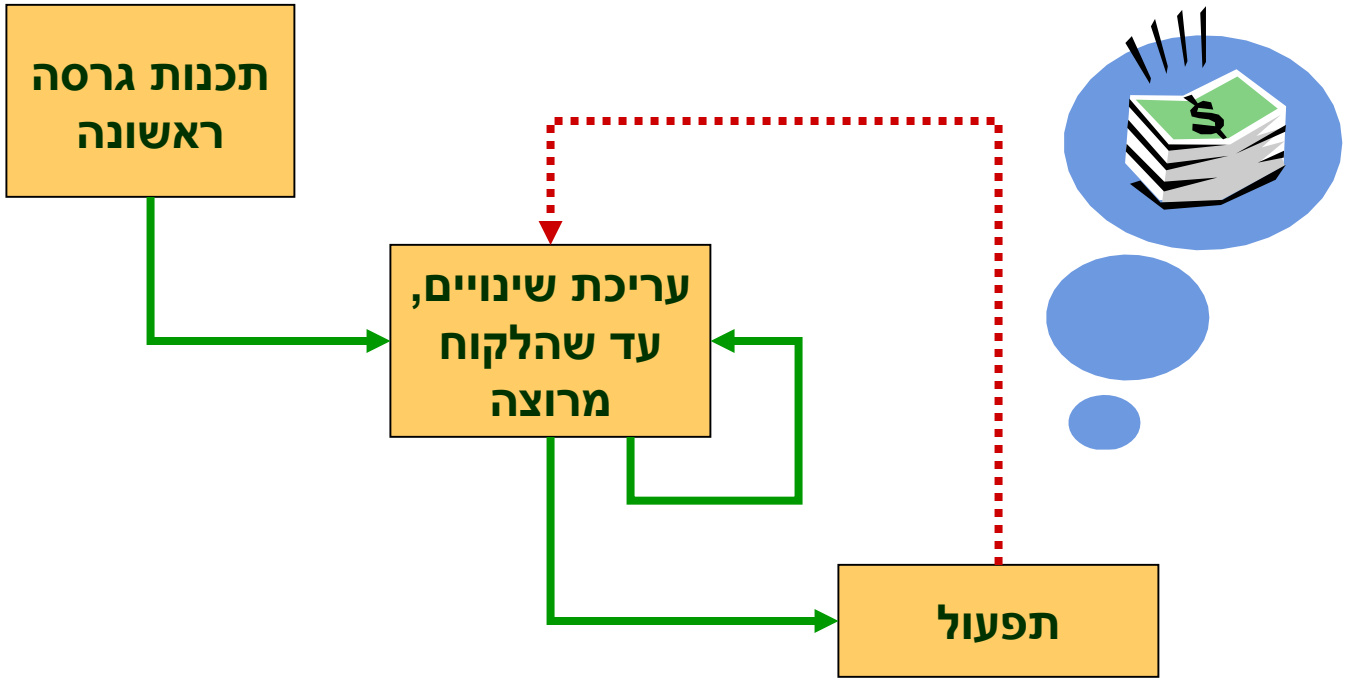
מודל מחזור חיים של מערכת / מוצר תוכנה



- מודל מחזור חיים – הגדרה

– "מסגרת המכילה את התהליכים, הפעילויות והמטלות הכרוכים בפיתוח, תפעול ואחזקה של מוצר תוכנה, לאורך כל חיי המערכת, החל מהגדרת דרישותיה וכלה בסיום השימוש בה" [ISO/IEC 12207]

מודל פשטני לפיתוח תוכנה: "תכנת ותקן" (code & fix)



"תכנת ותקן" - תכונות

- תהליך בלתי מוגדר
  - נסיון להבין את הבעיה דרך בניית פתרון בגישה של ניסוי וטעייה
- תהליך בלתי מתוכנן
  - יודעים מתי מתחילים, אך לא יודעים מתי זה יסתיים
- תהליך בלתי מבוקר
  - הדרישות יכולות להשתנות כל הזמן
- סיכונים עיקריים
  - הרבה מאד "עבודה חוזרת" (rework)
  - לא ברור אם אי פעם המוצר הסופי יתאים ללקוח
- מניע עיקרי
  - "אין זמן"
- תוצאות עיקריות
  - ככל שנמשכים הסבבים השינויים נהיים יותר מסורבלים
    - הכל נבנה "טלאי על טלאי"
  - אחזקה כמעט בלתי אפשרית

האם אפשר לבנות בית בשיטת "בנה ותקן"?

דרישות בעלי עניין נוספים:
- דרישות רישוי
- דרישות תשתיות (ממשקים)

צרכים, רצונות ואילוצים של הלקוח:
- מגורים ושירותים
- חזות
- אופציות עתידיות
- אילוצי תקציב ולו"ז

מימוש, שילוב ובחינה

מפרטי דרישות וארכיטקטורה:
- תכנית קירות, רצפות, גגות
- תכנית חזיתות
- תכנית נקודות חשמל ומים
- תכנית פתחים, מדרגות

תכן ומפרטים טכניים:
- תכנית קונסטרוקציה
- תכניות אינסטלציה (חשמל, מים, ביוב...)

הגדרה ופירוט

בניה:
הקמת שלד
התקנת חשמל, צנרת, ...
גימור

גם תוכנה לא!

דרישות בעלי עניין נוספים:
- דרישות תקנים
- דרישות תאימות (ממשקים)

צרכים, רצונות ואילוצים של הלקוח:
- שירותים / משימות
- ביצועים
- אופציות עתידיות
- אילוצי תקציב ולו"ז

מפרטי דרישות וארכיטקטורה:
- תרחישי פעולה
- ארכיטקטורה לוגית וממשקים
- ישויות מידע
- קונספט הפעלה

תכן ומפרטים טכניים:
- מודולים וממשקים פיזיים
- מבני נתונים / מסדי נתונים
- אלגוריתמים
- פרוטוקולים

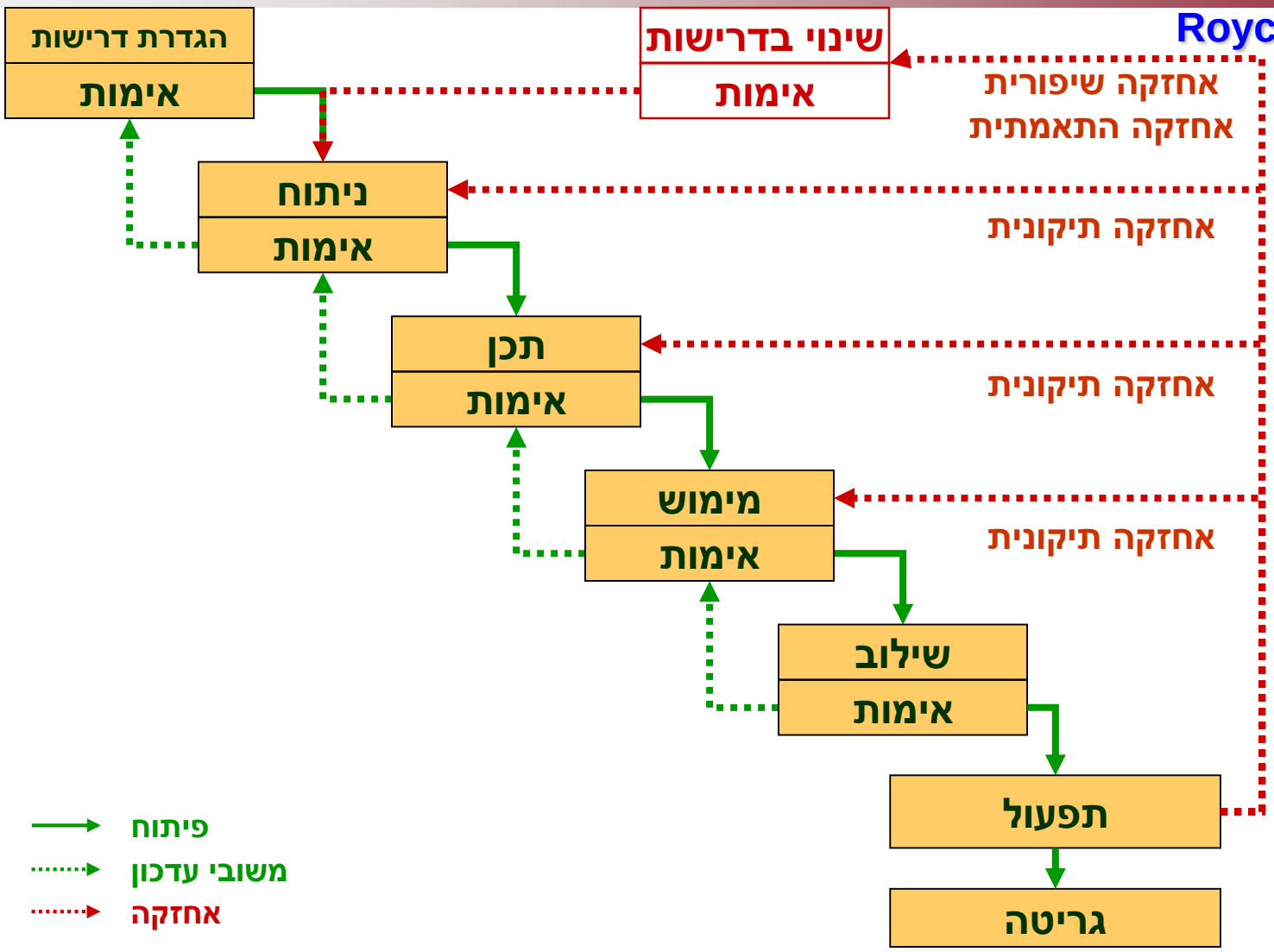
בניה:
קידוד
הידור (קומפילציה)
קישור
שילוב

מימוש, שילוב ובחינה

הגדרה ופירוט

מודל מפל המים (waterfall model)

Royce, 1970



מודל מפל המים - תכונות

- כל שלב מורכב מפעילות אחת בלבד
  - השאיפה היא לסיים כל פעילות בפאזה אחת
  - אין תכנון מראש לשינויים/תוספות
  - קיימים "חוגי משוב" בין שלבים עוקבים
    - במידת הצורך בלבד, לצורך תיקון בעיות שהתגלו
- התהליך הוא מונחה-תיעוד
  - בשלבים הראשונים התוצרים הינם "ניירת" בלבד
  - המעבר לשלב הבא מותנה (ותלוי!) בתיעוד השלב הקודם
    - מוביל לתופעת Paralysis by Analysis
- יתרונות
  - תהליך מתועד היטב
  - אחזקה קלה יותר
- חסרונות
  - אימות הניתוח והתכן יכול להיעשות אך ורק באמצעות התיעוד

חייו (התמוהים) ומותו (הצפוי) של מודל "מפל המים"

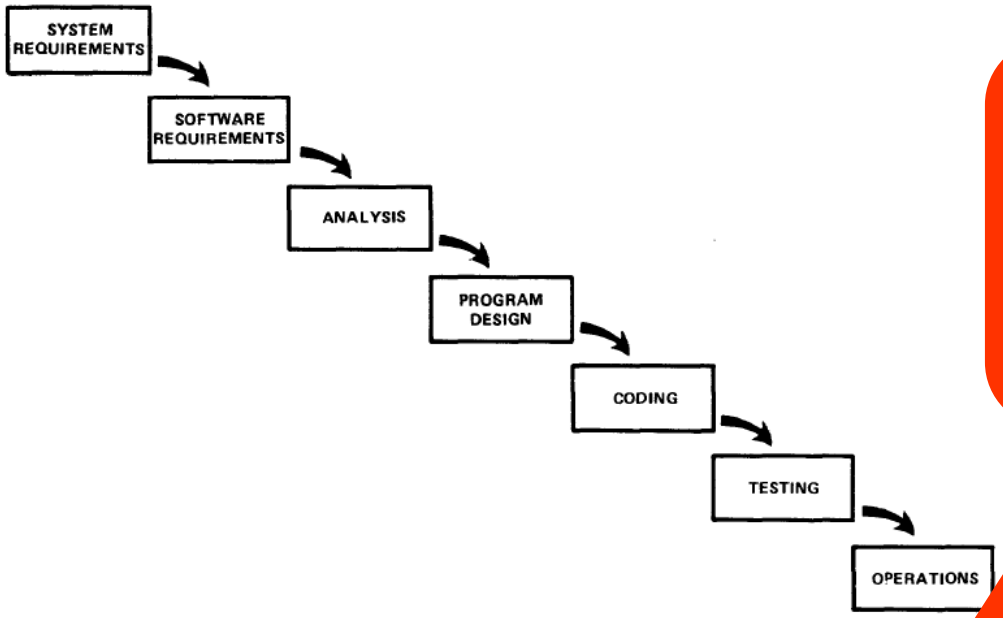
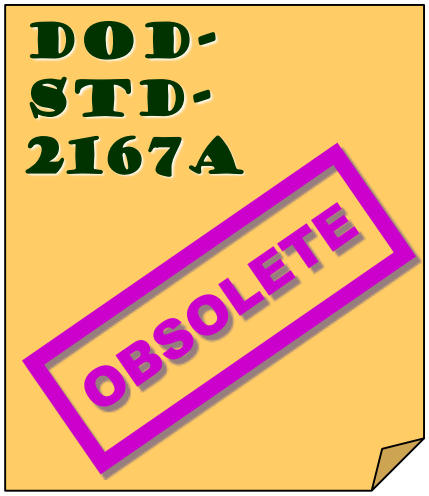


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

I believe in this concept, but the implementation described above is risky and invites failure. The problem is illustrated in Figure 4. The testing phase which occurs at the end of the development cycle is the first event for which timing, storage, input/output transfers, etc., are experienced as distinguished from analyzed. These phenomena are not precisely analyzable. They are not the solutions to the standard partial differential equations of mathematical physics for instance. Yet if these phenomena fail to satisfy the various external constraints, then invariably a major redesign is required. A simple octal patch or redo of some isolated code will not fix these kinds of difficulties. The required design changes are likely to be so disruptive that the software requirements upon which the design is based and which provides the rationale for everything are violated. Either the requirements must be modified, or a substantial change in the design is required. In effect the development process has returned to the origin and one can expect up to a 100-percent overrun in schedule and/or costs.

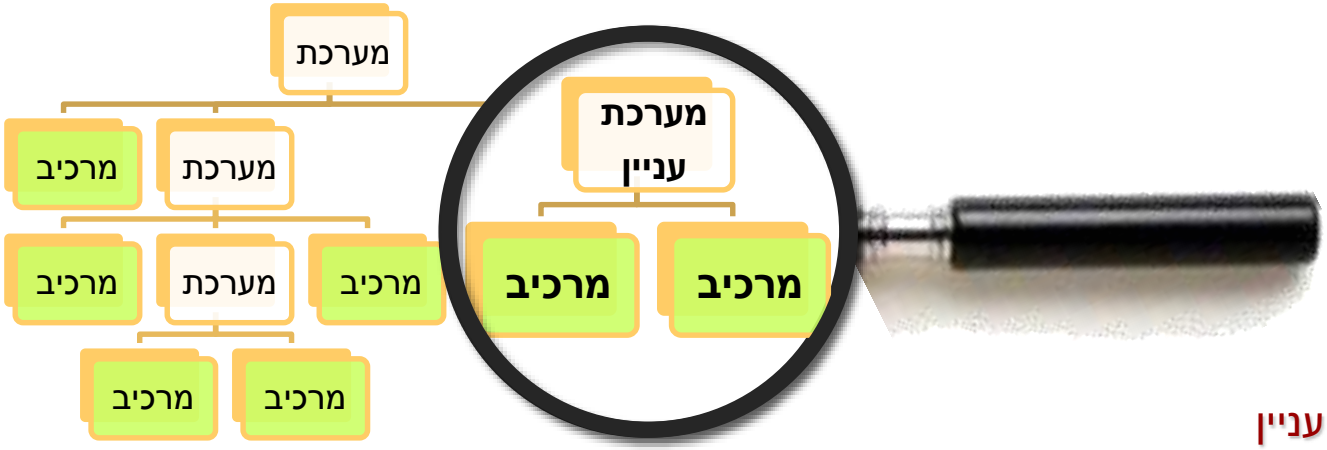
"... the implementation described above is risky and invites failure."

W. W. Royce, 1970



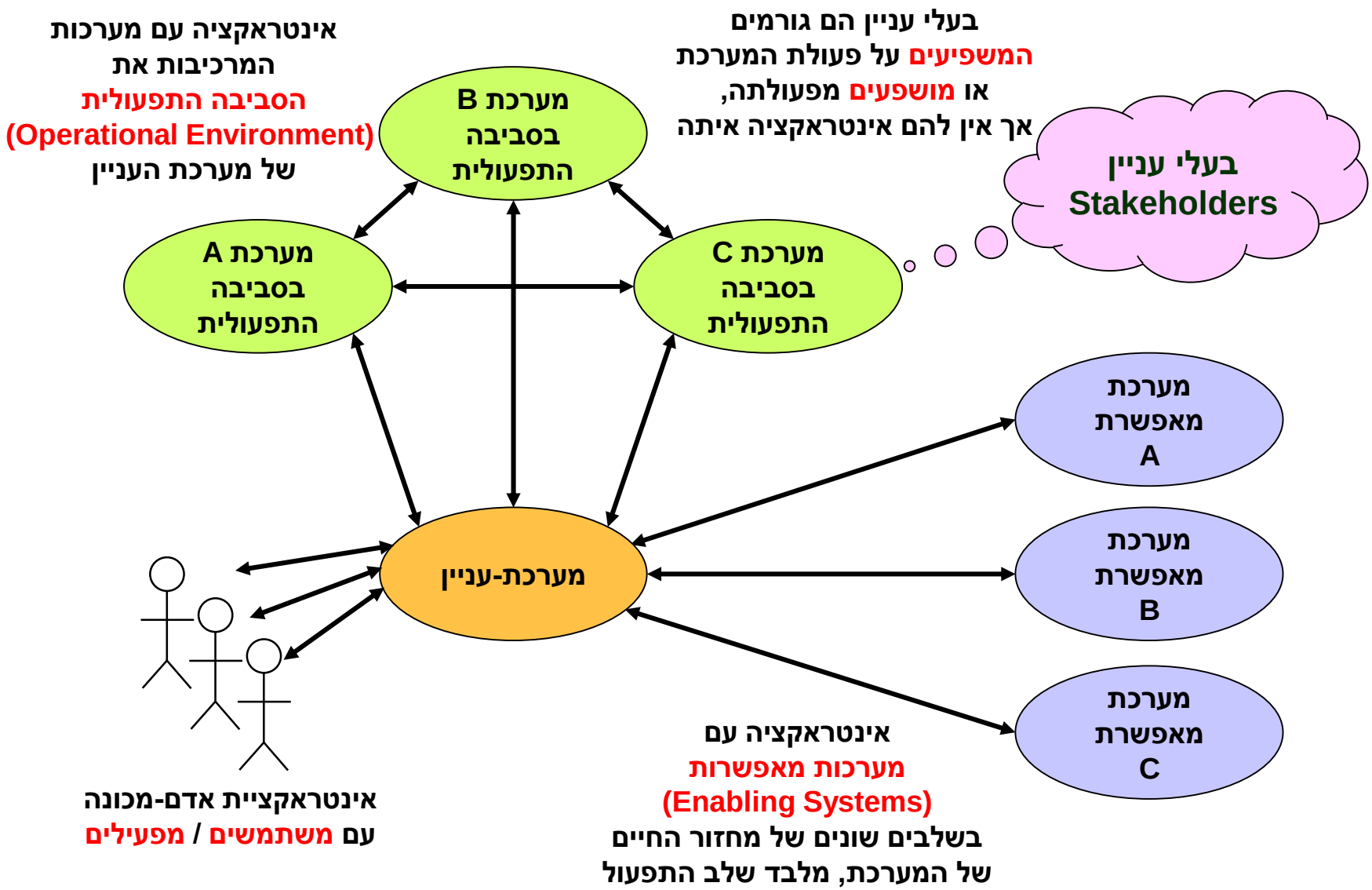
פיתוח תוכנה בראיה מערכתית

- מערכת – הגדרה
  - צירוף של אלמנטים (מרכיבים) המאורגנים ומבצעים אינטראקציה לצורך השגת מטרה מוצהרת אחת או יותר [ISO/IEC 15288]
- מערכת מאורגנת במבנה היררכי-רקורסיבי
  - מרכיב של מערכת יכול להיות מערכת בעצמו



- מערכת עניין
  - מערכת אליה מתייחס תהליך פיתוח נתון, הכולל פעילויות ותוצרים
- מערכת עתירת תוכנה – מע"ת (Software Intensive System - SIS)
  - מערכת אשר התוכנה מהווה בה חלק משמעותי מבחינת הפונקציונליות, עלות הפיתוח, סיכוני הפיתוח או משך הפיתוח.

מערכת עניין וסביבתה



רמות העניין האופייניות בפיתוח מערכת עתירת תוכנה

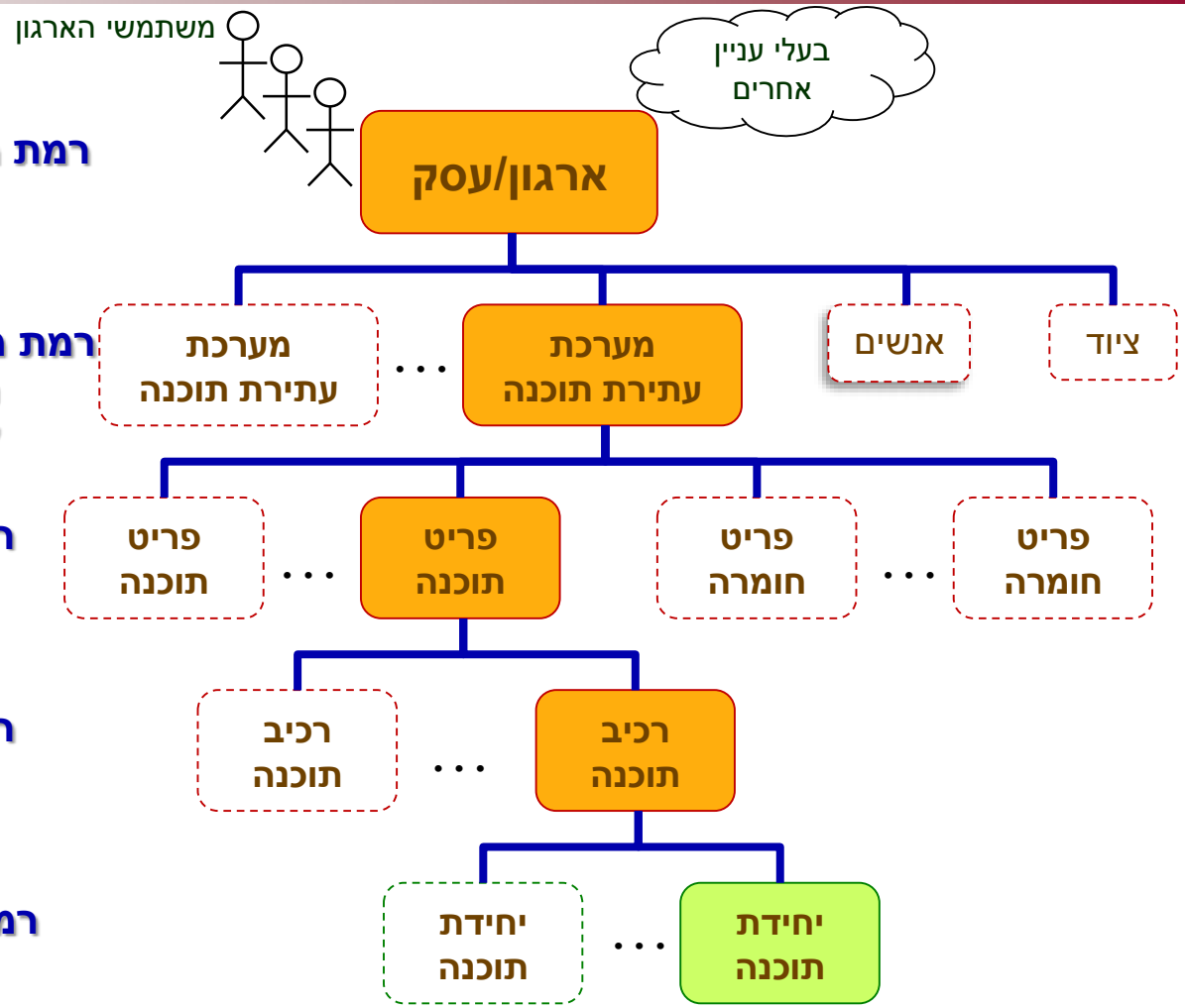
רמת הארגון / העסק
המסגרת התפעולית בה מותקנת המערכת
ואותה נועדה לשמש

רמת מערכת המיחשוב
מערכת חומרה+תוכנה המשרתת את הארגון
לצורך מסויים

רמת פריט התוכנה (CSCI)
יישום התוכנה המותקן על גבי
החומרה ומולו פועלים משתמשים
ומערכות אחרות

רמת רכיב התוכנה
המרכיבים מהם בנוי היישום
מרכיבי בניה: למשל ספריות
מרכיבי ריצה: למשל dll

רמת יחידת התוכנה
מודולי הקוד (למשל מחלקות) מהם
נוצר הרכיב



רמות העניין האופייניות בהקשר של מערכות רפואיות (דוגמה)

רמת הארגון / העסק

המוסד הרפואי בו מותקנת המערכת

המשימה הרפואית: אבחון / טיפול

לוגיקת העבודה: פרוצדורה רפואית

רמת מערכת המיחשוב

מערכת חומרה+תוכנה המסייעת למוסד הרפואי לתת את השירות

המשימה התפעולית: צילום CT

לוגיקת העבודה: צילום. עיבוד תמונות. פיענוח

רמת פריט התוכנה (CSCI)

תוכנה ייעודית, מוכוונת משימה

המשימה הטכנית: הפקת אוסף תמונות

לוגיקת העבודה: תפעול המכשיר

רמת רכיב התוכנה

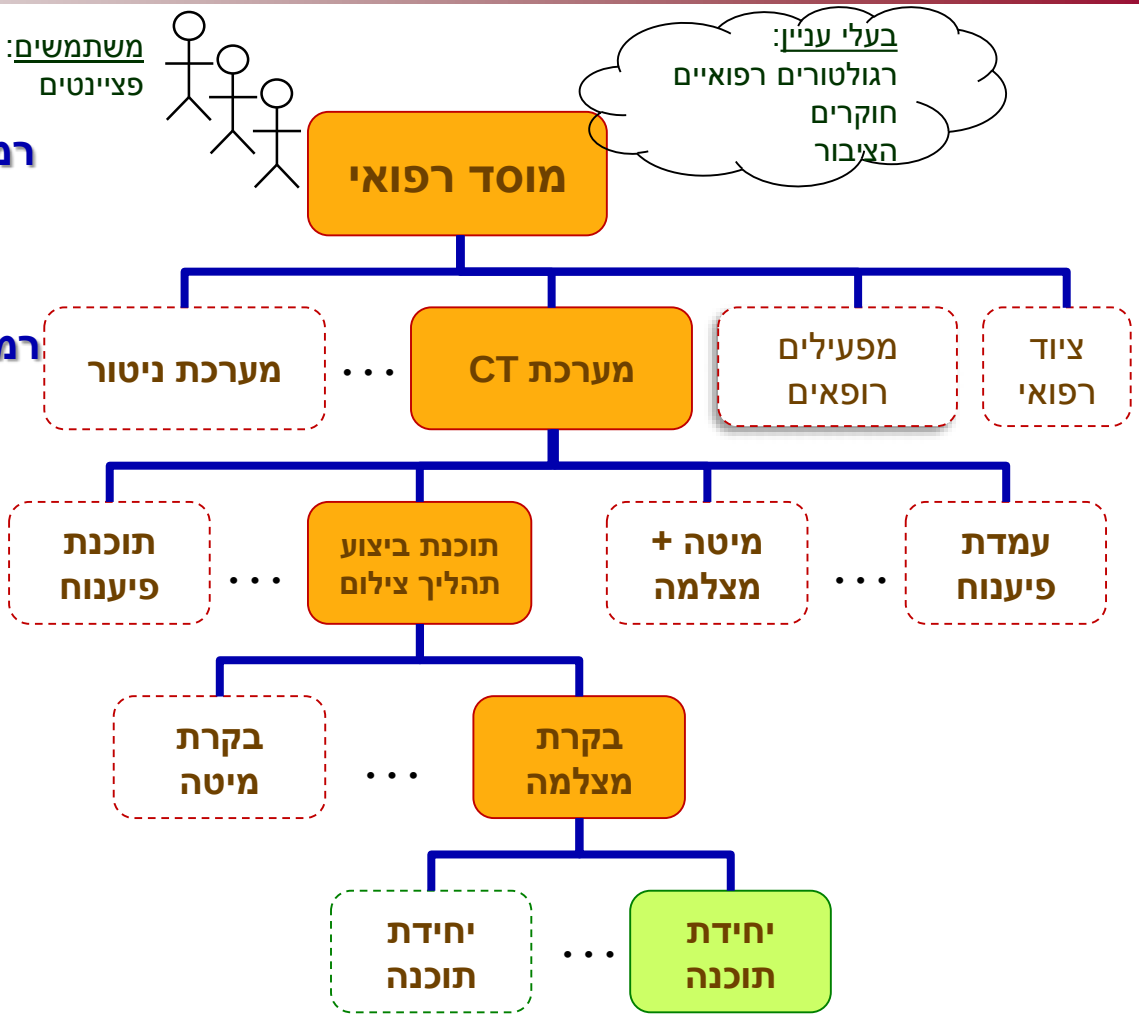
אבני בניין מהם מורכבת התוכנה

מרכיבי בניה: למשל ספריות עיבוד תמונה

מרכיבי ריצה: למשל חוג בקרת עקיבה

רמת יחידת התוכנה

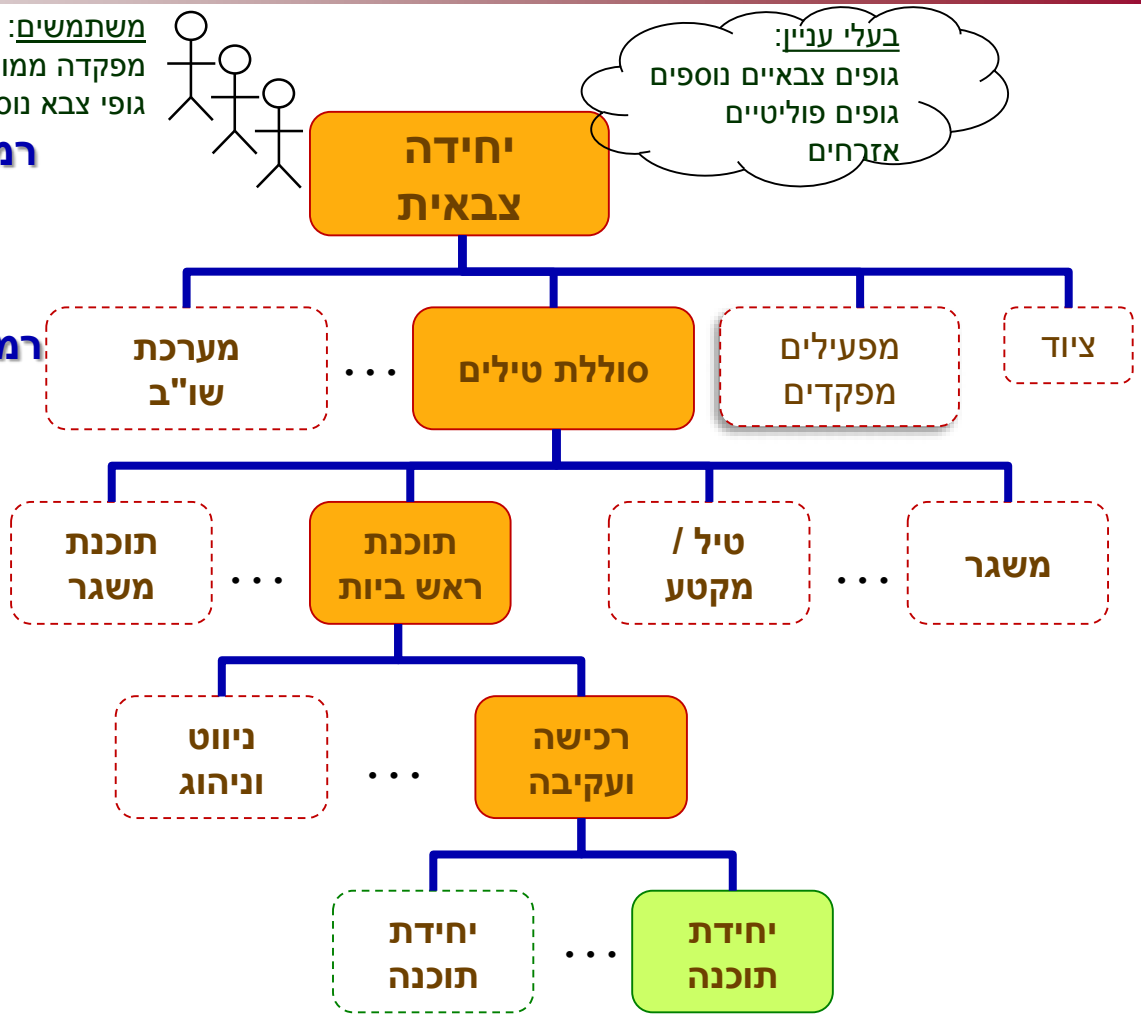
מודולי הקוד (למשל מחלקות) מהם נוצר הרכיב



רמות העניין האופייניות בהקשר של מערכות צבאיות (דוגמה)

משתמשים:
מפקדה ממונה
גופי צבא נוספים המשתמשים בשירותי היחידה

בעלי עניין:
גופים צבאיים נוספים
גופים פוליטיים
אזרחים



רמת הארגון / העסק

היחידה הצבאית בה מותקנת המערכת.

המשימה הצבאית: הגנה מפני מתקפה

לוגיקת העבודה: תו"ל

רמת מערכת המיחשוב

מערכת חומרה+תוכנה המסייעת ליחידה בביצוע משימותיה

המשימה התפעולית: יירוט איומים

לוגיקת העבודה: אופיוו תפעול

רמת פריט התוכנה (CSCI)

תוכנה ייעודית, מוכוונת משימה

המשימה הטכנית: גילוי, זיהוי, עקיבה, השמדה

לוגיקת העבודה: תהליכי מערכת

רמת רכיב התוכנה

אבני בניין מהם מורכבת התוכנה

מרכיבי בניה: למשל ספריות עיבוד תמונה

מרכיבי ריצה: למשל חוג בקרת עקיבה

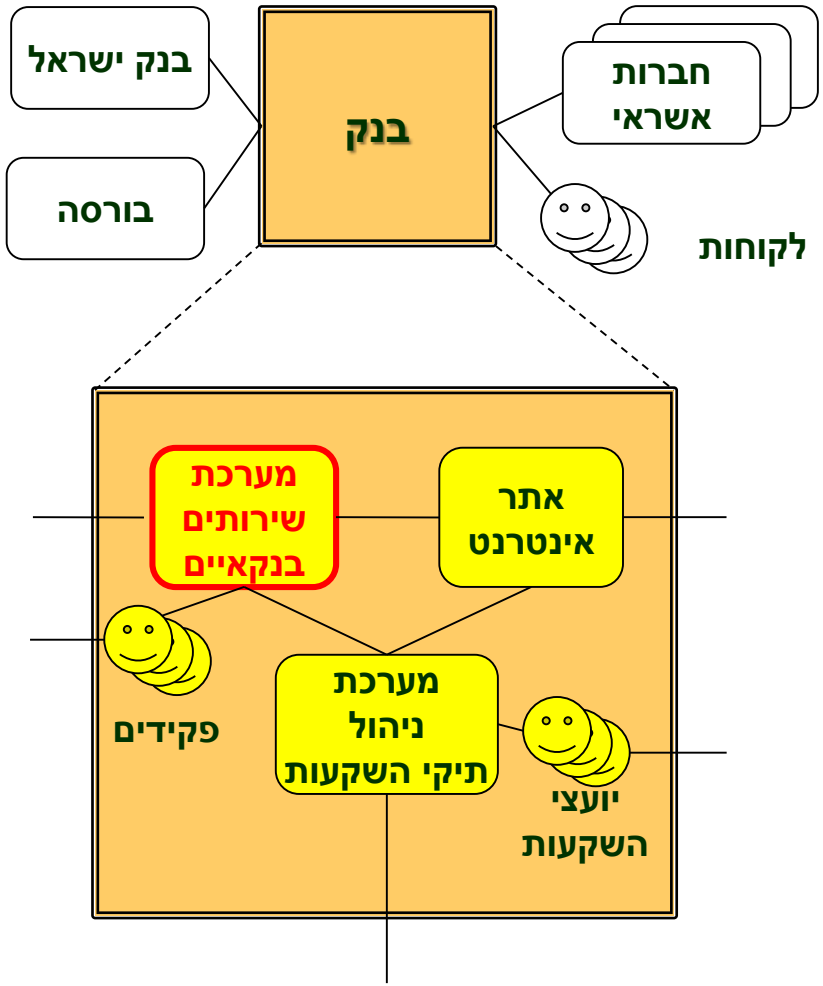
רמת יחידת התוכנה

מודולי הקוד (למשל מחלקות) מהם נוצר הרכיב

מערכת – מאפיינים

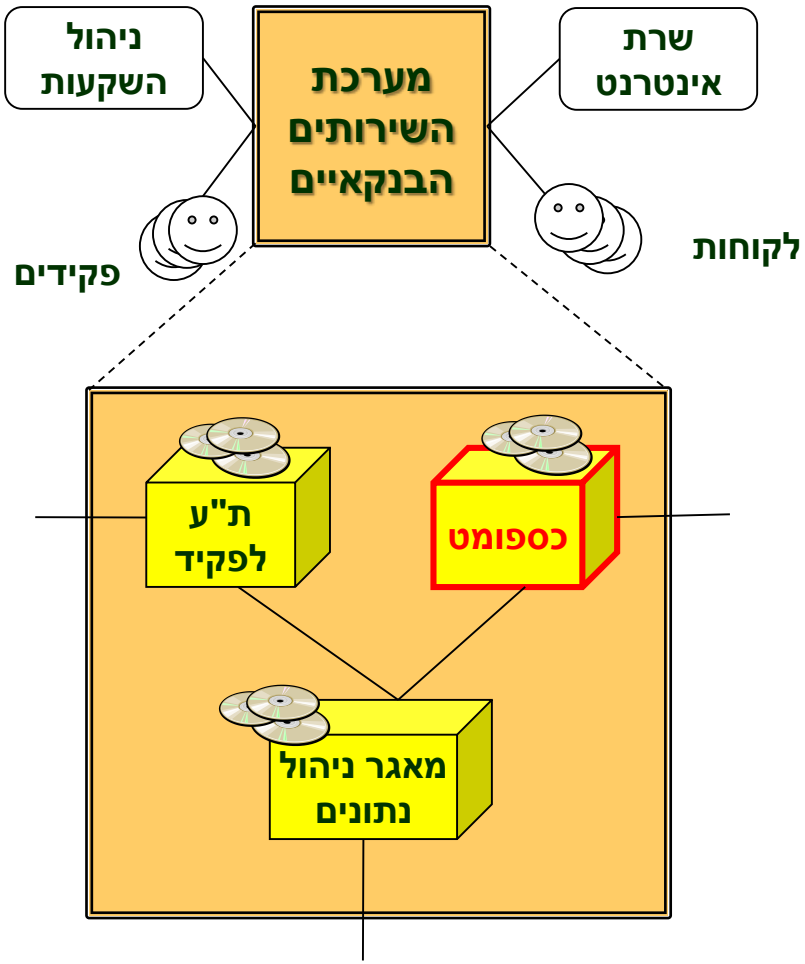
- מערכת (בכל רמה, כולל מרכיב אלמנטרי) מוגדרת על ידי המאפיינים הבאים:
  - מטרה
    - פתרון הבעיה / המענה לצורך (של הלקוח ובעלי העניין)
  - סביבה
    - הגורמים החיצוניים (אנושיים ולא-אנושיים) מולם פועלת המערכת
  - מרכיבים
    - מרכיבי הפתרון
  - ארגון = מבנה
    - הקשרים שבין המרכיבים
    - הקשרים בין המערכת (ומרכיביה) לבין הסביבה
  - אינטראקציה = התנהגות
    - האינטראקציה בין הרכיבים לבין עצמם, ובינם לבין הסביבה, לצורך השגת המטרה

רמת הארגון/העסק – Organization/Business Level



- **מטרה (צורך)**
  - מתן שירותים בנקאיים (עו"ש, השקעות, הלוואות) ללקוחות
- **סביבה**
  - לקוחות
  - חברות אשראי
  - בנק ישראל
  - בורסה
- **מרכיבים**
  - מערכות ממוחשבות
  - פקידים
  - יועצי השקעות
  - אתר אינטרנט
- **מבנה (קשרים)**
  - קישור (פנימי וחיצוני) בין המערכות
  - ממשקי משתמש פנימיים וחיצוניים
  - ממשקים בין לקוחות לפקידים / יועצים
- **התנהגות (אינטראקציה בין מרכיבים)**
  - אינטראקציה בין המערכת למשתמשים
  - אינטראקציה עם מערכות חיצוניות
  - אינטראקציה בין המרכיבים לבין עצמם

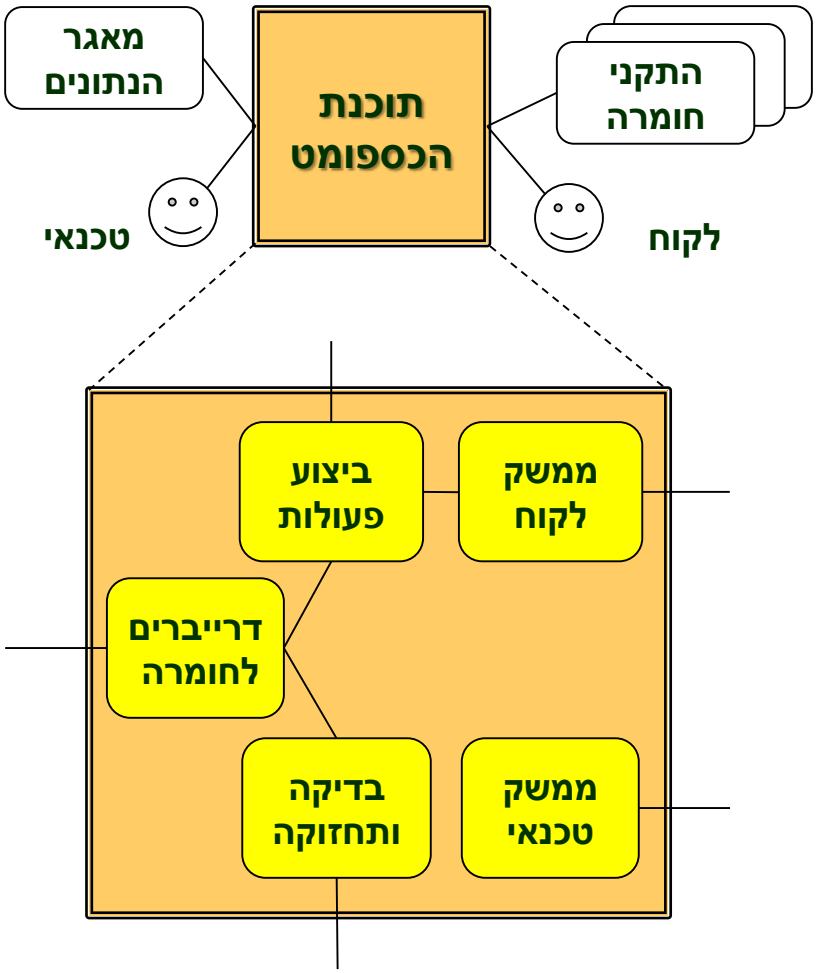
רמת המערכת – System Level



- מטרה (צורך)
  - לאפשר ללקוחות לקבל את שירותי הבנק ללא סיוע אנושי
  - לאפשר לפקידים לבצע פעולות בנקאיות באופן ממוחשב
- סביבה
  - המרכיבים המקושרים מהרמה הקודמת
- מרכיבים (חומרה, תוכנה)
  - תחנות עבודה לפקידים
  - כספומטים (ת"ע ללקוחות)
  - מאגר ניהול נתונים
- מבנה (קשרים בין מרכיבים)
  - פריסת התוכנה על גבי החומרה ("התקנה")
  - תקשורת פיזית פנימית וחיצונית
  - תקשורת נתונים (פרוטוקולים) פנימית וחיצונית
- התנהגות (אינטראקציה בין מרכיבים)
  - ביצוע פעולות בכספומט
  - ביצוע פעולות ע"י פקיד
  - ביצוע פעולות פנימיות (אתחול, גיבוי, ...)

רמת פריט התוכנה (היישום) – CSCI / Application Level

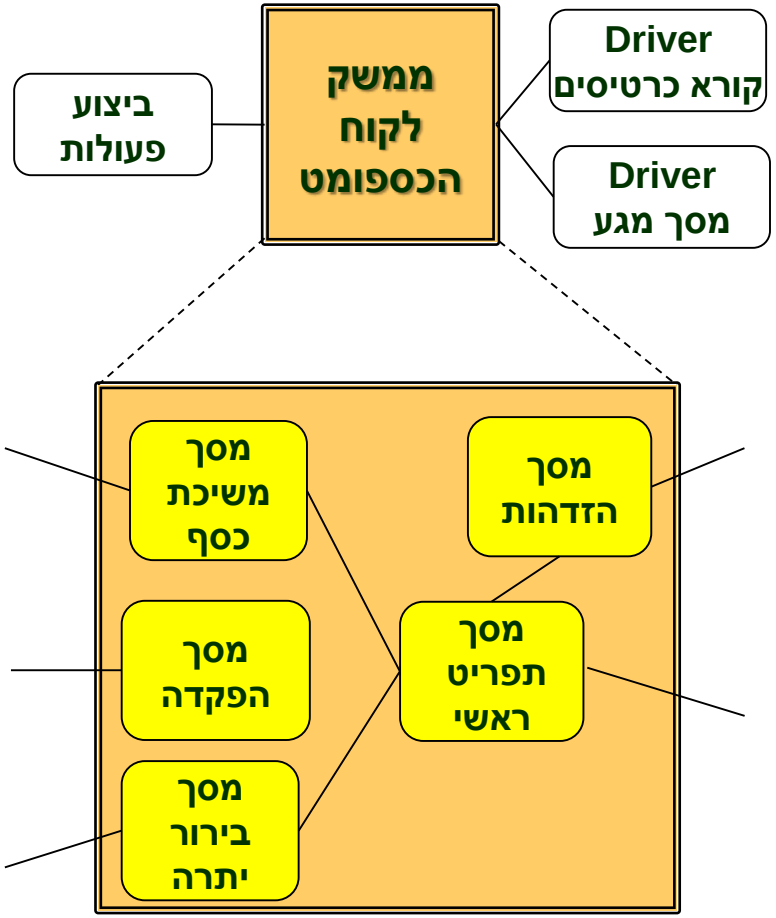
CSCI = Computer Software Configuration Item



- מטרה (צורך)
  - לאפשר ללקוח לבצע פעולות בחשבונו ללא סיוע פקיד
- סביבה
  - פריטי תוכנה אחרים
  - התקני חומרה
- מרכיבים (מודולי תוכנה, אובייקטים)
  - ממשקי משתמשים
  - דרייברים לחומרה
  - לוגיקת תפעול
- מבנה (קשרים בין מרכיבים)
  - קשר בין ממשקי משתמש לרכיבים הלוגיים
  - קשר בין רכיבים לוגיים לדרייברים
  - קשר בין דרייברים לחומרה
  - קשר בין רכיבים לוגיים למאגר הנתונים
- התנהגות (אינטראקציה בין מרכיבים)
  - תהליכי הזדהות, משיכה, הפקדה, ...
  - תהליכי איתחול, בדיקה, תחזוקה, ...
  - תהליכים פנימיים (BIT, בדיקת תקשורת, ...)

רמת רכיב התוכנה – CSC Level

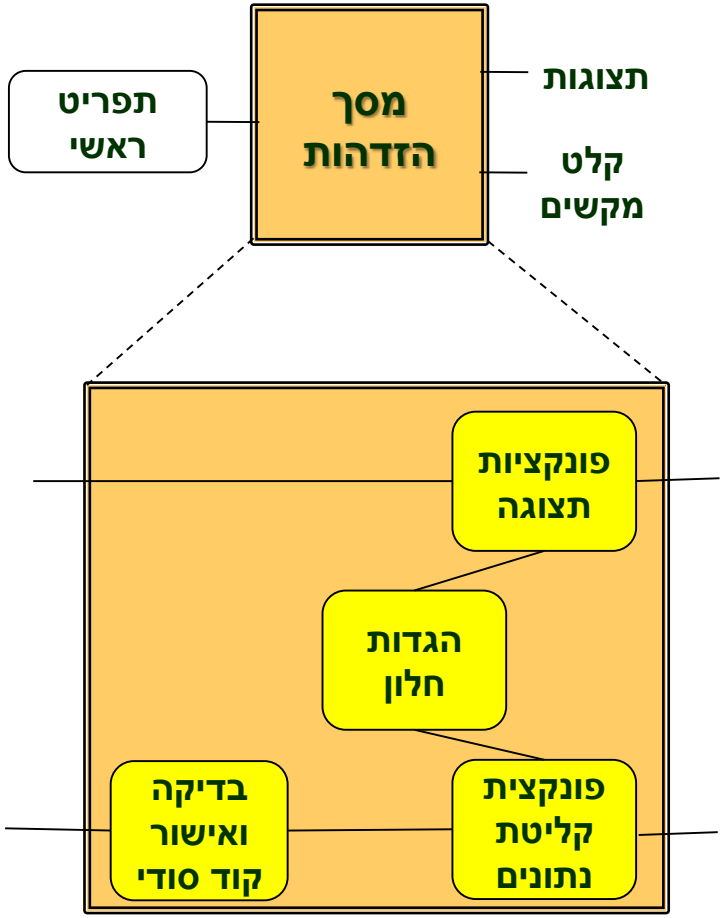
CSC = Computer Software Component



- מטרה (צורך)
  - לאפשר ללקוח להשתמש בכספומט באמצעות מסך מגע ואמצעים נוספים
- סביבה
  - רכיבי תוכנה אחרים
  - רכיבי חומרה
- מרכיבים (רכיבי תוכנה)
  - מסכים / תפריטים
- מבנה (קשרים בין מרכיבים)
  - קשר בין מסך ראשי למסכים יעודיים
  - קשר בין המסכים לרכיבים הלוגיים
- התנהגות (אינטראקציה בין מרכיבים)
  - אינטראקציה בין המסכים לבין הרכיבים המממשים תרחישים שונים
    - תרחיש משיכה מוצלח
    - תרחיש משיכה עם יתרה מוגבלת
    - תרחיש משיכה עם כמות כסף מוגבלת במכשיר
    - תרחיש התאוששות מנפילת תקשורת

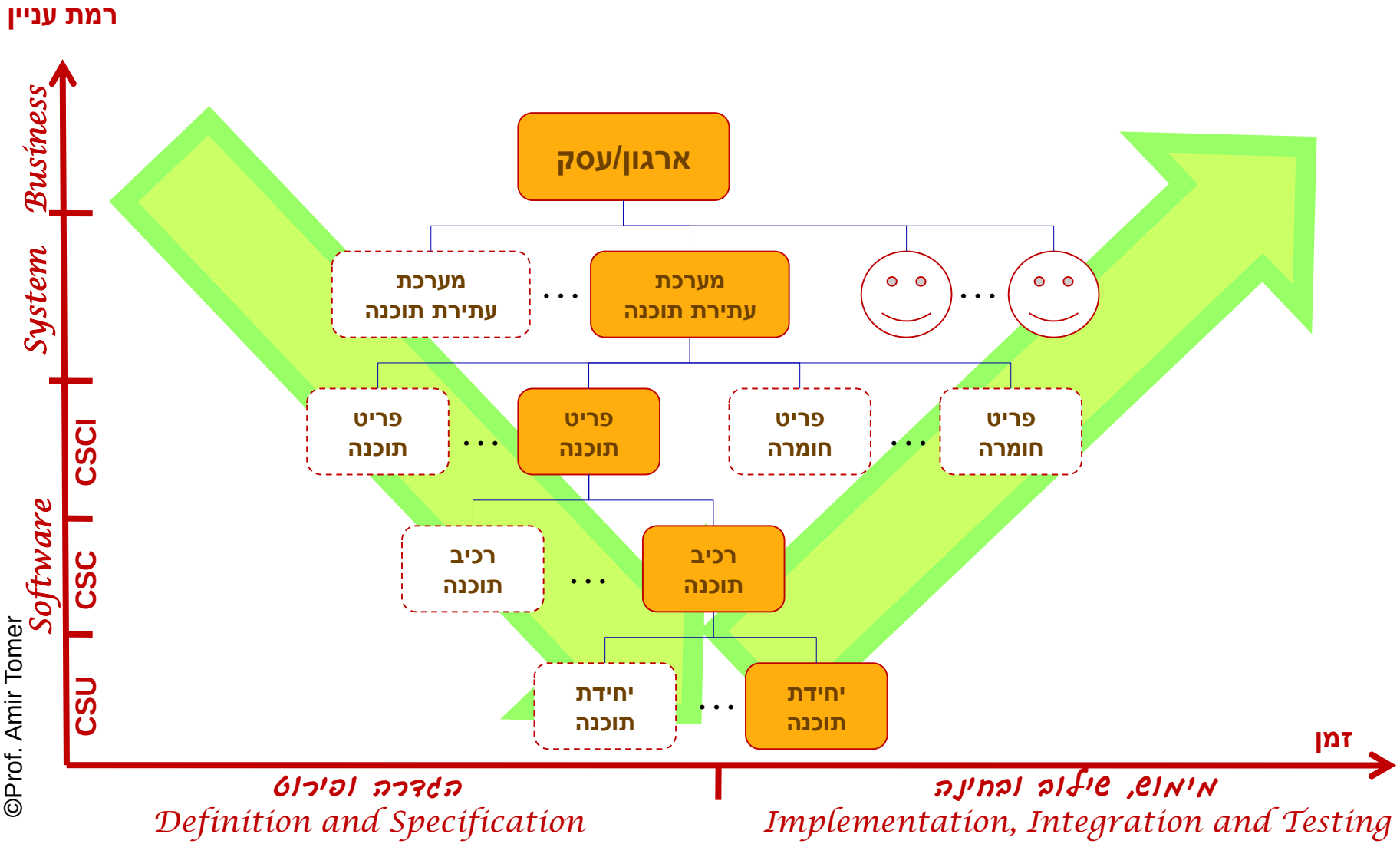
רמת יחידת התוכנה (מודול, אובייקט) – CSU Level

CSU = Computer Software Unit



- מטרה (צורך)
  - לספק פונקציונאליות מוגדרת
- סביבה
  - מודולי תוכנה אחרים
- מרכיבים
  - מבני נתונים (attributes)
  - פונקציות (methods)
- מבנה
  - מצביעים
  - API / "חתימות" של מתודות
- התנהגות
  - פעולת מתודות תוך שימוש במבני הנתונים

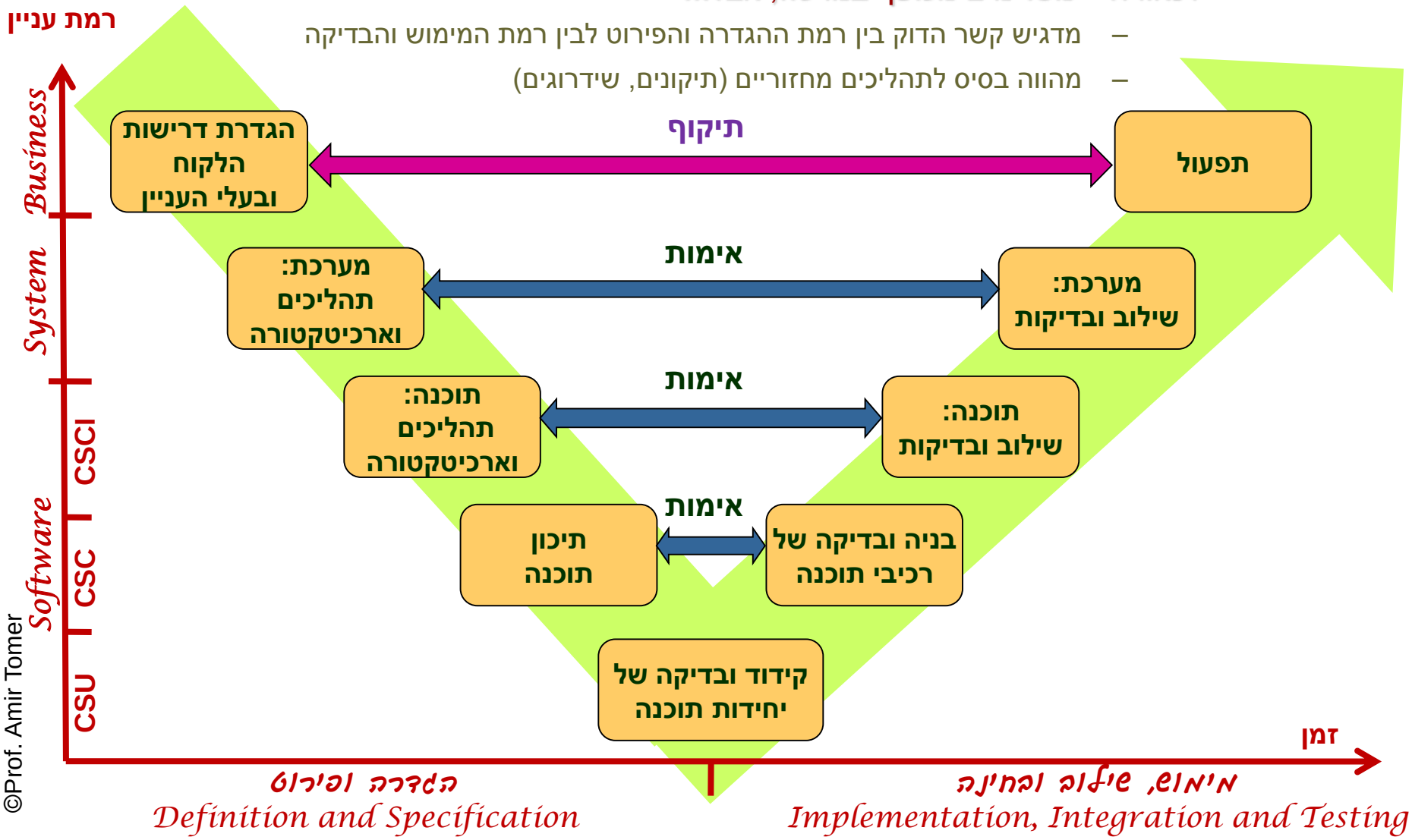
מסגרת פיתוח התוכנה של מערכת עתירת תוכנה



"מודל V" – מסגרת גנרית לתהליך פיתוח תוכנה

לכאורה – מפל מים מכופף במרכזו, אבל...

- מדגיש קשר הדוק בין רמת ההגדרה והפירוט לבין רמת המימוש והבדיקה
- מהווה בסיס לתהליכים מחזוריים (תיקונים, שידרוגים)

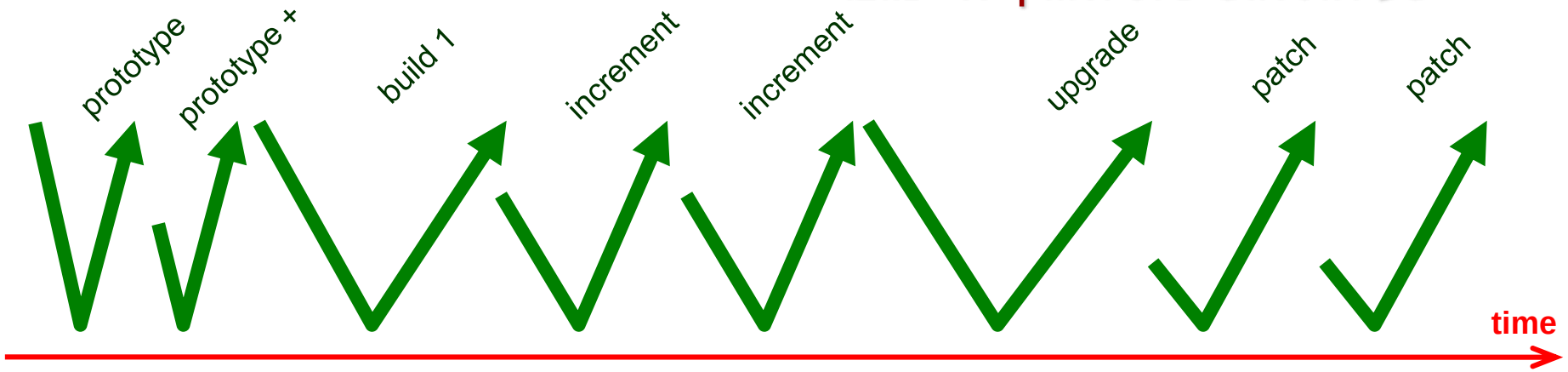


יישום מודל "V" לאורך מחזור חיי המוצר (1)

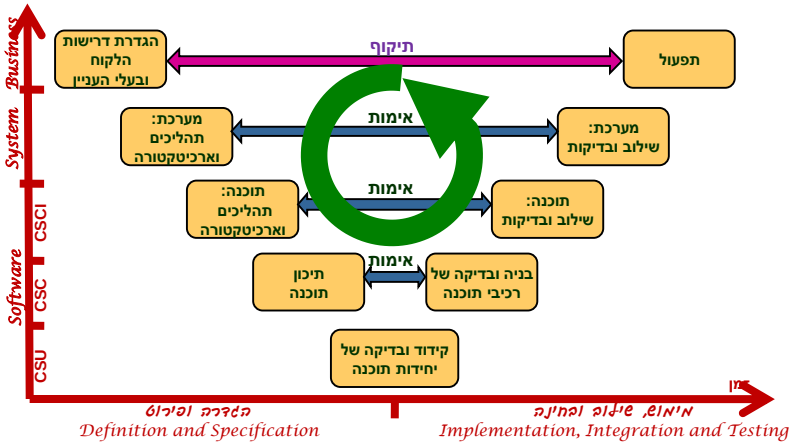
- מסלול ה-V חוזר על עצמו לאורך מחזור חיי המוצר, כדלהלן:
  - קדם-פיתוח / בניית אב-טיפוס (prototype)
    - בניית אבי-טיפוס ודגמים להוכחת היתכנות
    - ארכיטקטורה שלדית שתתמוך במרכיבים נסיוניים ותאפשר הדגמת יכולות
  - סבב פיתוח ראשון (build 1)
    - ייצוב ארכיטקטורה שתתמוך ככל האפשר בשינויים עתידיים
    - מימוש פונקציונליות מלאה או חלקית
  - סבבים נוספים בפיתוח ללא שינויי דרישות / תיקון תקלות (increments/patches)
    - ארכיטקטורה לא משתנה, ולכן הכניסה ל-V עשויה להיות בנקודה נמוכה יותר
    - תכן עשוי לעבור עדכונים (refactoring), בעיקר כדי לשפר ביצועים
    - שילובים ובחינה במסלול מלא עם דגש על בדיקות רגרסיה
  - סבבים נוספים בפיתוח עם שינויי דרישות / שידרוגים (upgrades)
    - הכניסה ל-V מתחילתו (דרישות הלקוח)
    - יש לנתח ולוודא שהארכיטקטורה עדיין תקפה או לערוך שינויים במידת הצורך
    - שילובים ובחינה – כנ"ל

יישום מודל "V" לאורך מחזור חיי המוצר (2)

- פעילות אופיינית לאורך חיי מוצר



- בפועל: ה-"V" הגנרי מיושם במחזורים משתנים

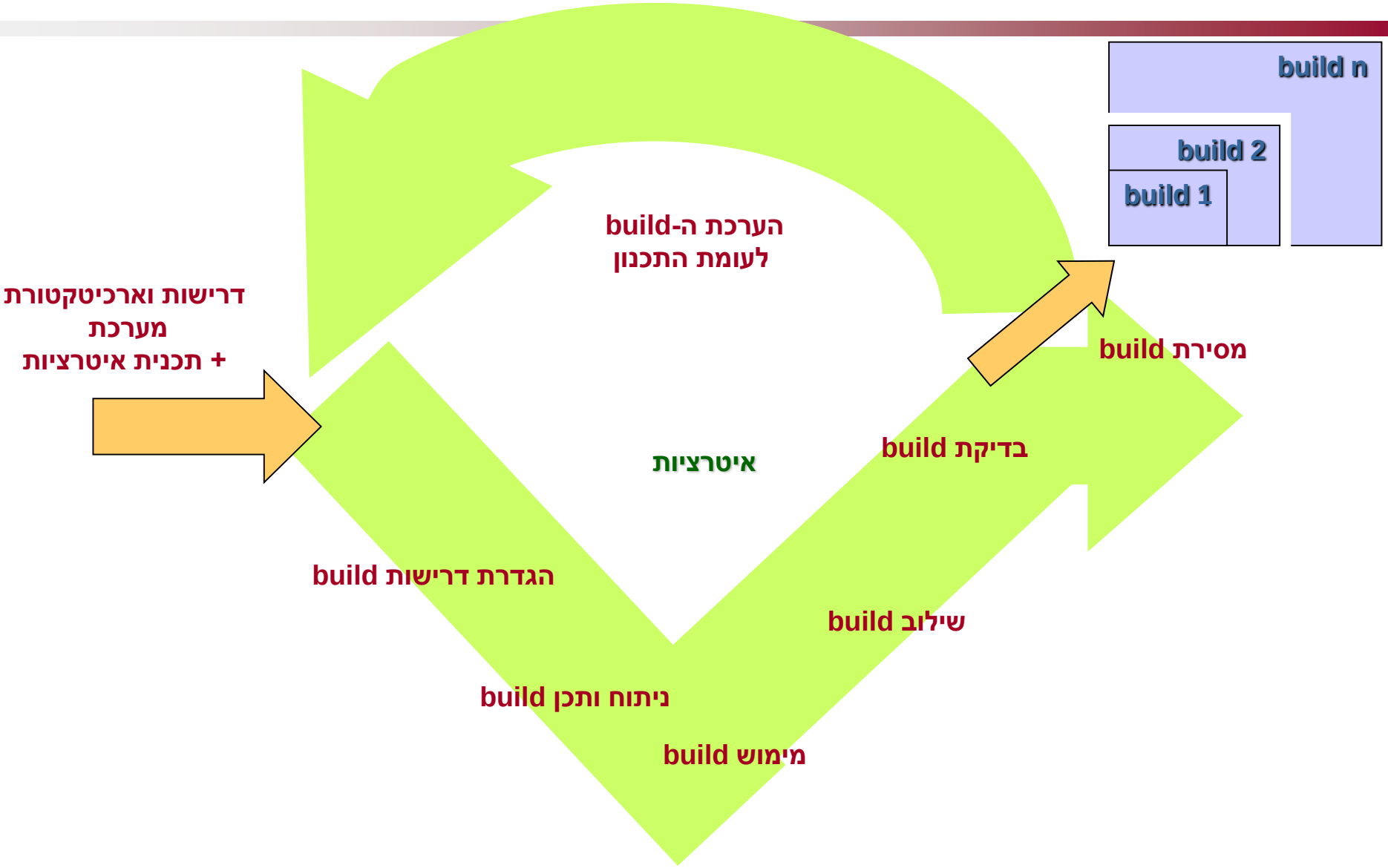


לקראת סיום הקורס נדון בהרחבה במודלים שונים של מחזור החיים

פיתוח איטרטיבי/אינקרמנטלי

- הרעיון: חלוקת הפרויקט ל"פרוסות"
  - תכנית איטרציות כללית וניהול הדוק של כל איטרציה
  - תוצרי ביניים (builds) תפעוליים/מבצעיים – בדוקים ועובדים
  - הצמדת האיטרציות לסיכונים
    - כל תוצר ביניים מסיר סיכון עיקרי
- אין בספרות הגדרה ברורה של המונחים
  - אפשר להבין אותם כך:
    - המוצר הוא אינקרמנטלי
    - תהליך הפיתוח הוא איטרטיבי

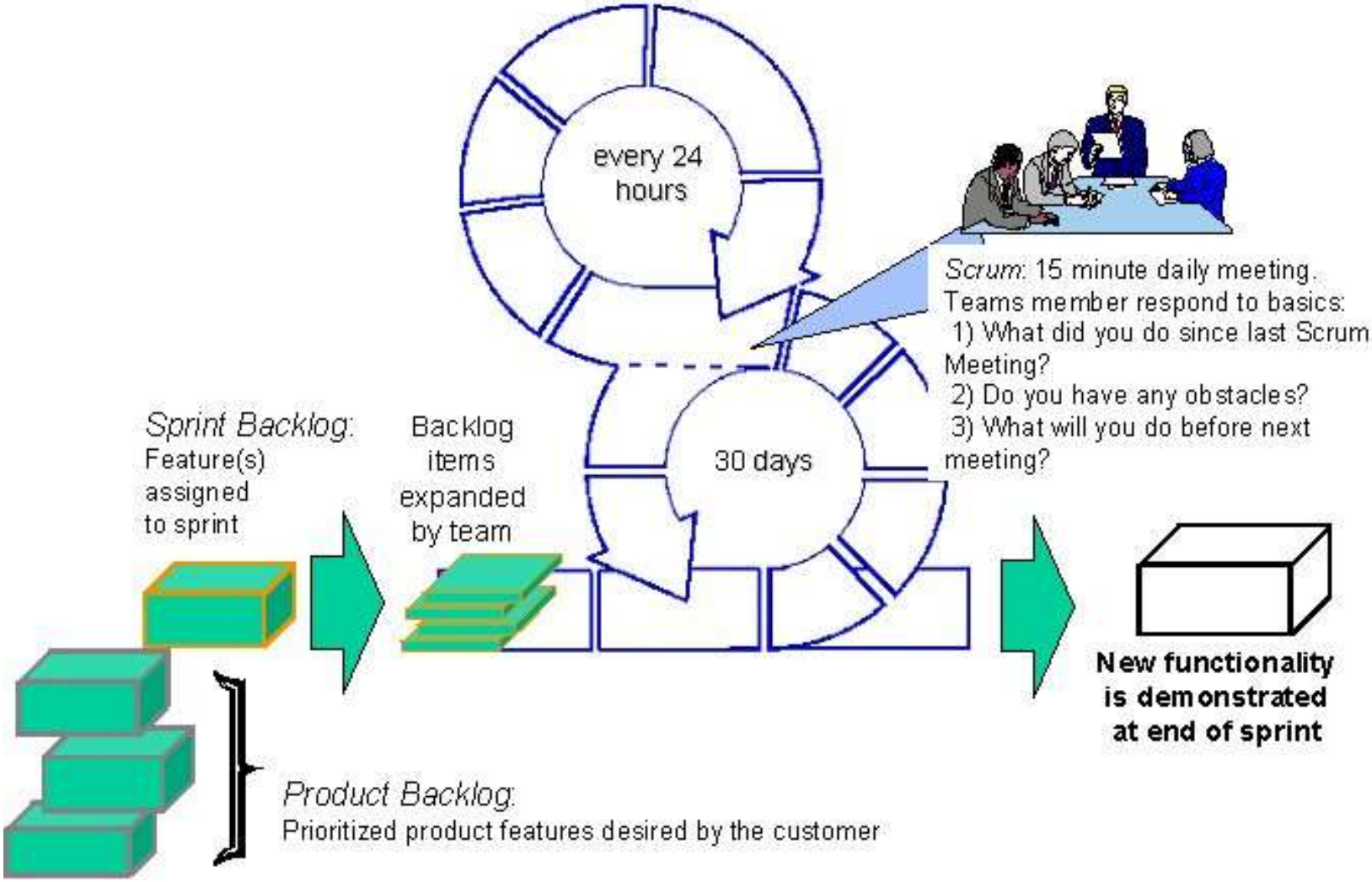
פיתוח איטרטיבי / אינקרמנטלי



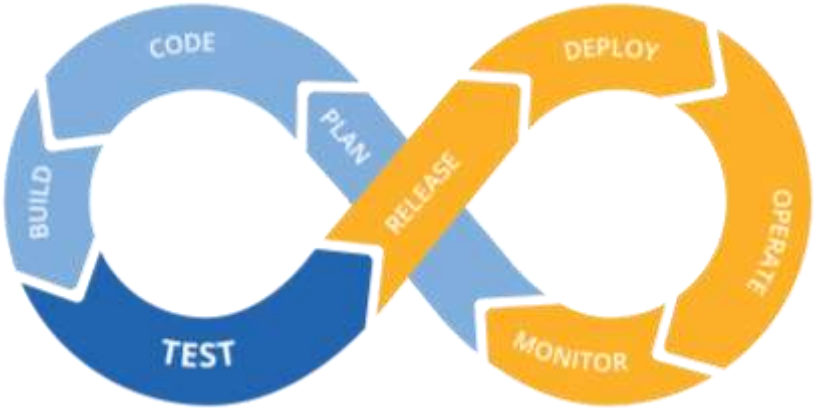
תהליכי פיתוח "זריזים" (Agile Processes)

- המודלים הקודמים הינם "מוכווני-תוכנית" (plan-driven)
- טענה
  - פיתוח תוכנה אינו יכול להיעשות על פי תוכנית "קשיחה", כי
    - קשה לראות מראש את כל התמונה
    - האפיון משתנה באופן דינמי ובקצב מהיר
    - תוכנה גמישה לשינויים ויש בכך יתרון
    - היעדר פאזת ה"ייצור" מאפשרת מחזורים מקוצרים של פיתוח ואספקה
- ולכן
  - נדרש תהליך פיתוח גמיש וזריז שיוכל לתמוך בפיתוח מוצרי תוכנה בכפוף לתנאים הנ"ל

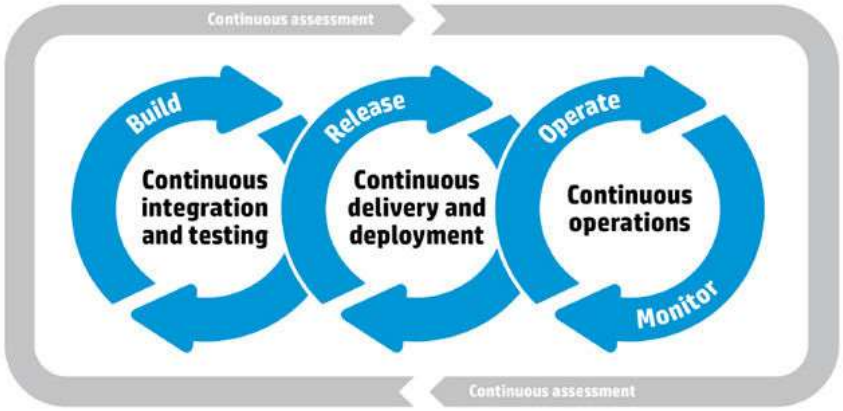
פיתוח אג'ילי - SCRUM



DevOps – תהליך מחזורי רציף (ואינסופי):
פיתוח (Development) ותפעול בשטח (Operations)



Picture Source: http://cdn.tricentis.com



Picture Source: https://www.linkedin.com/pulse/what-really-devops-does-benone-bitencourt

הקשר בין מבנה הקורס למחזור החיים

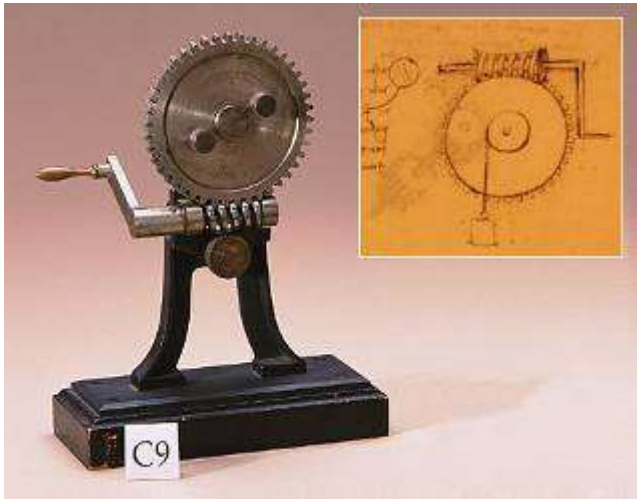
- במהלך הקורס נעבור **סבב אחד מלא** של מודל V, על מנת לערוך היכרות עם
  - רמות העניין השונות בפיתוח מערכות תוכנה
  - מודלים לתיאור אספקטים שונים של הפיתוח
  - טכניקות, כלים ושיקולים ברמות שונות
    - חלופות ארכיטקטורה
    - דפוסי תכן (design patterns)
    - שימוש ברכיבי מדף ושימוש חוזר ברכיבים
  - המעבר מרמה לרמה תוך שמירת עקביות
  - רמות אינטגרציה ובדיקות
- יש לזכור שפיתוח של מוצר במציאות, לאורך מחזור החיים שלו, נעשה בסבבים שונים
  - מלאים / חלקיים
  - ארוכים / קצרים
- בחלק האחרון של הקורס נדון בראיה הכוללת של מחזור החיים
  - פיתוח איטרטיבי, פיתוח זריז (agile), אבולוציה של המוצר
  - הבטחת איכות במחזור החיים
  - פעילויות תומכות בפיתוח
  - אחזקה

Model Based Development – פיתוח מבוסס מודלים

- Modeling
  - A means to capture ideas, relationships, decisions and requirements in a well-defined notation that can be applied to many different domains

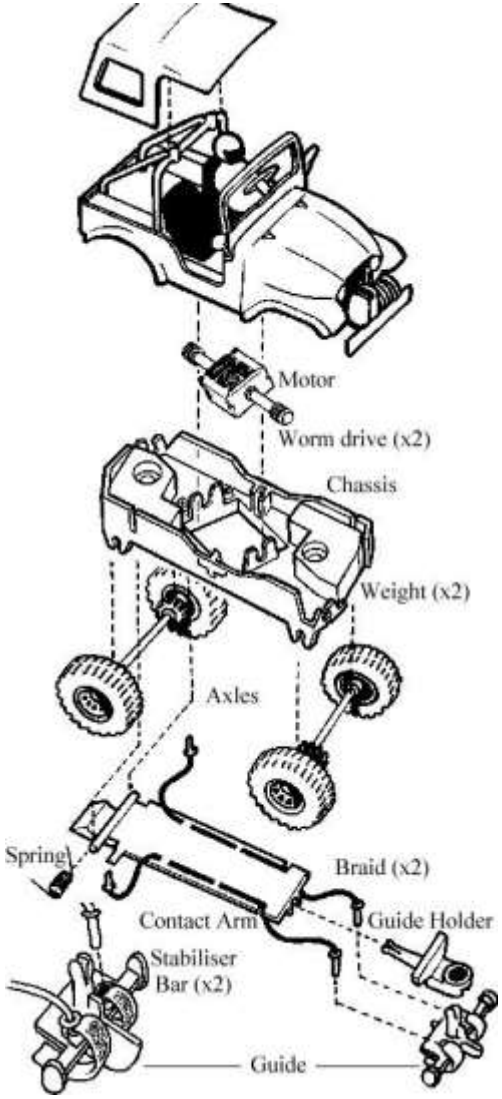
    [Pilone, D., *UML 2.0 in a Nutshell*, O'REILLY®, 2005

- מודלים משמשים לתיאור מפושט (simplified) ומופשט (abstract) של ישויות מורכבות
  - מודל מתמקד באלמנטים העקרוניים ללא ירידה לפרטים
  - מודל דורש "תרגום" לישות האמיתית
  - במודל יש דרגות חופש לפרשנויות שונות

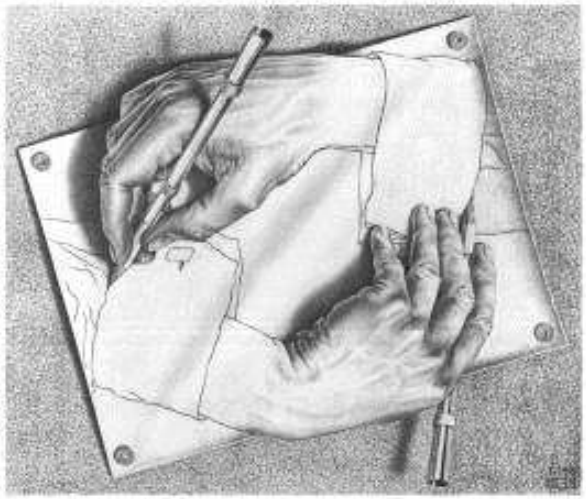


השימוש במודלים

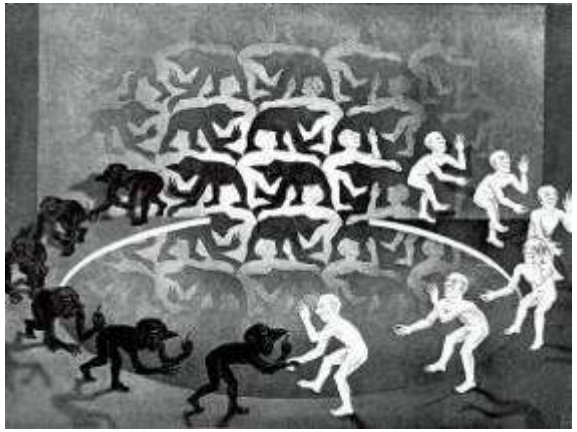
- מודלים משמשים בשני כיוונים
  - **מידול לפנים**: מידול לפני המימוש
    - סקיצות של רעיונות חדשים
    - סיעור מוחות לגבי פתרונות
    - הערכת חלופות פתרון
    - הנחיית הפיתוח
  - **מידול לאחור**: מידול לאחר המימוש
    - תיעוד של המערכת "כפי שהיא" (as built)
    - הסברת המערכת לאחרים
    - תמיכה בייצור / תחזוקה / שדרוג של המערכת
    - שימוש חוזר ב"מידול לפנים" של פרויקטים עתידיים



מודלים סטטיים ומודלים דינמיים



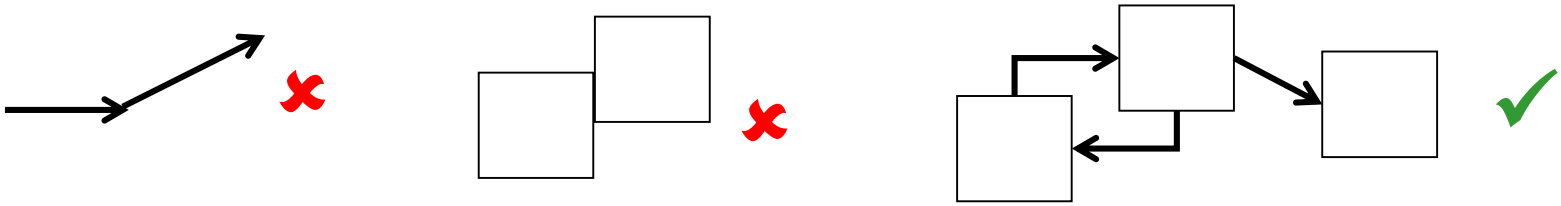
- **מודל סטטי / מבני**
  - מודל המתאר ישויות וקשרים ביניהם. דוגמאות:
    - מבנה ארגוני
    - שרטוט של חלק מכני
    - מבנה מולקולרי
    - טבלה (רלציה) בבסיס-נתונים
- **מודל דינמי / התנהגותי / תפקודי**
  - מודל המתאר זרימה של תהליך או שינויי מצב. דוגמאות:
    - תרשים זרימה / אלגוריתם
    - גרף של פונקציה על ציר זמן
    - אוטומט (במדעי המחשב)
    - אנימציה / סימולציה
- **מודל יכול להיות גראפי, טקסטואלי או משולב**



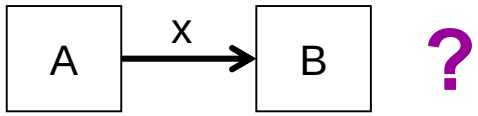
מידול - שפה גראפית

- אוסף של סמלים חוקיים (אלף-בית):

- צירופים חוקיים (תחביר):



- משמעויות (סמנטיקה):



"מחשב A מחובר למחשב B באמצעות ממשק x"

"תהליך A מעביר מידע x לתהליך B"

- כושר ביטוי (expressiveness):

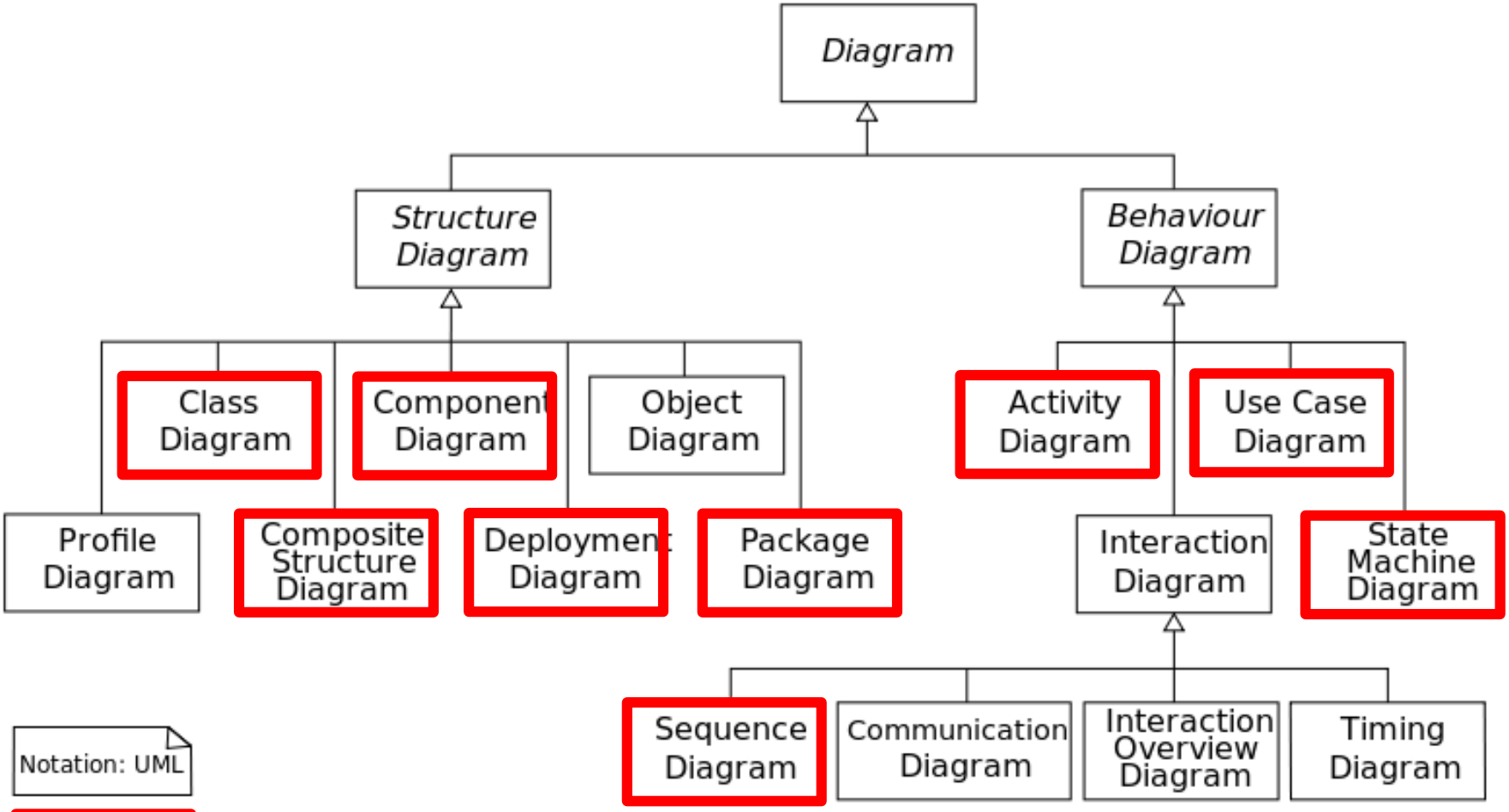
"מחשב A מחובר למחשב B באמצעות ממשק x"

"מחשב A מחובר לפחות למחשב אחד מסוג B, אבל לא ליותר מ-10 מחשבים מסוג B בו-זמנית"

UML = Unified Modeling Language

- שפה (גראפית) לניתוח ולתכן מונחה עצמים (OOA&D)
  - גרסה נוכחית – UML 2.5 (יוני 2015)
- מכילה אוסף של כלי מידול לתיאור היבטים שונים של התוכנה
- התפתחה תוך כדי מיזוג של מספר שיטות:
  - Grady Booch, 1991-1996
  - James Rumbaugh (OMT), 1992-1996
  - Ivar Jacobson (Objectory), 1992-1997
- פיתוח השפה וכלי CASE בחברת Rational (IBM)
- 1997: אומצה כתקן בפועל (ad-hoc standard) ע"י OMG
  - OMG = Object Management Group
    - מעין "ועדה ממליצה" בחסות 800 חברות וארגונים

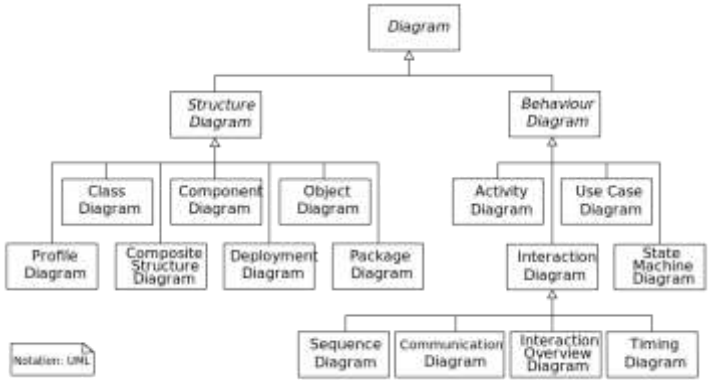
משפחת המודלים של UML

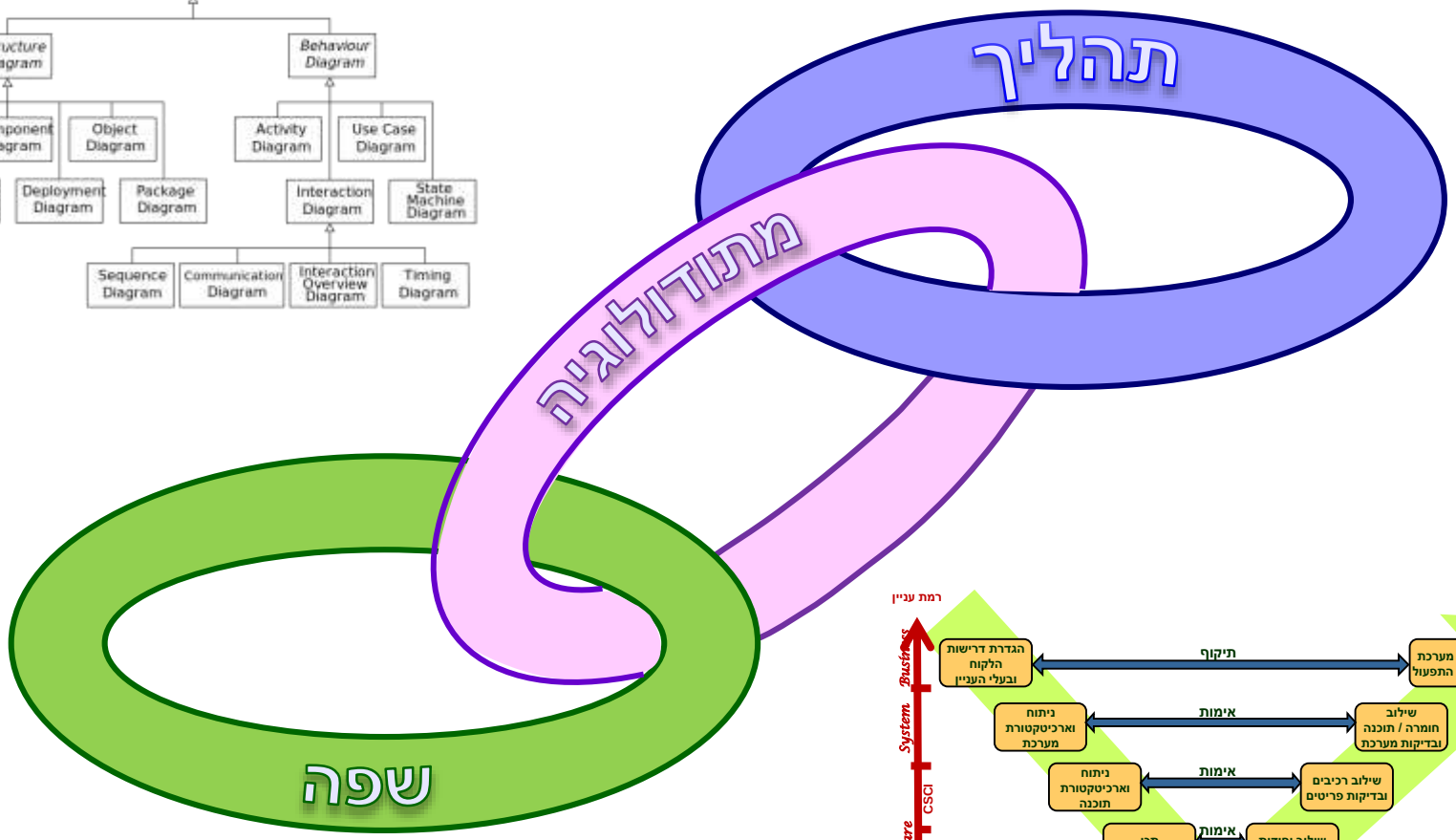


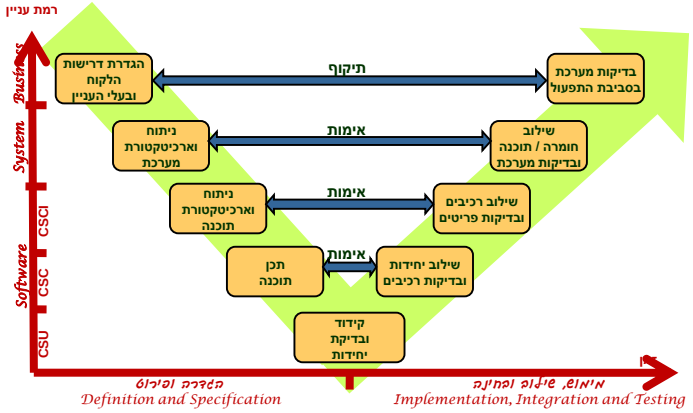
Notation: UML

מודלים בהם נעסוק בקורס זה

מתודולוגיה: החוליה המקשרת בין שפה לתהליך







- השאלות העיקריות עליהן עונה מתודולוגית MBSD
  - מתי וכיצד משתמשים בכל סוג של מודל?
  - כיצד שומרים על עקביות בין המודלים?