ניהול פרויקטי תוכנה

SOFTWARE PROJECTS MANAGEMENT

2021

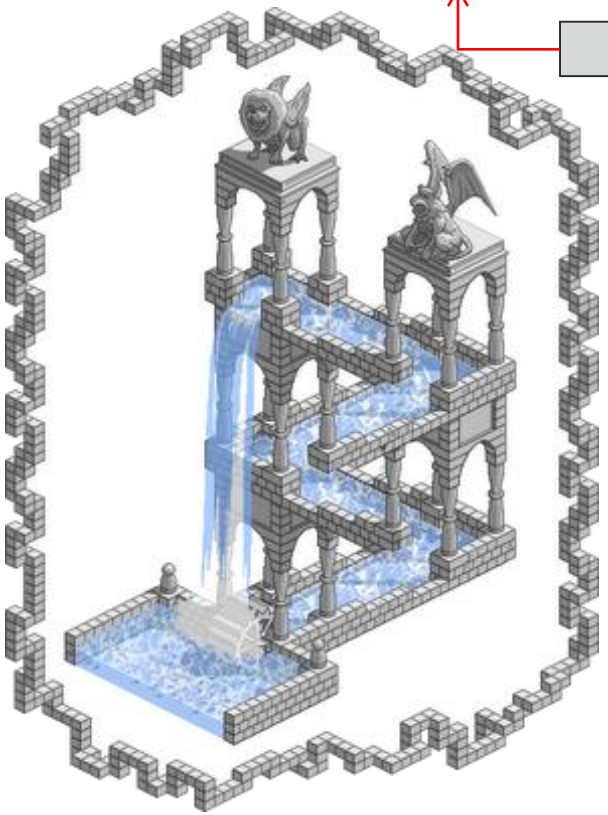
Intro (Lesson#2)

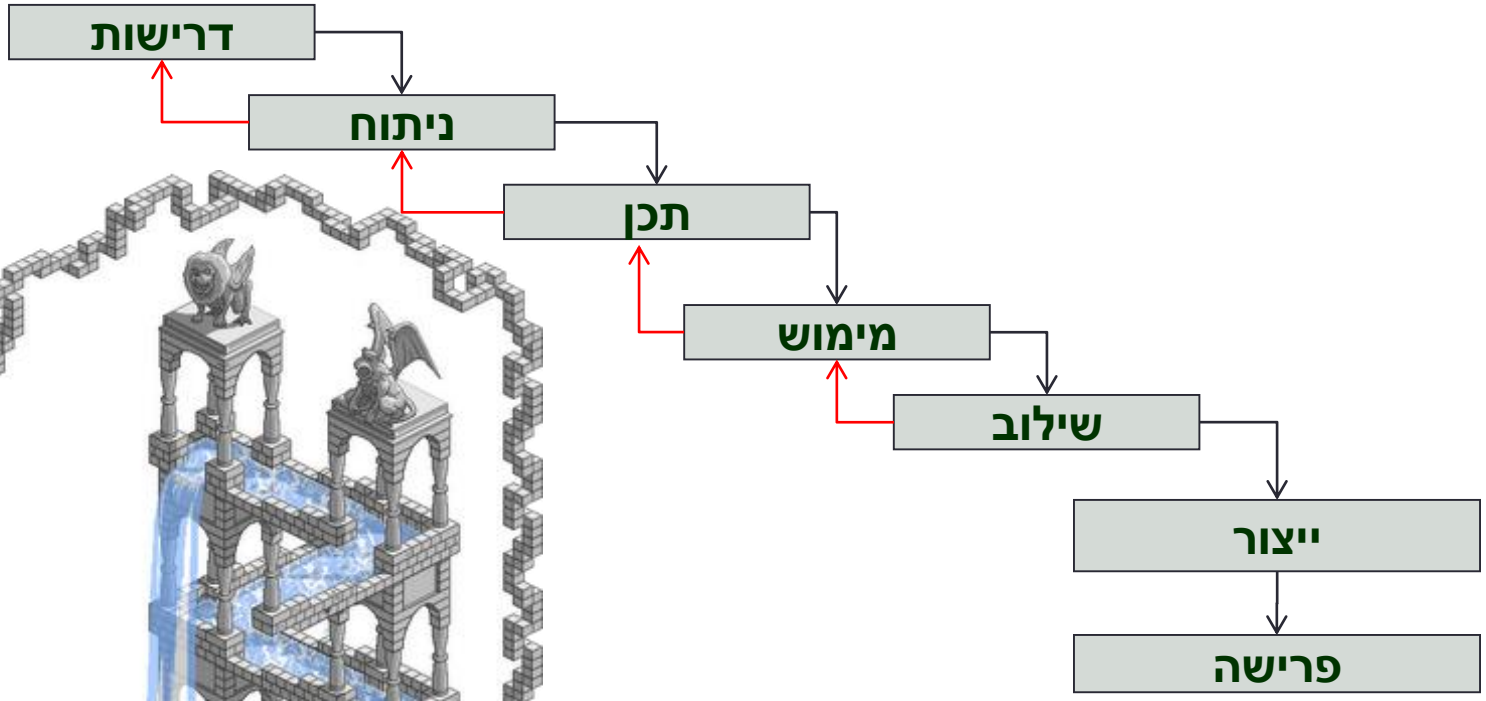
דר' הדס שוורץ-חסידים

המכללה האקדמית להנדסה ע"ש סמי שמעון

תהליכי פיתוח תוכנה- ריענון...

Waterfall Royce 1970





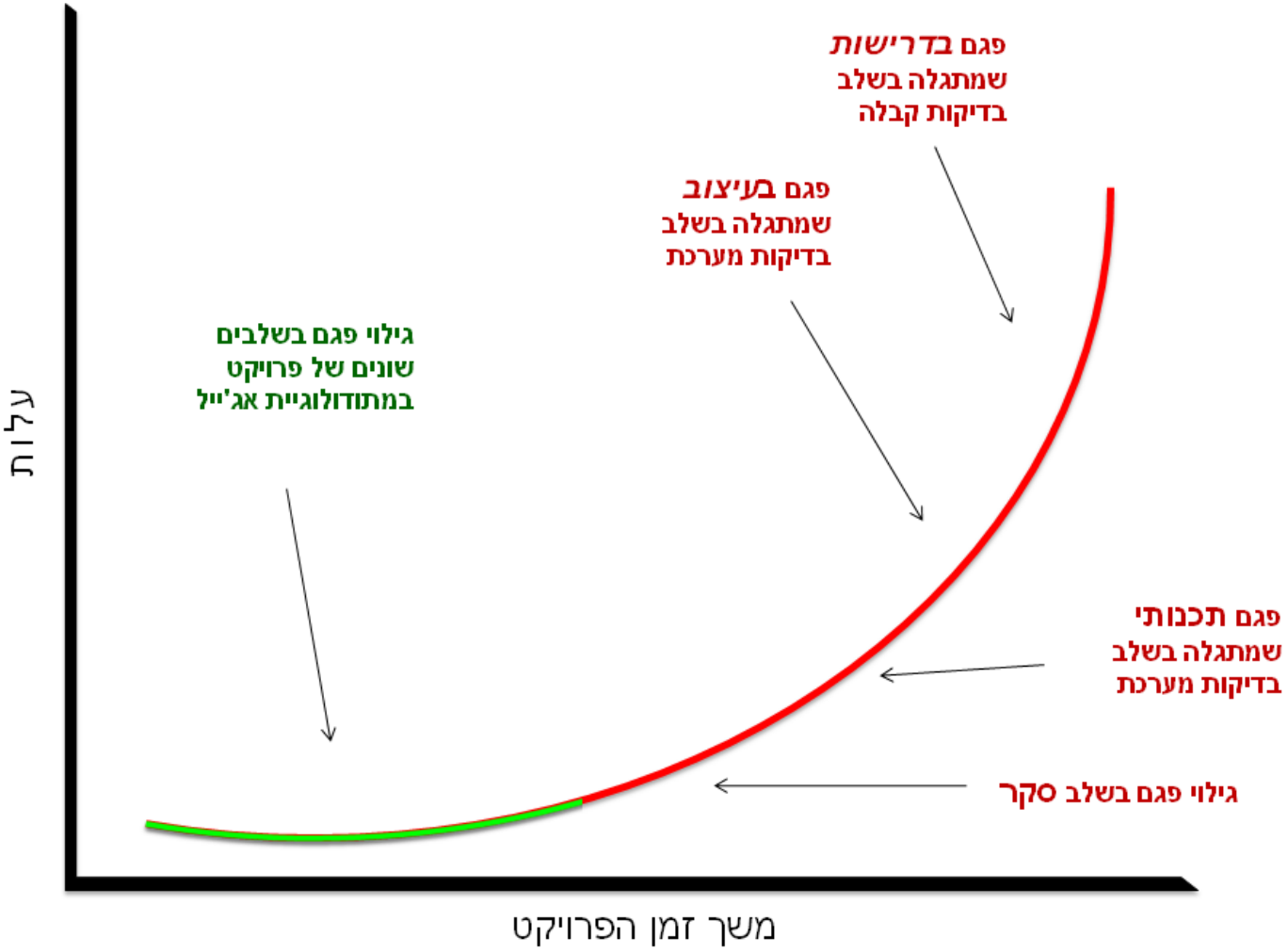
מהו משבר התוכנה, מה היו הגורמים העיקריים?

שורשי משבר התוכנה (תזכורת...)

- מורכבות
- ציפיות מוגזמות
- תכנון לקוי
- מטרות ויעדים לא ברורים
- יעדים משתנים
- שינוי דרישות
- הערכת זמנים ועלות לא מציאותיים
- חוסר שיתוף פעולה
- ליקויים בתקשורת ובפעולות צוות
- חוסר מיומנות

Tilmann, George and Weinberger, Joshua (2004) Technology never fails, but project can [Online journal] Baseline, Volume 1, Issue 26, page 28.

עלויות גילוי השגיאות לאורך SDLC



גישת ה"זמיש" Flexible+Agile

- **אנשים ויחסי גומלין על פני תהליכים וכלים.**
- **תוכנה עובדת על פני תיעוד נרחב.**
- **שיתוף הלקוח על פני משא ומתן חוזי.**
- **גמישות לשינויים על פני הצמדות לתוכנית.**

is all about people and not solutions based on technology.

עקרונות המנשר Agile

הגישה הזריזה אינה מתודולוגיה אחת, אלא אוסף של שנים עשר עקרונות כלליים:

1. עדיפות הגבוהה ביותר של המפתחים היא לספק ולרצות את הלקוח בכך שיספקו לו תוכנה משמעותית העונה על דרישותיו.
2. גמישות לשינויים –שינויים מתקבלים בברכה.
3. מתן תוצרים ללקוח- מהיר מוקדם ככל האפשר, מתמשך.
4. משוב מידי בזמן אמת - שמירה על תקשורת רציפה עם הלקוח ושיתופו בתהליך.
5. צוותים מיומנים, מגובשים בעלי מוטיבציה גבוהה, בסביבת העבודה הטובה ביותר האפשרית, ועם כלים מתאימים.

עקרונות המנשר Agile

6. צוות פיתוח טוב הוא צוות שהתקשורת בין חברי הצוות ושיתוף הפעולה בניהם טוב ויעיל.
7. תוכנה עובדת הוא המדד העיקרי להתקדמות נכונה בתהליך הפיתוח.
8. לא ניתן להאיץ פיתוח תוכנה, המטרה היא לשמור על קצב פיתוח שישאיר את רמת הפיתוח איכותית ביותר.
9. כל תוכנה שיוצאת לשוק מתוחזקת גם לאחר מסירתה ללקוח, מדי פעם יש צורך בשינויים קטנים וליטוש התוכנה על מנת שתמשיך לשרת את העסק, עיצוב וקידוד טוב במהלך הפיתוח יאפשרו שיהיה יותר נוח וקל לתחזק ולהבין את הקוד בעתיד.

עקרונות המנשר Agile

10. התכנון צריך להיות פשוט ולכלול רק את מה שהלקוח ביקש ולא יותר. ככול שהתכנון יהיה פשוט יותר כך גם הבדיקות יהיו יותר קלות ויעילות.
11. אנשי צוות צריכים לעבוד בתאום מלא ויש להטמיע בארגון עבודה בשיפור מתמיד.
12. הצוות מקיים פגישות באופן קבוע וכל אחד מחברי הצוות יכול לעלות בפני כולם בעיות בתהליך, כך הצוות בוחן את הרגלי העבודה שלו ואת התקדמותו בתהליך והצוות עצמו הוא זה שמשפר את עצמו ומתאים את דרכי העבודה לגישת ה agile.

Considerations for the decision Agile/Waterfall

1. Requirements changes- שינויים בדרישות
2. Scope changing -שינויים בתכולה
3. Experience -ניסיון של חברי הפרויקט
4. Resources/dedicated/part time -הקצאת משאבים
5. Physical location(the same?)-מיקום מבוזר\מרוכז
6. Customer involvement and feedbacks -מעורבות לקוח
7. Time line (fixed/flexible)-לוחות זמנים
8. Documentation (reduced or full?) -סוג התיעוד
9. Environment -סביבת פיתוח

ניהול תצורה עסקי – הקשר עם הלקוח

- **ניהול שינויים**
  - האם מתבצע רישום שוטף של פניות לקוחות?
  - מי יוזם את השינוי?
  - איך יוזמים ?
  - כיצד בודקים את משמעות השינוי?
  - עם מי מתייעצים?
  - את מי מעדכנים על השינוי?
  - מי מאשר שינוי?
  - האם ואיך מתעדפים?
  - האם יש נציג לקוח בצוות שינויים?
- **שילוב במהדורות**
  - האם הלקוח מעורב בתכולת מהדורה ותזמונה?
  - האם הלקוח מעורב בניהול דרישות ומשימות?
  - האם הלקוח מעורב באבני דרך?

ניהול ובקרת סביבות הפיתוח

סביבות פיתוח

- היכן מבוצע הפיתוח?
- מהי סביבת הפיתוח?
- שיטת ניהול שטחי עבודה וספריות?
- מהי השיטה לזיהוי מרכיבי תוכנה – תצורת שמות קבצים ורכיבים?
- זיהוי מהדורות – תכולה
- שיטת בניית מהדורה
- האם קיים כלי לבקרת תצורה
- כיצד מתבצעת בקרת תצורה בסביבת הפיתוח – נעילות, check in/out
- כיצד משלבים תוכניות שמפותחות על ידי מפתחים שונים (אחריות, שיטת עבודה, העברה פיסית)?

שילוב במהדורות

- איך מאופיינת מהדורה בסביבת הפיתוח?
- איך, באופן פיזי, מתבצעת ההעברה מסביבה הפיתוח לניסוי?
- איך ואם מתבצע תהליך העברת Sources בעת סגירת מהדורה ואיך הם מועברים כשיש תיקונים לאחר בדיקות?
- איך משתלב מנהל המערכת/מנהל הפיתוח באישור הסופי של המהדורה לפני העברתה לסביבת הניסוי?

שיקולים עסקיים לשימוש בפיתוח "זמיש"

- הפחתת עלות הפיתוח
- שיפור אמינות המוצר
- שיפור באיכות ועמידה בתקציב
- ניהול שינויים קל יותר
- הפחתת זמן יצור
- ווידוא שהמוצר תומך באינטרס הלקוח ולא פוגע בו
- מספק מוטיבציה, שביעות רצון ושיפור בתקשורת של אנשי הצוות
- נותן תועלת הן ללקוח והן לצוות המפתח

תכונות עיקריות Agile

- איטרטיבי
- מעורבות הלקוח בתהליך הפיתוח
- 80/20 כאשר 20% מהתכונות העיקריות ימומשו ב-80% מהזמן של תהליך הפיתוח.
- זמן קבוע בכל איטרציה.
- תיעדוף
- הסתגלות לדרישות לקוח חדשות בכל עת.
- עבודה בצוותים
- יותר זמן מוקדש לפיתוח ולבדיקות מאשר לתיעוד ורישום שאינם קשורים לפיתוח.
- תהליך פיתוח מהיר – נעשה במהירות תוך שימוש בטכנולגיית פיתוח קלת משקל.
- יותר משמעת – בגלל שתהליך הפיתוח נעשה לפי הגישה הזריזה שמים יותר דגש על משמעת בצוותים ודיוק בזמנים ובארגון.
- פשטות – שמירה על תהליכים פשוטים ככל האפשר ופתוחים לשינויים (גמישים).

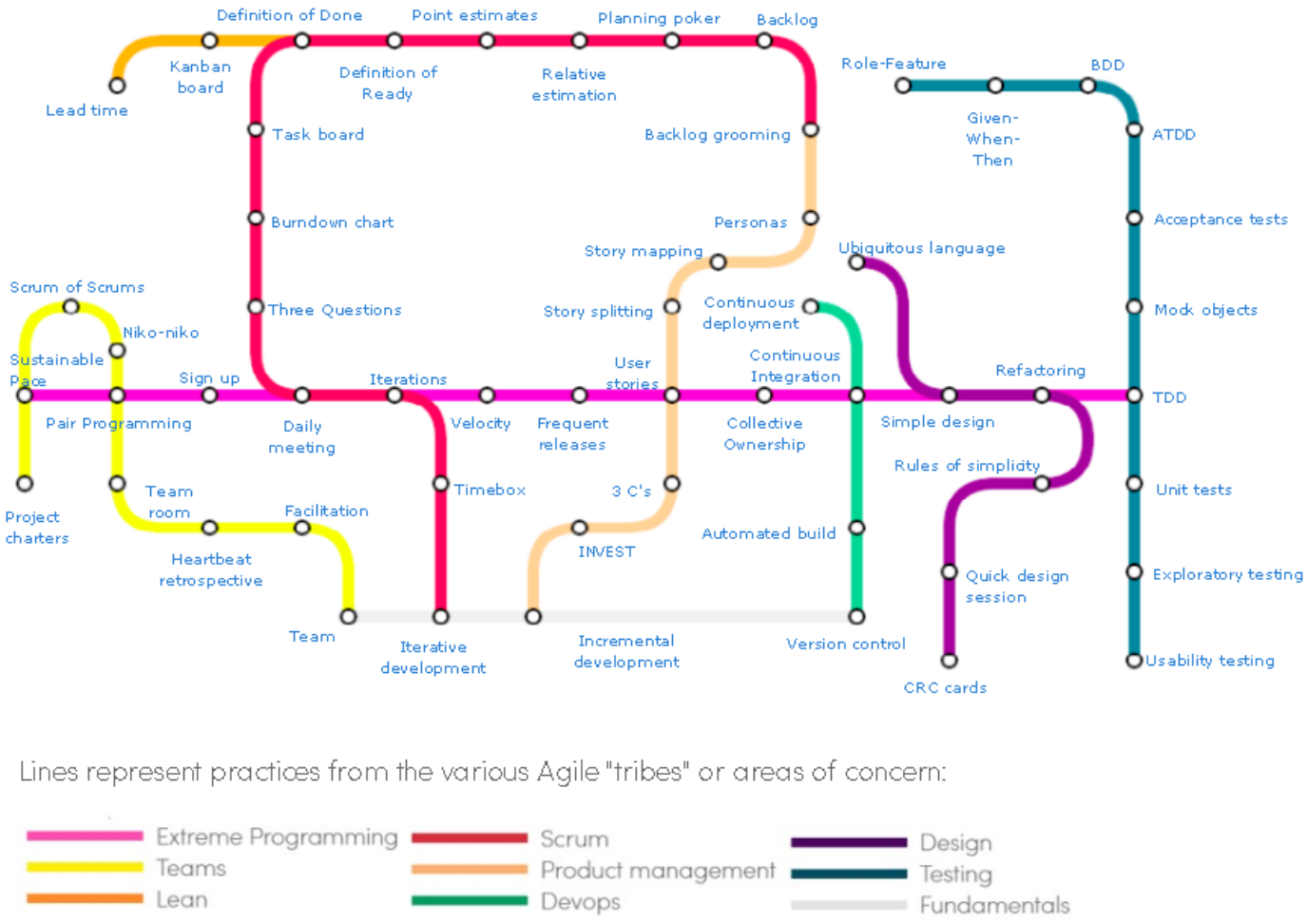
תועלת ללקוח Agile

- הלקוח מעורב יותר בפיתוח ומקבל עדיפות גבוהה.
- הלקוח מודע למצב ההתקדמות בתהליך הפיתוח בכל עת.
- הדרישות מוסכמות אחרי כל איטרציה.
- זמן השיווק ללקוח קצר יותר והוא יכול להתחיל לעבוד עם הגרסה הראשונית של המוצר.
- לוח הזמנים מוגדר מראש והלקוח יכול לדעת מתי הוא אמור לקבל כל גירסה.
- מקדישים יותר זמן לבדיקות כך שהמוצר שהלקוח מקבל הוא איכותי יותר.

Agile תועלת לצוות הפיתוח

- צוותי הפיתוח מעורבים בכל שלבי התהליך ויכולים לקבל החלטות יעילות יותר.
- בגלל שתהליך הפיתוח הוא מצטבר הצוותים יכולים להתמקד בדרישות ספציפיות הרלוונטיות לאותה נקודת זמן.
- התמקדות בפיתוח המוצר ולא במסמכים שונים.
- הצוותים מקבלים יותר משובים על עבודתם מהלקוחות
- פחות זמן מתבזבז על איסוף דרישות...
- פחות זמן לתכנון.
- פחות עלויות עבור דברים שאינם קשורים לפיתוח.
- הפיתוח נעשה בצוותים תוך כדי שיתוף פעולה.

Agile Subway Map



משפחת Agile

- גישה\פילוסופיה ולא מתודולוגיה
- Scrum
- Crystal
- XP
- Lean
- Kanban
- FDD

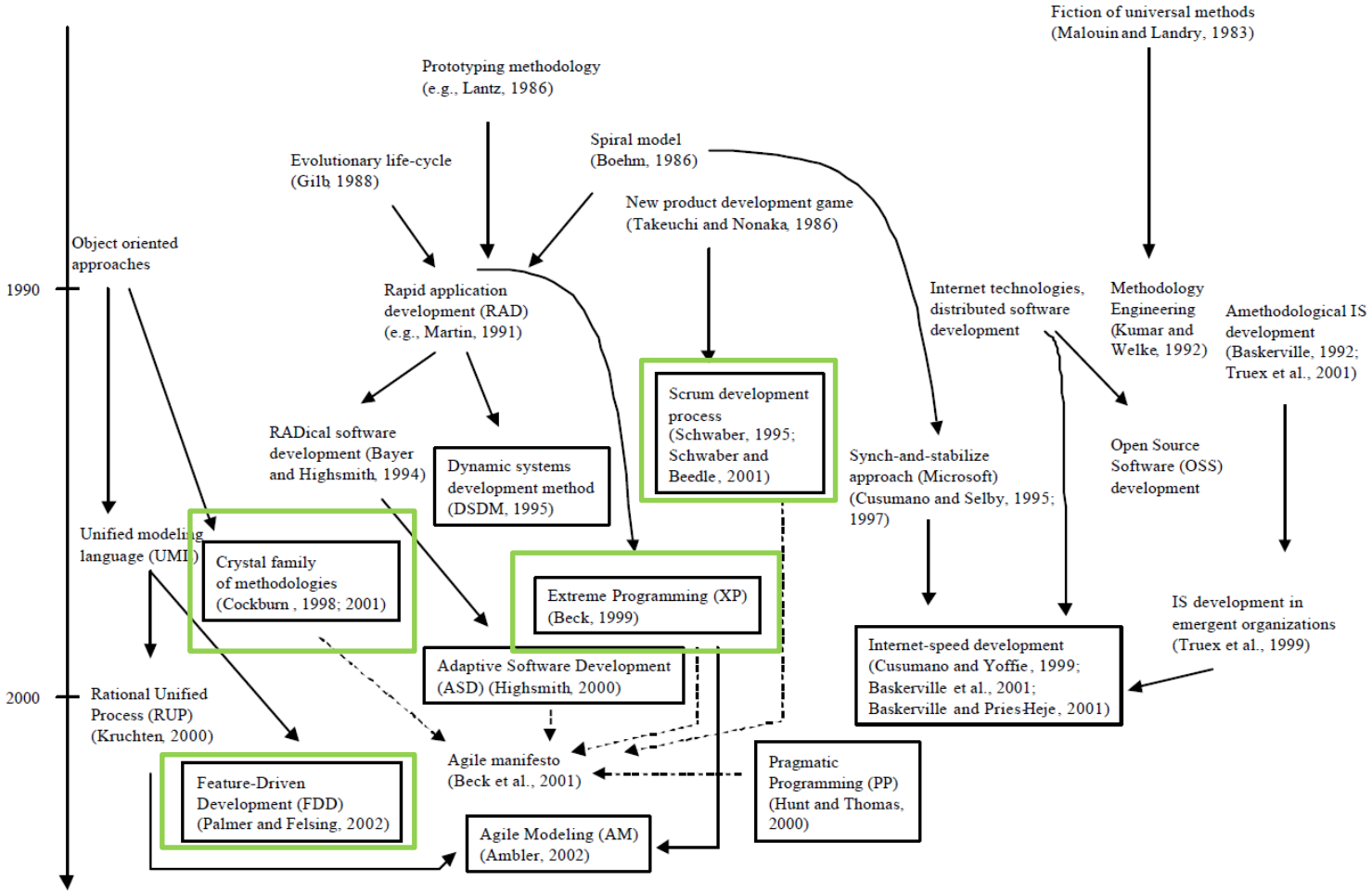
השוואה בין מתודולוגיות

	מתודולוגיות זמישות	מתודולוגיות "כבדות משקל"
השיטה	התאמות ושינויים תוך כדי פיתוח	תכנון מראש
מדד להצלחה	רווח עסקי	עמידה בתכנון
גודל פרויקט	קטן עד בינוני	גדול
ניהול	שונה משיטה לשיטה אך בד"כ מבוזר	אוטוקרטי
תחום	בלתי צפוי	צפוי מראש
שינויים	גמיש לשינויים	הימנעות משינויים
מחזוריות	מספר מחזורים רב	מוגבל
תכנון ארוך תווך	מינימאלי	גדול/מקיף

השוואות בין מתודולוגיות

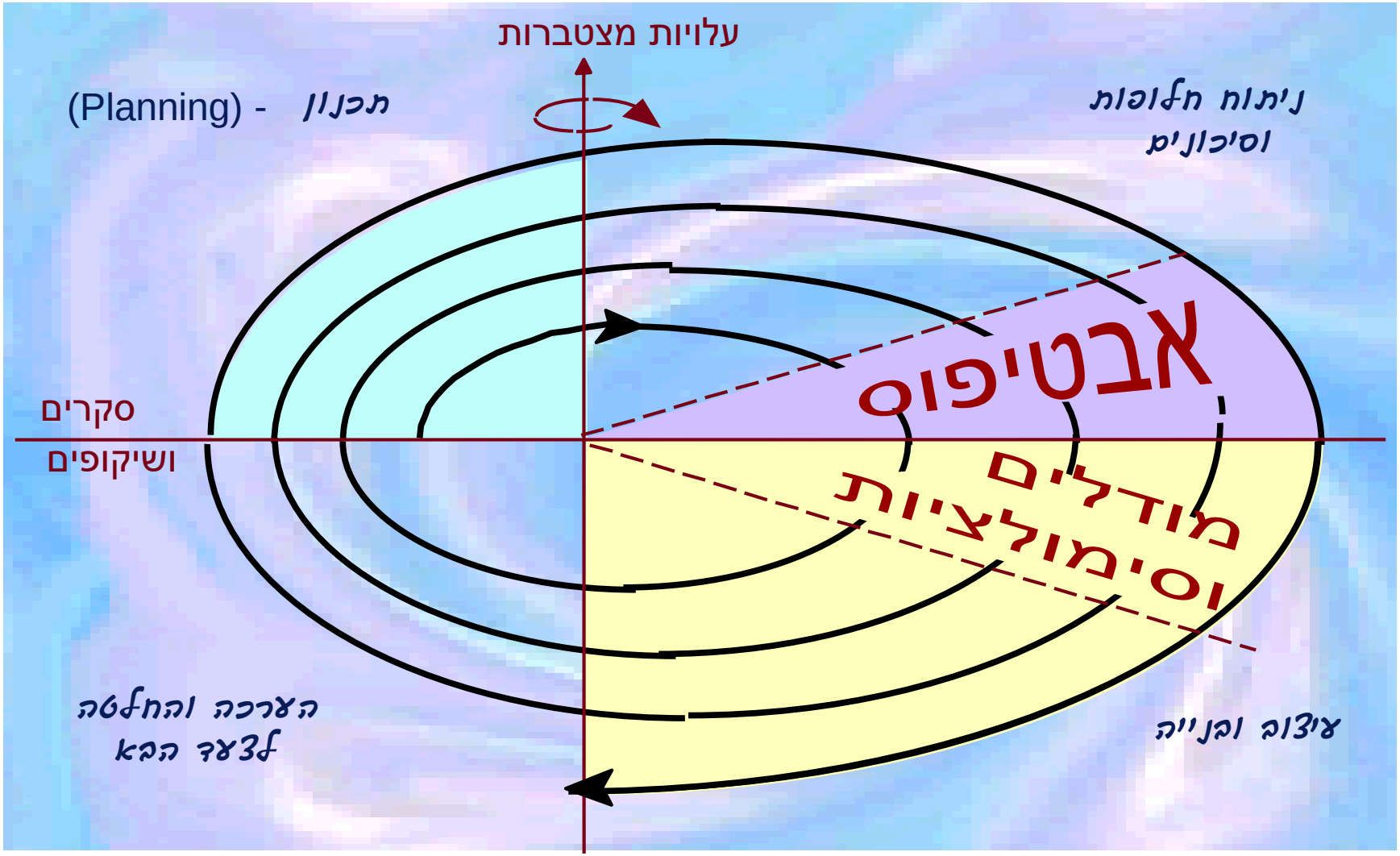
	מתודולוגיות זמישות	מתודולוגיות "כבדות משקל"
החזר השקעה	בתחילת הפרויקט	בסוף הפרויקט
גודל צוות	קטן	גדול
מיומנות אנשי צוות	הכרחית	עדיפות
דרישות	נתון לשינויים	יציב
ארכיטקטורה	משתנה ומותאם לדרישות העדכניות	יציב ומתוכנן מראש
שינויים	גמיש לשינויים	הימנעות משינויים
תיעוד	רק מה שנדרש	חשוב והכרחי
Refactoring	לא יקר	יקר

המפה האבולוציונית של השיטות הזמישות



Abrahamsson P. et. Al. “New Directions on Agile Methods: A Comparative Analysis” Published in the Proceedings of the International Conference on Software Engineering, May 3-5, 2003, Portland, Oregon, USA

המודל הספירלי- גישה שונה לבעיה



המודל הספירלי

- המערכת נבנית בסבבים ובהדרגה
- כל סבב פיתוח נחלק למקטעים ברורים: תכנון, חקר ישימות, בנייה, בחינה והערכה
- התמקדות ב"ליבה" של מחזור החיים: אפיון מפורט, עיצוב ובנייה, שילוב, בדיקות, דגמים ומעורבות המשתמש
- שמות אחרים:
  - Iterative / Incremental Development

*You should use iterative development only on projects that you want to succeed…" (*Martin Fowler)

המודל הספירלי חוזקות

- מספק אינדקציה מוקדמת של סיכונים
- המשתמשים נחשפים למערכת בשלבים מוקדמים
- קודם כל מפתחים פונקציות קריטיות ובעלות סיכון גבוה
- העיצוב לא חייב להיות מושלם
- המשתמשים יכולים להשתלב בצורה הדוקה לאורך כל מחזור החיים
- משובים תכופים ומוקדמים
- הערכת עלות לעיתים קרובות

המודל הספירלי – חולשות

- מודל מורכב
- השקעת זמן בהערכת סיכונים בפרויקטים קטנים או בעלי סיכון נמוך
- השקעת זמן מוגזם בתכנון, ניתוח סיכונים ואבי טיפוס
- נדרשת מומחיות בהערכת סיכונים
- ספירליות ללא סוף...
- קושי בקביעת אבני דרך ונקודות צומת
- תוכנה חלקית, לא בהכרח עובדת

Scrum

Scrum

- Scrum היא טכניקת פיתוח בעבודת צוות מרוכזת בסבבים קצרים.
- המונח Scrum לקוח מעולם הרוגבי, ולפיכך סבבי הפיתוח נקראים ספרינטים.
- **כל ספרינט אורך בין שבועיים לחודש.** אין שינויים במהלך הספרינט !
- בסקראם נדבר על: ניהול עצמי של הצוות, תיעדוף, שיתוף פעולה, שחרור גרסאות וסיפוק רצון הלקוח.

"It’s all about people and not solutions based on technology…”

מדוע Scrum?

- שיטה מוכחת וממומשת בפרויקטים רבים.
- מסייעת לגיבוש צוותים.
- אינטואיטיבית יותר להטמעה.
- המתחרים שלהם עובדים ככה.

Scrum - Roles

Scrum master

- אחראי לכך שהפרויקט מנוהל ומיוצר ע"פ כללי ה-scrum.
- יוצר קשר עם צוות הפיתוח כמו גם עם הלקוח עצמו ועם המנהלה.
- מוביל ולא מנהל!
- מנחה הקבוצה ומוביל תהליך הפיתוח מתחילתו ועד סופו.
- "מסיר מחסומים" ומונע הפרעות שעלולות להזיק לצוות או לתהליך הפיתוח.

Scrum team

- בודדים האחראים לתוצאות הספרינט.
- בעלי אוסף של כישורים השייכים לדיסציפלינות שונות.
- בד"כ כ 6-8 אנשי צוות.
- דוגל בניהול עצמי.
- אינו משתנה בזמן המאוץ, לא מבחינת הרכב הצוות ולא מבחינת הדרישות !

לקוח Product Owner

- האיש האחראי על הפרויקט, בקרת המוצר.
- לו המילה האחרונה ב-Product Backlog.
- האיש האחראי על רווחיות המוצר.
- קובע ומתאם רשימת דרישות לכל ספרינט.
- קובע תיעדופים וקדימויות בכל ספרינט.
- מקבל או דוחה את תוצאות העבודה

תיעדוף

1. Financial value
2. cost
3. new knowledge
4. risk removed

Risk	High Risk, Low value AVOID	High Risk High Value Do second
	Low Risk Low Value Do last	Low Risk High Value Do first
		Value

כיצד מעריכים עבודה

- תהליך WBS story points
- פירוק סיפורי משתמש למשימות- הערכת זמן והתקדמות
- עדכון שוטף בכלי (JIRA)
- Log work –daily

Product Backlog & Sprint Product Backlog

- **Product Backlog (PB)** – "רשימת מכולת" של כל הדברים שאנחנו עשויים לעבוד עליהם. **Sprint Product Backlog** – רשימת המשימות שנעבוד עליהם בספרינט הקרוב. **נקרא גם SPB\PBI.**

כיצד מחליטים על דרישות ?!

- **כדי להגדיר דרישות יש ללכת לפי שיטת 3 ה-W:**

1. **Who** – מי מבצע את הפעולה.
2. **What** – מה מבצעים.
3. **Why** – למה מבצעים.

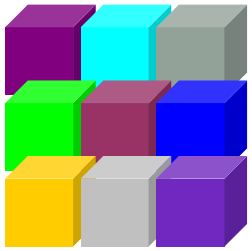
דוגמא לסיפור משתמש:

המרצה יוכל לעדכן את ציוני הסטודנטים אונליין ,

כך שהוא לא יזדקק לעזרת מזכירת המחלקה.

תמצית תהליך ה- Scrum

Product backlog

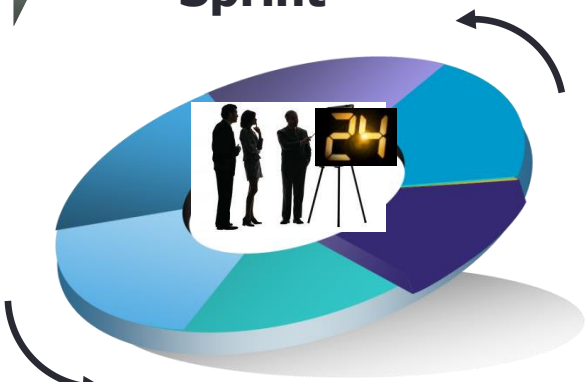


דרישות פתוחות למוצר, כולל תעדוף

Sprint backlog

הדרישות אשר נכללות בספרינט הקרוב

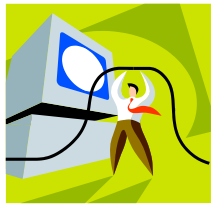
Sprint



2-4 שבועות

נעילה של הדרישות במהלך הספרינט

Working Increment of the product



אין חובה למסור בסיום ספרינט

Daily Scrum Meeting

- פגישה במעומד למשך 20 דקות.
- כל חבר בצוות צריך לענות על 3 שאלות:

1. **מה ביצעתי מאז הפגישה האחרונה?**
2. **מה אבצע לקראת הפגישה הבאה?**
3. **מה מונע ממני להשלים המשימות?**

- מטרת המפגש הוא עדכון כל הצוות ולטובת כל הצוות.

Sprint Planning (SP) Meeting

- משתתפים ה- Product Owner והקבוצה
- סקירת ה- Product Backlog
- מו"מ למה ייכלל במאוץ
- ה- Product Owner מספק הגדרות ופרטים לכל התכונות (Features) והפונקציות (Functions)
- קביעה על אילו פריטים (SPB) נתמקד בספרינט.
- לבדוק האם יתכנו צרכים נוספים שיש להוסיף ל- Product Backlog
- קביעת מטרות המאוץ והיעדים הפונקציונאליים
- רשימה ראשונית של משימות (Tasks) המאוץ
- תוצאות הפגישה יובילו להגדרת Sprint Goals

תכנות בזוגות

- טכניקה בפיתוח תוכנה זריז בה זוג מתכנתים עובדים יחדיו מול עמדה אחת.
- האחד מקליד את הקוד, בזמן שהשני סורק כל שורת קוד בזמן הקלדתה.
- מטאפורת "הטייס והנווט"\"הנהג והצופה".
- "הנווט"\"הצופה" תפקידו לאתר תקלות תחביריות\לוגיות בקוד, שקילת

  הכיוון האסטרטגי, הגיית רעיונות וצפיית בעיות עתידיות.
- הדבר מאפשר לטייס\לנהג לרכז את מלוא תשומת ליבו לקוד תוך שימוש

  ב"נווט" כמדריך ורשת ביטחון.
- **שניהם מחליפים תפקידים לעיתים תכופות כך שכל אחד מהם מעורב בקוד.**

תיכנות בזוגות- יתרונות

- מתכנתים העובדים בזוגות מייצרים תכניות קצרות ויעילות יותר.
- תכנון טוב יותר ופחות באגים מאשר עבודה ביחידים.
- ידע עובר בין זוג המתכנתים.
- רמת משמעת גבוהה יותר, מורל גבוה יותר וביטחון גדול יותר בקוד.
- ניהול זמן טוב יותר ופחות יבזבזו זמן בהתעסקות בדברים שוליים.
- **ידע על המערכת מתפשט בין כל חברי הצוות וכך מוקטן הסיכון הניהולי כאשר מתכנת עוזב את הצוות.**

תיכנות בזוגות -חסרונות

- מנהל צריך לאזן בין השלמה מהירה יותר של העבודה וירידת זמן בדיקות
  וחיפוש באגים לעומת עלות גבוהה יותר של ייצור קוד.
- במשימות פשוטות בהן הזוג מבין את המשימה, ציוות גורם לירידה בתפוקה.
- תכנות בזוגות יקר משמעותית מתכנות יחידי.
- **החסרונות מתגמדים לעומת היתרונות !**

Team Work &
Pair Programming

Extreme Programming (XP)

על פי שיטה זו מפתחים פונקציונאליות פשוטה ואחר כך מוסיפים פונקציונאליות בקצב עולה, תוך שיפור מתמיד של הקוד. מערכות המפותחות לפי שיטה זו גמישות מאוד לשינויים, וניתן להרחיבן בקלות ובאופן בטוח.

Crystal

שיטה זו מתייחסת ליעילות בפיתוח עם דגש ל"סבלנות" לשינויים . שיטה זו מפרטת עקרונות לתכנות נכון ומתמקדת יותר בעיצוב התוכנה וניהול הפרויקט.

FDD - Feature-Driven Development

המפתח להצלחה – תיעדוף תכונות

Lean

שיטה המתרכזת בהקטנת הבזבוז בתהליכי הפתוח.
צמצום הבזבוז גורם להגדלת האיכות, הקטנת זמן הייצור והקטנת ההוצאות.

ASD - Adaptive Software Development

מספקת מסגרת מקיפה של מושגים, קווים מנחים במטרה לעודד פיתוח איטראטיבי, אינקרמנטלי ומהיר במערכות גדולות ומורכבות תוך כדי ייצור מתמיד של אבי טיפוס.

חולשה	מאפיין ייחודי	נקודות מפתח	שם המתודולוגיה
תלות במיומנות וביכולת האישית ,אין דגש על תהליכי ניהול .	Refactoring Test Driven Development.	פיתוח מונחה לקוח - הלקוח שותף מלא בפיתוח. קבוצות קטנות שחרור גרסאות רבות.	**Xp**
אין מענה לאינטגרציה ולבדיקת קבלה.	Product backlog sprints	קבוצה קטנה , פיתוח תוך ארגוני, קבוצת פיתוח, תהליך חוזר ,מחזור עד 30 ימים	**Scrum**
עדיין לא יושמו ונבחנו כל השיטות	היכולת להתאים ולבחור את השיטה המתאימה ביותר כתלות בגודל הפרויקט והקריטקליות	משפחה של שיטות בעלי ערכי ליבה משותפים. כאשר הטכניקה החוקים הכלים והסטנדטים משתנים. שיטה התומכת בפרויקטים בגדלים שונים .	**Kanban**

תכן / עיצוב

שלב התכן/עיצוב :

הגדרת...

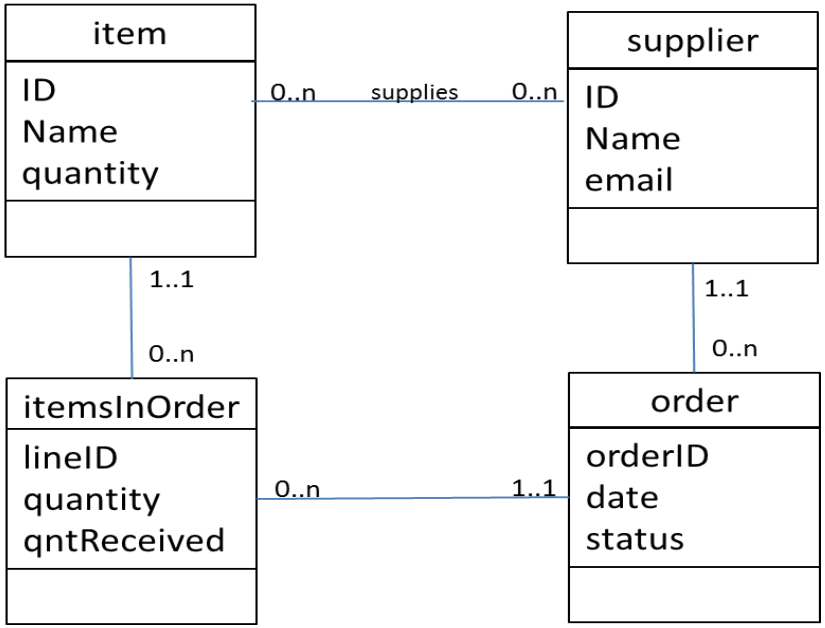
- הארכיטקטורה של המערכת.
- הטכנולוגיה למימוש - בהתאם לכך גם משתנה העיצוב (חלוקה לפרוצדורות ב C,למחלקות ב JAVAוב C++, או הגדרת ממשקים ב - COM
- השפה (פרוצדורלית/מוכוונת עצמים).

Unified Modeling Language (UML)

- שפה למידול ניתוח ועיצוב תוכנה.
- משפחה של סימונים גרפיים שמטרתם מידול של מערכות תוכנה בגישת Object Oriented, כל תרשים אספקט אחר (כמו תוכנית של בית שמורכבת מתרשים של הבית עצמו (חדרים, דלתות וכו'), אך גם תרשים נפרד של האינסטלציה, של החשמל, של החזיתות וכו'.)
- נוצרה ע"י ה- **Object Management Group** שהוא קונסורציום של **ארגונים שהתאחדו לצורך יצירת סטנדרטיזציה** בעולם המידול של Object Oriented.
- כל הדיאגרמות מפתוחות באופן **איטרטיבי ואינקרמנטלי**, ולעיתים קרובות במקביל זו לזו.
- כלי CASE מסייעים במידול.

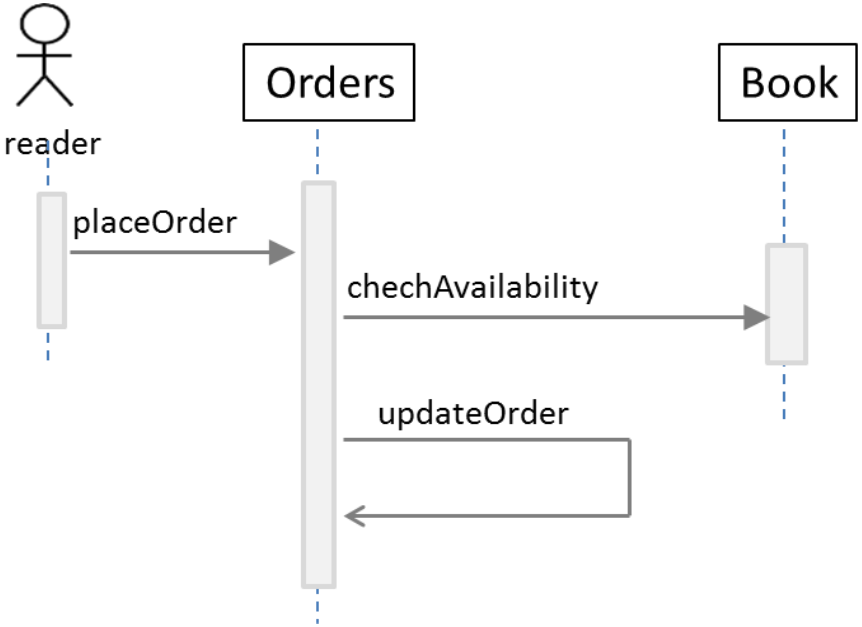
Class diagram (CD)

- כל class בדיאגרמה מורכב משלושה חלקים- name, properties, operation.
- Name- שם התכונה קצר ותמציתי. נכתב בגדול וב- bold.
- Properties– מאפיינים, אפשר לחשוב עליהם כעל שדות ב- class.
- Operations– פעולות שה- class מבצע ושמבצעים עליו.



Sequence Diagram

- האינטראקציה בין אובייקטים הנחוצה לצורך ביצוע תסריט אחד של UC או חלק ממנו
- Interaction diagrams
- מיוצר לאחר יצירת תרשים מחלקות CD
- תרשים דו מימדי בציר ה X נמצאים האובייקטים (classes ,actors) כשלכל אחד יש את קו החיים שלו (lifeline) בציר ה Y האנכי



Reuse in Agile

- מובנה
- ניתן להפרדה והגדרה
- מודולרי
- ניתן למכירה בחלקים
- תיעוד פשוט וברור
- חלקים עצמאיים
- קל לתיחזוק (הוספה, עדכון ושינויים)

בהצלחה!