

אם השפה אינה נכונה, אזי הדברים הנאמרים אינם משקפים את הכוונה;
אם מה שנאמר אינו משקף את הכוונה, אזי מה שיש לעשות אינו נעשה."
-קונפוציוס

הנדסת תוכנה

2. תהליכים



מה היום?

- מעט על עבודת צוות
- מהו **מחזור חיים** של תוכנה?
- מדוע יש צורך בתהליך פיתוח?
- מודלים של **תהליכי פיתוח** והערכתם
 - "קודד ותקן"
 - מפל המים
 - איטרטיבי וספיראלי
 - שיטות זריזות/גמישות (זמישות) – agile



- **תקנים**
- **מצגות הצעות פרויקטים (הרעיון, מימוש וישימות, שאלות)**
- **בתרגיל – עבודה עם [github/wiki](https://github.com)**
- **תרגיל אישי - html**
- **בפרויקט: יצירת קבוצות, אתחול הפרויקט והאתר שלו, סקר לשתי קבוצות (רישום)**

איפה אנחנו בפרויקט (בקורס)?

- למה?
בעיה (פלט: הצעת פרויקט\חזון\SOW)
- מי?
צוות (Inception, אתחול\תכנון פרויקט)
- מה?
דרישות (SRS)
- איך?
תיכון (ארכיטקטורה) (SDS)
- מתי?
תכנון וניהול – (ZFR)
- הלאה
(איטרציות, Code)

www.startwithwhy.com



עבודת צוות

**“Talent wins games, but
teamwork and intelligence wins
championships”**

מייקל ג'ורדן



"**People** are more important than **any process**.
Good people with a good process will
outperform good people with no process every
time."

- Grady Booch, Object Solutions: Managing the
Object-Oriented Project. Addison-Wesley, 1996

[@simonsinek](#): A team is not a group of people
that work together. A team is a group of people
that trust each other.

כמה קישורים

- [The Drucker Exercise](#)
- J. Spolsky, [How to be a program manager](#)
- Will Felps, “[How, when, and why bad apples spoil the barrel: Negative group members and dysfunctional groups.](#)”
[מה לעשות כשחברי הקבוצה אינם מסתדרים?](#)
- O'Brien, [People the missing ingredient](#) (talk)





תרגיל צוות

- הצלחה מגיעה בין השאר מחזון משותף
- תרגיל אישי, (2 דקות) חשוב ורשום:
 - מהם שלשה הדברים שאני הכי טוב בהם?
 - איך אני אוהב לעבוד?
 - מה הכי חשוב לי?
 - מה אפשר לצפות ממני? ("אני א...")
- מיני-תרגיל קבוצתי, חמש דקות דיון
 - כל אחד מספר חוויה טובה שהייתה לו בעבודה קבוצתית
 - דנים במחויבויות לצוות שלכם לאור התרגיל האישי
- פרסום בויקי הפרויקט את החזון המשותף לעבודת צוות

מבנה צוות

- צוותים דמוקרטיים מול ראש צוות
- צוותים מסביב לגורו
- צוותי Agile ו- Scrum (נראה)
- צוותי קוד פתוח
- מה המבנה שלכם?
- שימו לב ל [Conway's law](#)

כלים

"Instoolation (n) :
Belief that process
problems can be
solved by installing
a tool" - [G. Adzic](#)

- ללא כלים אלקט'! (באותו חדר, ללא)
- עבודה מרחוק: מייל, IRC, סקייפ, שיתוף שולחן עבודה,
- כלים ייעודיים, למשל: [tmux](#) (תכנות בזוגות מרחוק),
sococo (משרדים ווירטואליים), [join.me](#), [grou.ps](#),
סביבות עבודה בענן – למשל Cloud9, codenvy,
[תרשימים](#), משימות (למשל trello)



- כלי ניהול פרויקט (github)

2. תהליכי פיתוח תוכנה - מקורות

- Pressman, Chap. 2+3
- Laplante, Software Engineering
- Jalote, Chap. 2
- Wikipedia: Software Development Processes
<http://www.ambyssoft.com/essays/agileLifecycle.html#Iteration0>

עד עכשיו:

- הגדרת הנדסת תוכנה

– אוסף תהליכים, שיטות וכלים לפיתוח מוצר תוכנה בעל ערך ללקוח תוך שימוש מיטבי ואיכותי במשאבים, משלב הרעיון ועד לשלב הפרישה

- הבעיה: כיצד לפתח מוצר תוכנה ברמה תעשייתית

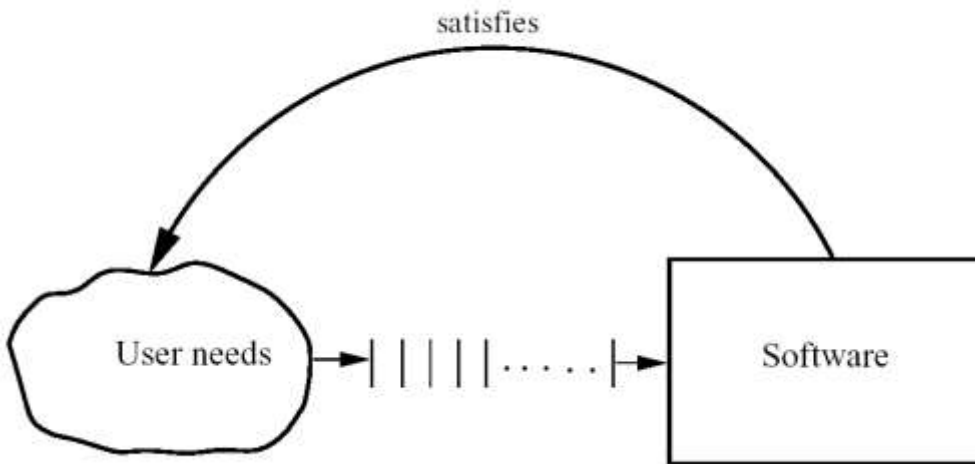
– מעבר למוצר יש לקחת בחשבון: עלויות, זמנים ואיכות

- מה עומד לרשותנו?

– אנשים

– טכנולוגיה

– תהליכים



ראשית: פיתוח תוכנה אד-הוק

- **פיתוח אד הוק:** יצירת תוכנה ללא כל תהליך רשמי או הוראות עבודה מסודרות
- חסרונות של פיתוח אד הוק:
 - ויתור על שלבים חשובים (תכנון, בדיקות)
 - סיכוי לפספוסים
 - לא ברור מתי לבצע משימות שונות
 - לא מתאים לקבוצות גדולות
 - קשה להעריך תוצרים
- אבחנה ידועה (AS/400,Boehm81IBM) – עלות תיקון בעיה עולה ככל שהיא מתגלה מאוחר יותר

מהו מחזור חיי תוכנה\תהליך פיתוח?

- הגדרה: אוסף צעדים \ שלבים, שבאמצעותם מוצר תוכנה מיוצר
 - משלב הרעיון עד ליציאה משימוש
 - יכול להימשך על פני חודשים או שנים
- מטרות לכל שלב
 - ציון ברור של הצעדים הדרושים לביצוע
 - תוצר ברור המאפשר סקירת העבודה
 - הגדרת פעולות לביצוע לשלב הבא
- לפעמים נקרא גם מתודולוגיה\מודל פיתוח\מחזור חיים ועוד

מהו בכלל מודל?

- 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, 2005]
- מודל משמש לתיאור מפושט של ישות מורכבת
 - מודל מתמקד בפרטים העקרוניים ללא ירידה לפרטים
 - מודל דורש "תרגום" לישות האמיתית
 - אמור לאפשר מימושים ופרשנויות שונות

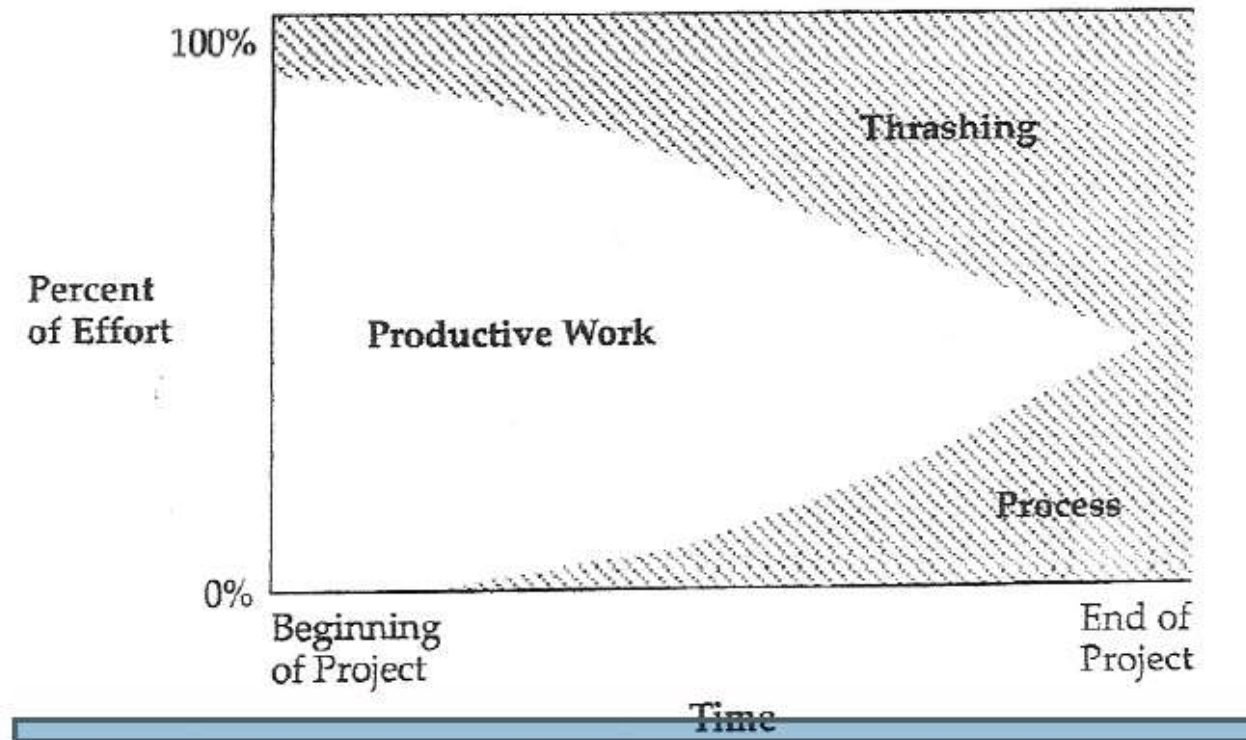
All models are wrong but some are useful
- George E. P. Box

מהם היתרונות לשימוש במחזור חיים?

- מסגרת לעבודה על המוצר
- מכריח אותנו לחשוב על "התמונה הגדולה" ומאפשר להגיע אליה צעד צעד
- אחרת, החלטות שיילקחו אולי נכונות מקומית ואישית, אך מפספסות את המטרה הכללית
- כלי ניהולי, הבטחת איכות

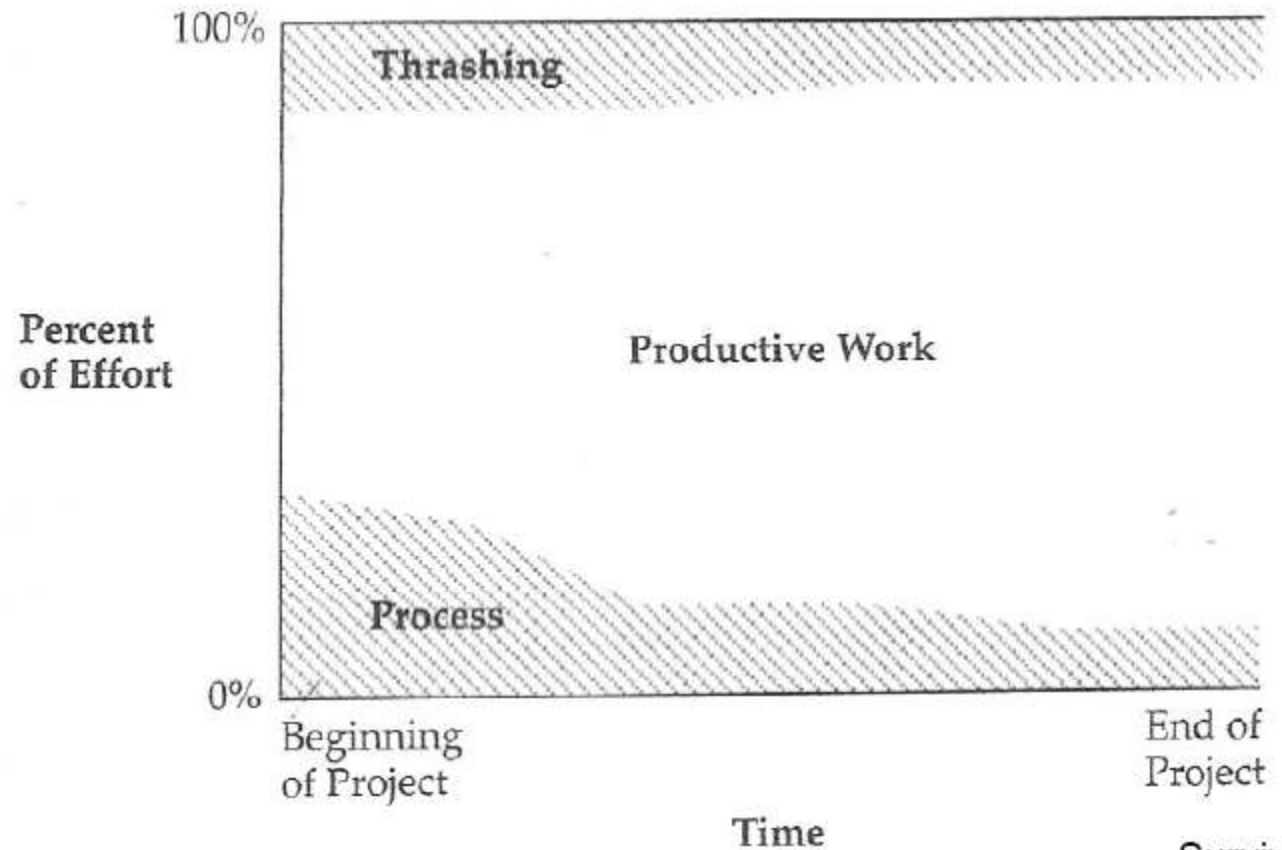
חסרונות?

פרויקט עם מעט תהליכים



McConnell, Project Survival
Guide, p. 24

פרויקט עם תהליכים



McConnell, Project Survival
Guide, p. 25

מחסנית הנדסת תוכנה



פעילויות לדוגמא

- דרישות
 - איסוף, ניתוח (Analysis)
 - תיכון (Design)
 - מימוש
 - בדיקות\אימות\שילוב
 - תיעוד
 - הטמעה\תמיכה\אחזקה
- פעילויות תומכות
 - ניהול הפיתוח
 - הבטחת איכות
 - סקרים
 - ניהול תצורה
 - כלים

לכל פעילות בד"כ יש תוצרים

איך בונים בית?

דרישות בעלי עניין נוספים:

- דרישות רישוי
- דרישות תשתיות (ממשקים)

צרכי הלקוח:

- מגורים ושירותים
- חזות
- אופציות עתידיות
- אילוצי תקציב ולו"ז



מפרטי דרישות וארכיטקטורה:

- תכנית קירות, רצפות, גגות
- תכנית חזיתות
- תכנית נקודות חשמל ומים
- תכנית פתחים, מדרגות



תכנון ומפרטים טכניים:

- תכנית קונסטרוקציה
- תכנית אינסטלציה (חשמל, מים, ביוב...)



בניה:

הקמת שלד
התקנת חשמל, צנרת, ...
גימור



תכנון / פירוט

Open Main Window

תוכנה בונים כמו בית...

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

הקמה / בדיקה

צרכי הלקוח:
- פונקציונליות
- 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



תכנון / פירוט

בניה:
קידוד
הידור (קומפילציה)
קישור
שילוב



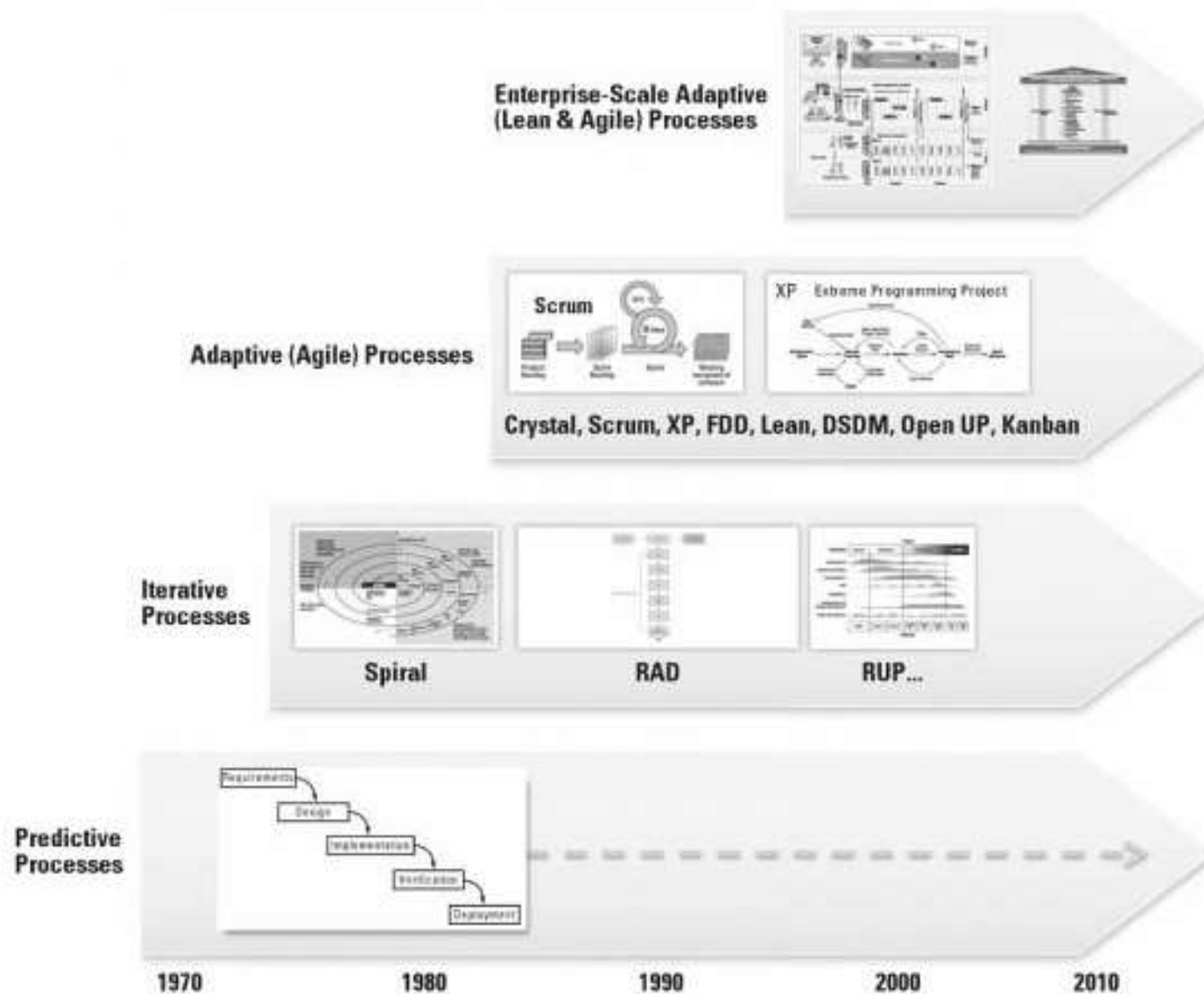
האם?

מהו ההבדל העיקרי בין הנדסת בניין לתוכנה?

1. לתוכנה אין ארכיטקטורה
2. עלות שינויים מאוחרים
3. פיתוח תוכנה לא מצריך שלבים כ"כ שונים ומופרדים
4. תשובות 2 ו-3 נכונות

קיצור תולדות תהליכי פיתוח תוכנה

- דור 1: ללא תהליך – קודד ותקן
- דור 2: מתוכנן – מפל המים
- דור 3: איטרטיבי/אינקרמנטלי – ספיראלי, RAD, RUP
- דור 4: מסתגל – Agile, Scrum/Lean
- דור 5: ?



מודל "קודד ותקן" (~אד-הוק)



"קודד ותקן" – בעד ונגד

- יתרונות

- אין (כמעט) תקורה של התהליך, "מסתערים" על הפרויקט

- מתאים לפרויקטים קטנים וממוקדים

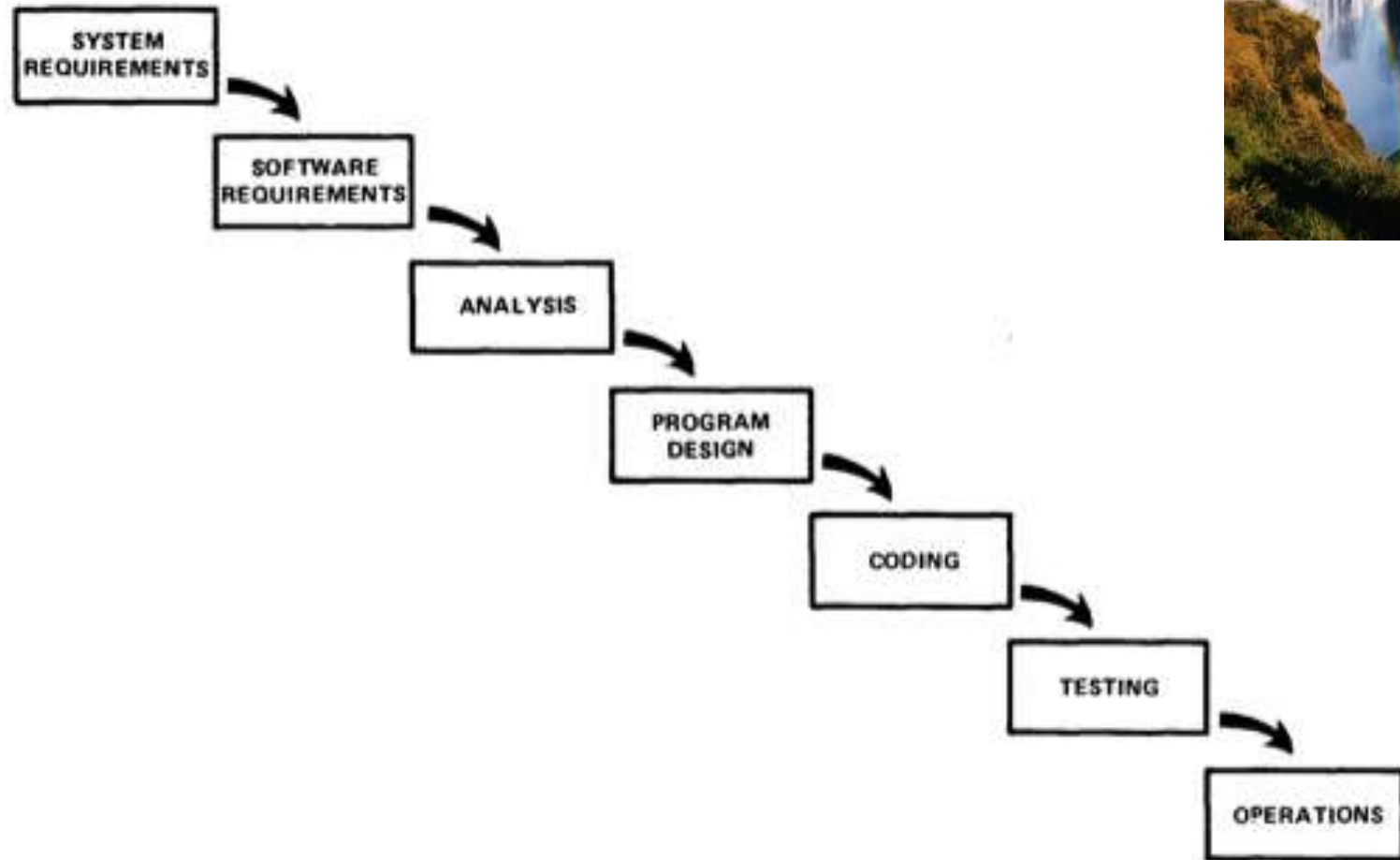
- חסרונות

- אין דרך **למדוד** התקדמות, איכות או סיכונים

- שינויים מאוחרים עלולים להצריך שינויים משמעותיים בתכנ

- לא ברור מתי מסיימים, תכולות, תמיכה וכו'

דור ו - מודל "מפל המים"



מפל המים

- מיוחס ל-Royce [1970] (שבעצם התכוון הפוך... *)
- לא מתקדמים לשלב הבא לפני סיום השלב הנוכחי
- השם משמש בעיקר לביקורת על המודל
- ...בשימוש נרחב ואפילו מחויב לפי חלק מהתקנים

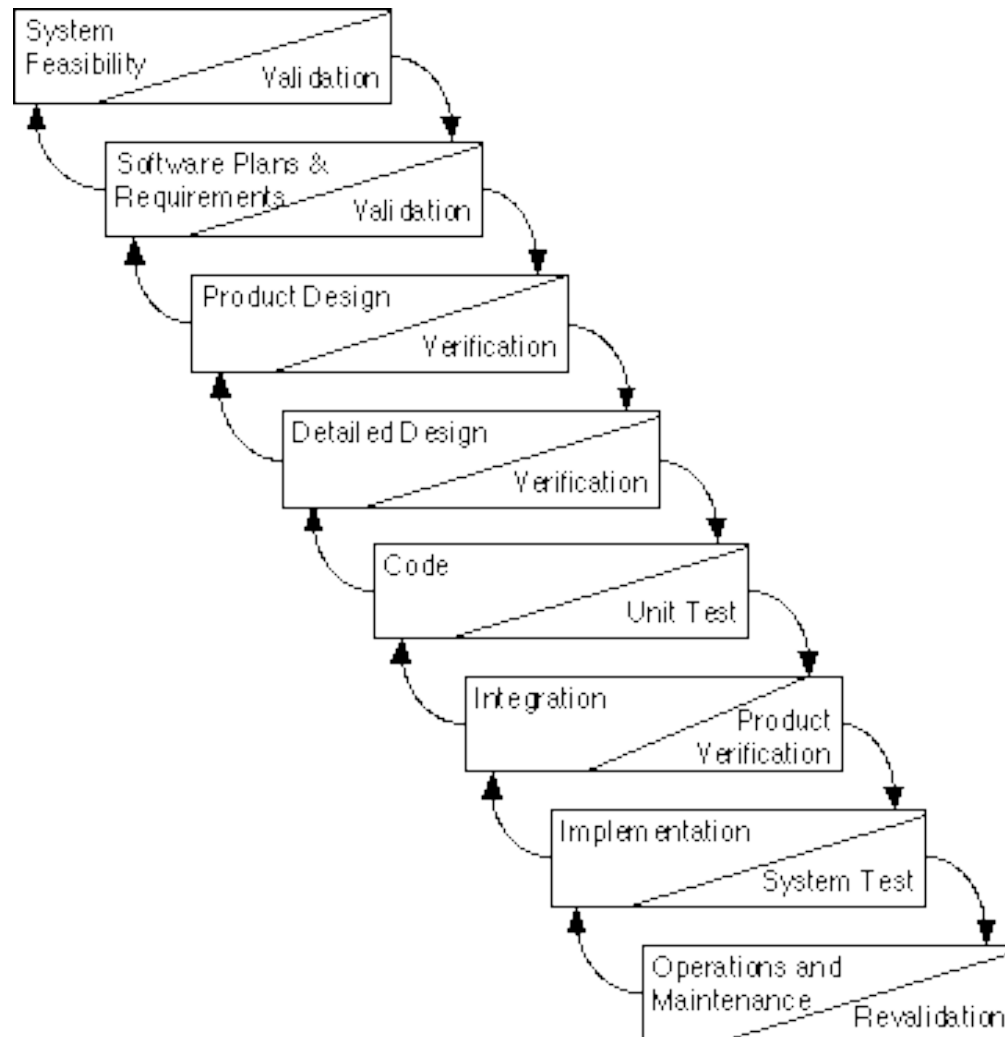
מפל המים - יתרונות

- מתאים לפרוייקטים צפויים ומובנים אך מצד שני מורכבים (למשל מערכות מידע)
 - מאפשר לתכנן (כמעט) הכל מראש
 - התהליך האידיאלי, אם אין שינויים באמצע
- מתאים לצוות לא מנוסה
 - מודל סדרתי שקל לעקוב אחריו
 - בכל שלב אפשר לבחון אם להתקדם הלאה
- מתאים למודל הכלכלי של הרבה לקוחות (כיום)

מפל המים - חסרונות

- בד"כ אי-אפשר לחשוב על הכל מראש
- אין הרגשת התקדמות לפני הסוף
- אינטגרציה מתבצעת רק בסוף
- נוגד את מה שכבר מוסכם שצריך לשלב מוקדם ובקביעות
- המוצר יוצא קשיח בהעדר משוב במהלך הדרך
- ייתכן שבסוף התהליך יצא מוצר שהלקוח אינו מעוניין בו
- עלול לגרום לחוסר נצילות בצוות
- חוסר גמישות\פתיחות לשינויים

מפל המים – עם משוב (הרחבה V)



דור III – איטרטיבי \ אינקרמנטלי

- נציגים מרכזיים:

- ספיראלי

- RAD –

- RUP –

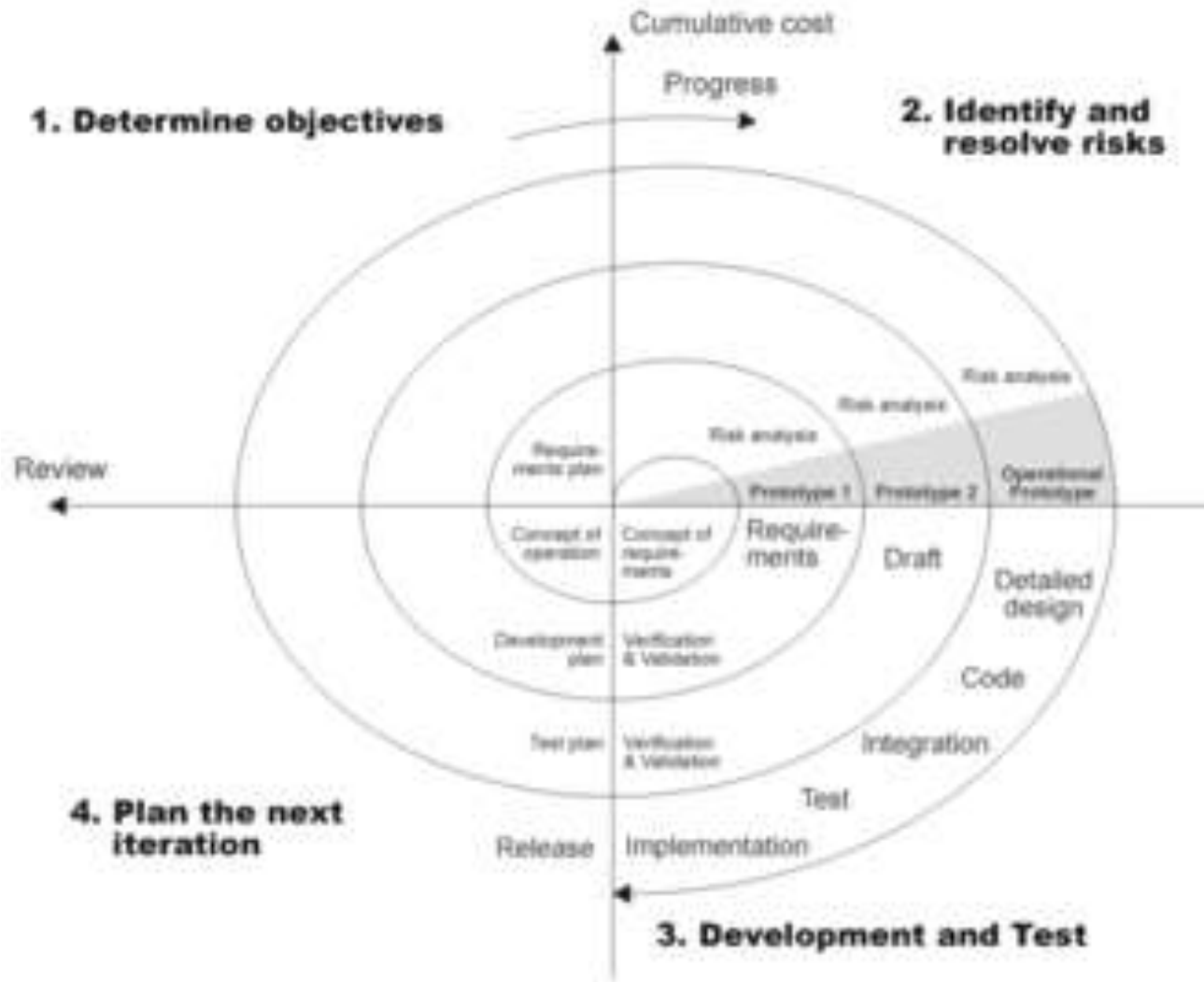
מודלים איטרטיביים\אינקרמנטליים

- **סבבים** (קצרים בד"כ) של הוספת פונקציונליות
 - אפשר לתכנן ולממש חלקים במקביל
- כל סבב אפשר לחלק **לאיטרציות**
 - לפעמים רק מסגרת זמן (שבוע\חודש)
- יתרונות
 - כל התקדמות קלה להבנה ולניהול
 - מימוש ראשוני של פונקציונלית עיקרית ואח"כ לפי משלב מהלקוח
 - הורדת סיכונים

המודל הספיראלי

- הוצע ע"י Boehm ב-1988
- הוסיף למפל המים את המימד האיטרטיבי\אבולוציוני
- מוכוון הורדת סיכונים בשלבים, החל מהגדולים ופעולות בהתאם
 - האם אנו בונים את המוצר הנכון
 - האם יש לקוחות למוצר
 - האם ניתן לממש את המוצר בעזרת הטכנולוגיה הקיימת היום? מחר?
- התקדמות זהירה יותר אל התוצאה דרך **אבות טיפוס**
 - בכל שלב המטרות מתבררות יותר
- כבר פחות מקובל

המודל הספיראלי – מוכוון סיכונים

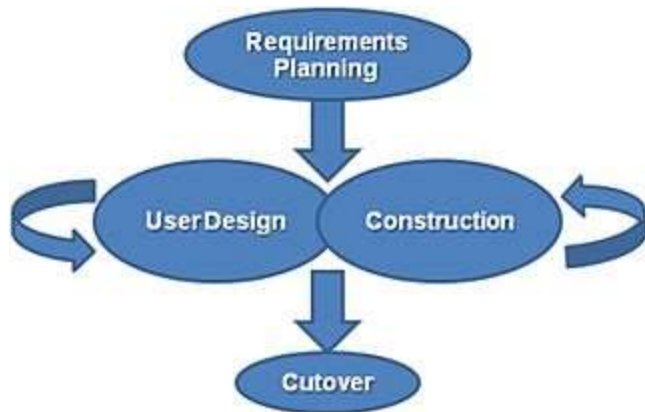


המודל הספיראלי - יתרונות

- מתאים גם לפרויקט גדול, אבל כזה שבו הדרישות בתחילת הדרך לא לגמרי ברורות
- מספק משוב וזיהוי מוקדם של בעיות, מאפשר התאמה לשינויים
- בהשקעה מתאימה, מורידה סיכונים – כדאי להתחיל ולהתמקד בסיכון הגדול ביותר
- חסרונות?
 - "זה עובד, בא נמכור אותו", הרבה ניהול ותכנון, מצריך מומחיות בניהול סיכונים

Rapid Application Development

- תכנון מועט בכדי להגיע לאב-טיפוס מהיר
- אוסף שיטות המסתמכות גם על כלי פיתוח

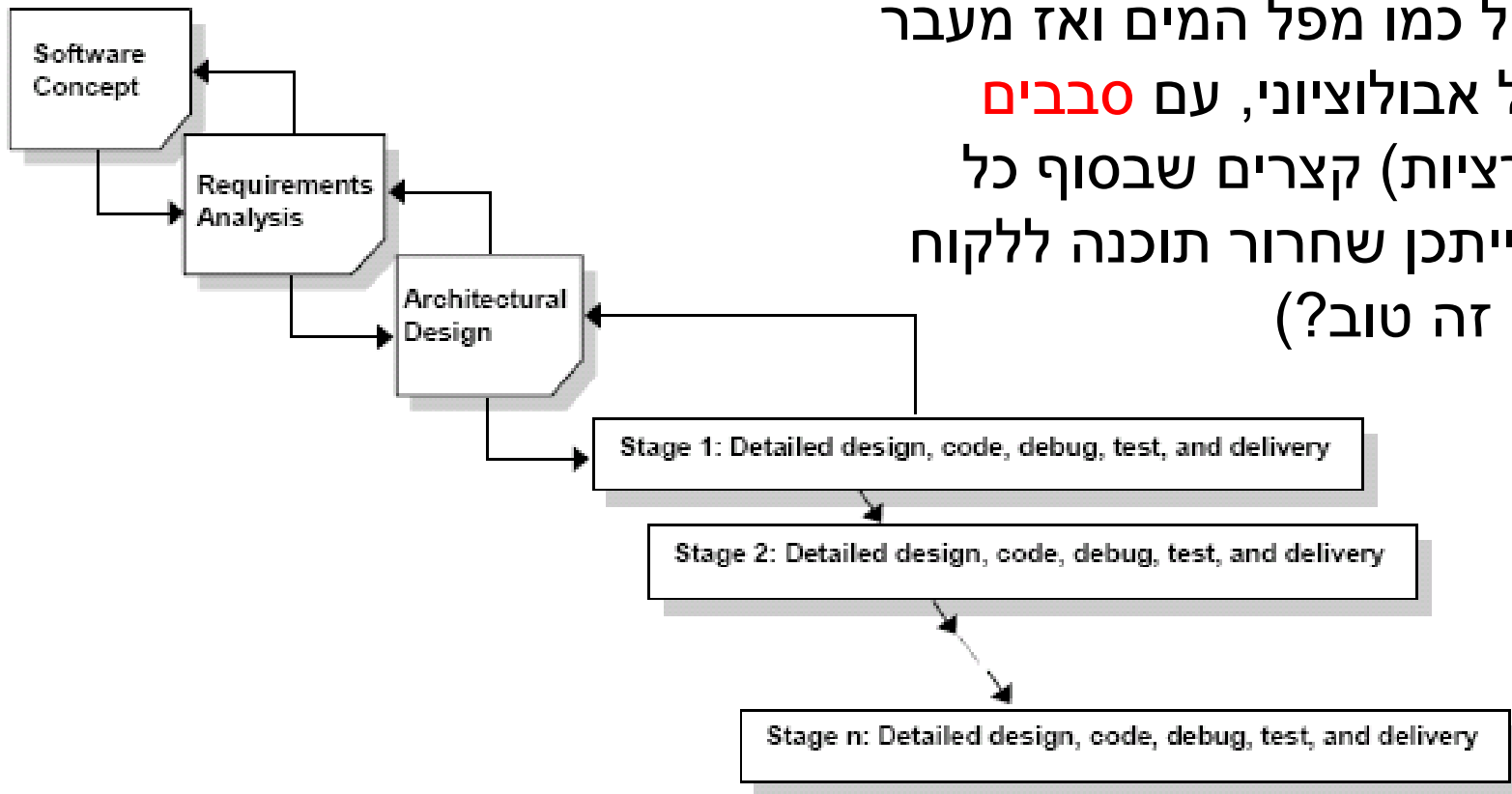


שחרור בשלבים

- McConnell
- חלוקה למודלים ע"י ארכיטקט, לכל מודל מחזור פיתוח מלא וקצר
- יתרונות
 - של מפל המים (זיהוי פונקציונלית עיקרית, הורדת סיכונים על ידי זיהוי מוקדם של בעיות)
 - עדיין גמיש לשינויים
 - בעיות אינטגרציה מזהות מוקדם
- חסרונות
 - עלות שחרור\אינטגרציה
 - האם תמיד ניתן לחלק?
- היה בשימוש די נרחב

מודל "שחרור בשלבים"

Staged/Rapid Delivery



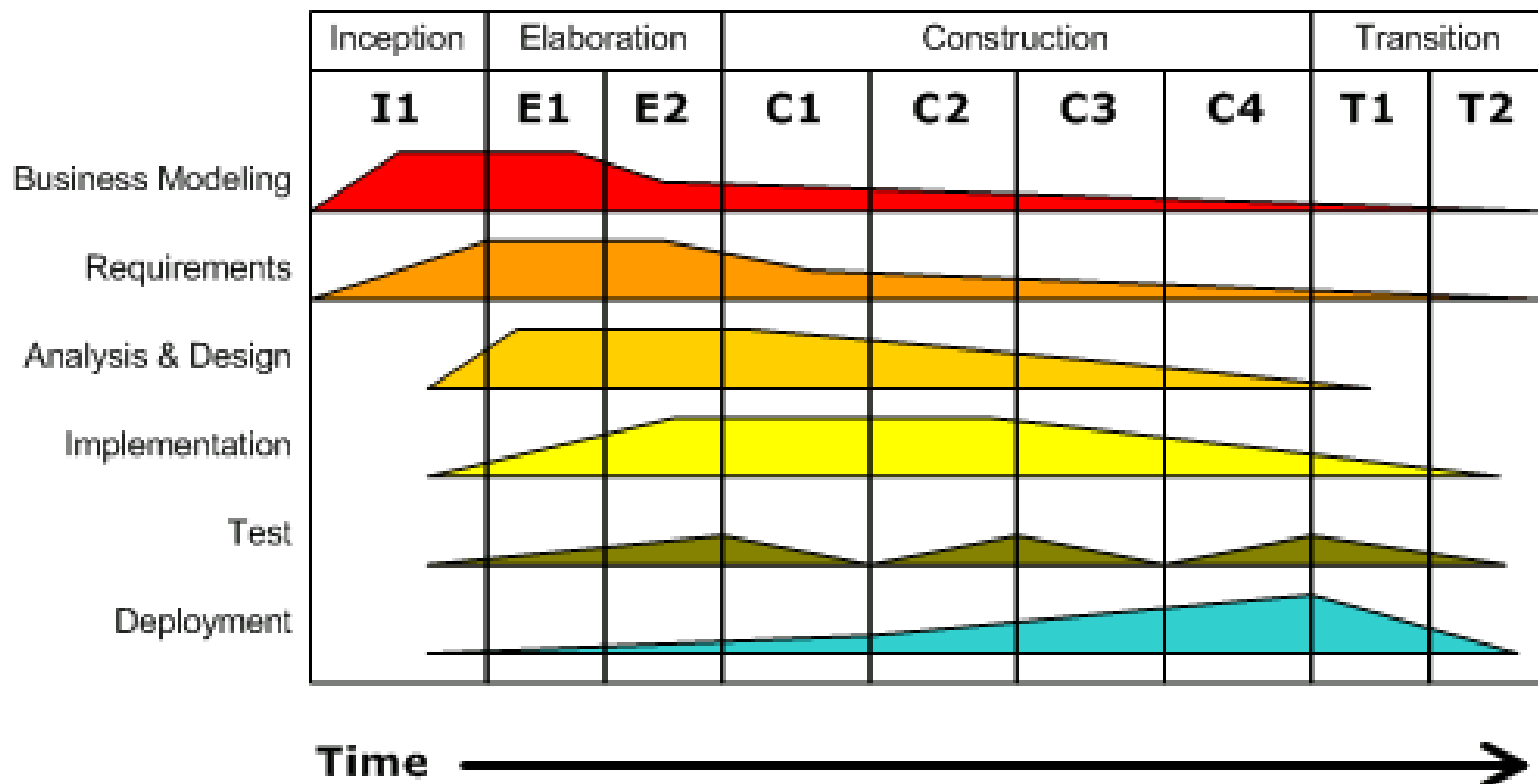
Unified Process

- התפתח מתוך המודל הספיראלי
- Rumbauch, Booch, Jacobson 1999
- מעטפת תהליכים לפיתוח מונחה עצמים בעזרת UML (בהמשך)
- IBM/Rational UP
- מתאים לפרויקטים ענקיים
- התפתח ל- OpenUP, Agile RUP

Unified Process

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



Unified Process

Inception phase

Vision document
Initial use-case model
Initial project glossary
Initial business case
Initial risk assessment.
Project plan,
phases and iterations.
Business model,
if necessary.
One or more prototypes

Elaboration phase

Use-case model
Supplementary requirements
including non-functional
Analysis model
Software architecture
Description.
Executable architectural
prototype.
Preliminary design model
Revised risk list
Project plan including
iteration plan
adapted workflows
milestones
technical work products
Preliminary user manual

Construction phase

Design model
Software components
Integrated software
increment
Test plan and procedure
Test cases
Support documentation
user manuals
installation manuals
description of current
increment

Transition phase

Delivered software increment
Beta test reports
General user feedback

דור 4 - מודלים זריזים Agile

- Beck, 2001 ועוד 16 מובילים מפרסמים "[מנשר](#)" ([עברית](#), [ויקי](#)):

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the **right**, we value the items on the **left** more.

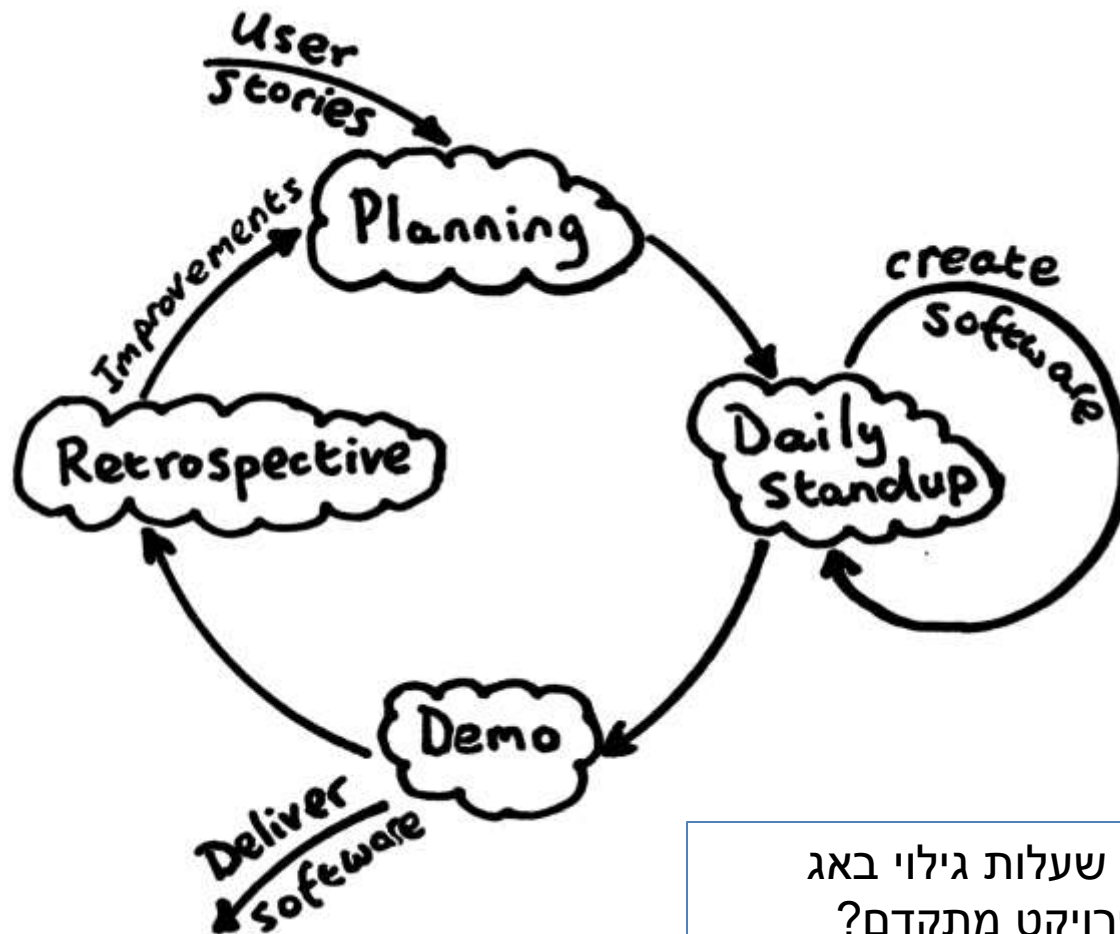
- כוללים עקרונות מפורטים יותר
- קצת אחרת: [Manifesto for Software Craftsmanship](#) – [וביקורת נגדית](#)

קיצור ערכי Agile

- **People + Feedback** = Decent Chance of Success [Cockburn]
- Embrace **Change**
- Small **Steps**



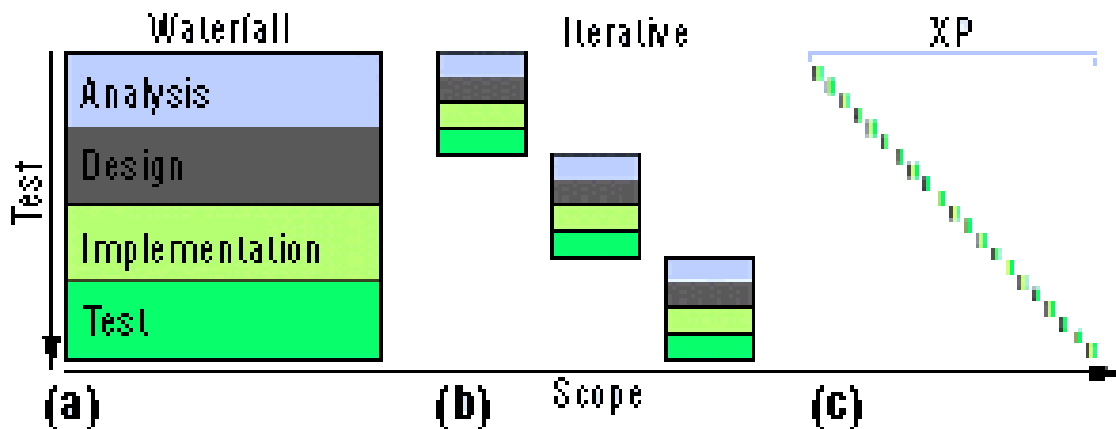
מודל למחזור החיים העיקרי ב-Agile



מה עם הטענה שעלות גילוי באג
עולה ככל שהפרויקט מתקדם?

XP – Extreme Programming

- Kent Beck 1999 (תוך כדי הומצא הויקי ע"י Cunningham! האם במקרה?)
- איסוף "best practices" למתודולוגיה אחת "טוטליטארית" (ומטלטלת)
- כוללת ערכי יסוד, עקרונות ומיומנויות

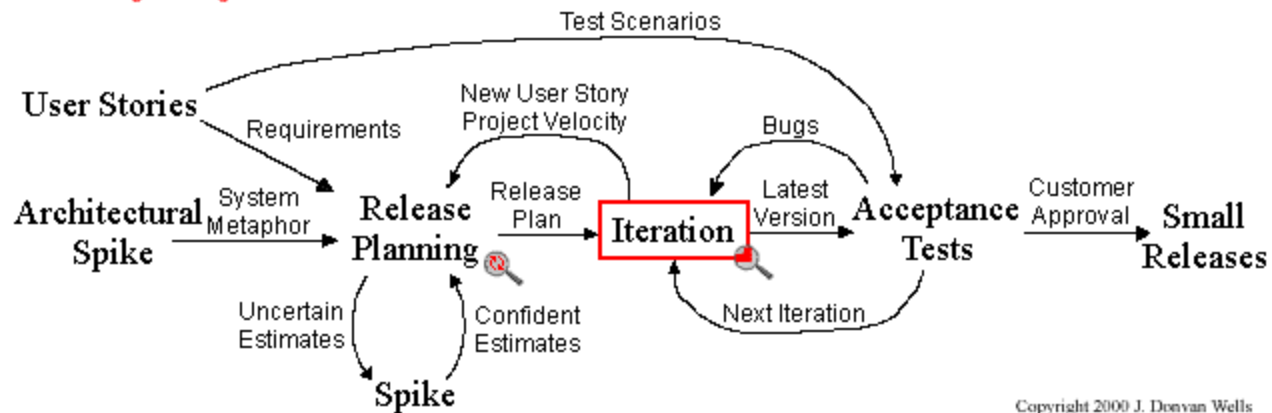


Kent Beck, "Embracing Change with eXtreme Programming"

XP Project



Extreme Programming Project



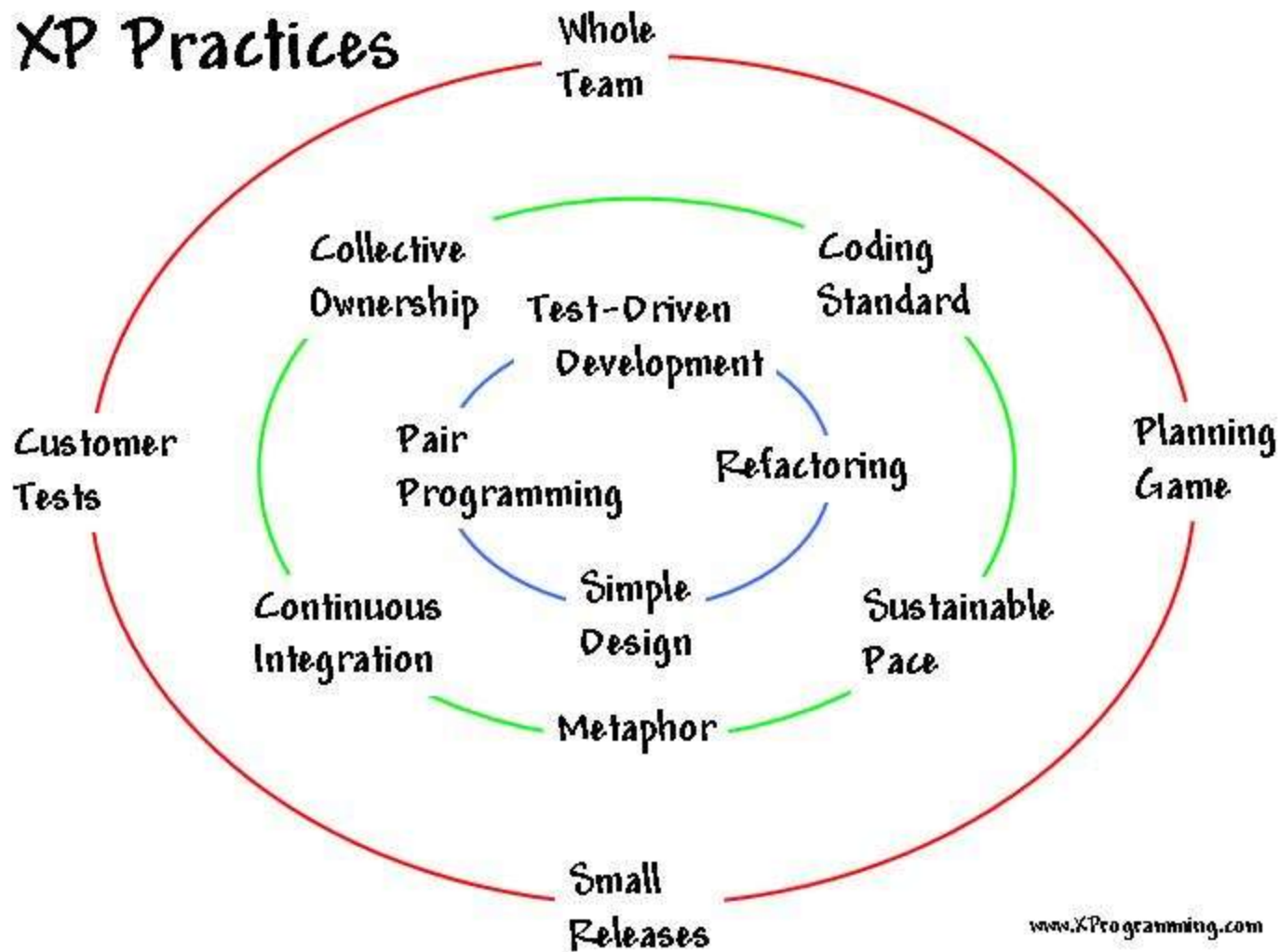
Copyright 2000 J. Donovan Wells

שתיים עשרה המיומנויות של XP

- "משחק" התכנון
- שחרור גרסאות קטנות
- מטאפורה
- תכן פשוט
- בדיקות רצופות
- עיצוב מחדש
- תכנות בזוגות
- בעלות משותפת
- שילובים רצופים
- שבוע עבודה בן 40 שעות
- הלקוח זמין באתר הפיתוח
- כללי קידוד

כיצד באים לידי ביטוי ערכי Agile?

XP Practices

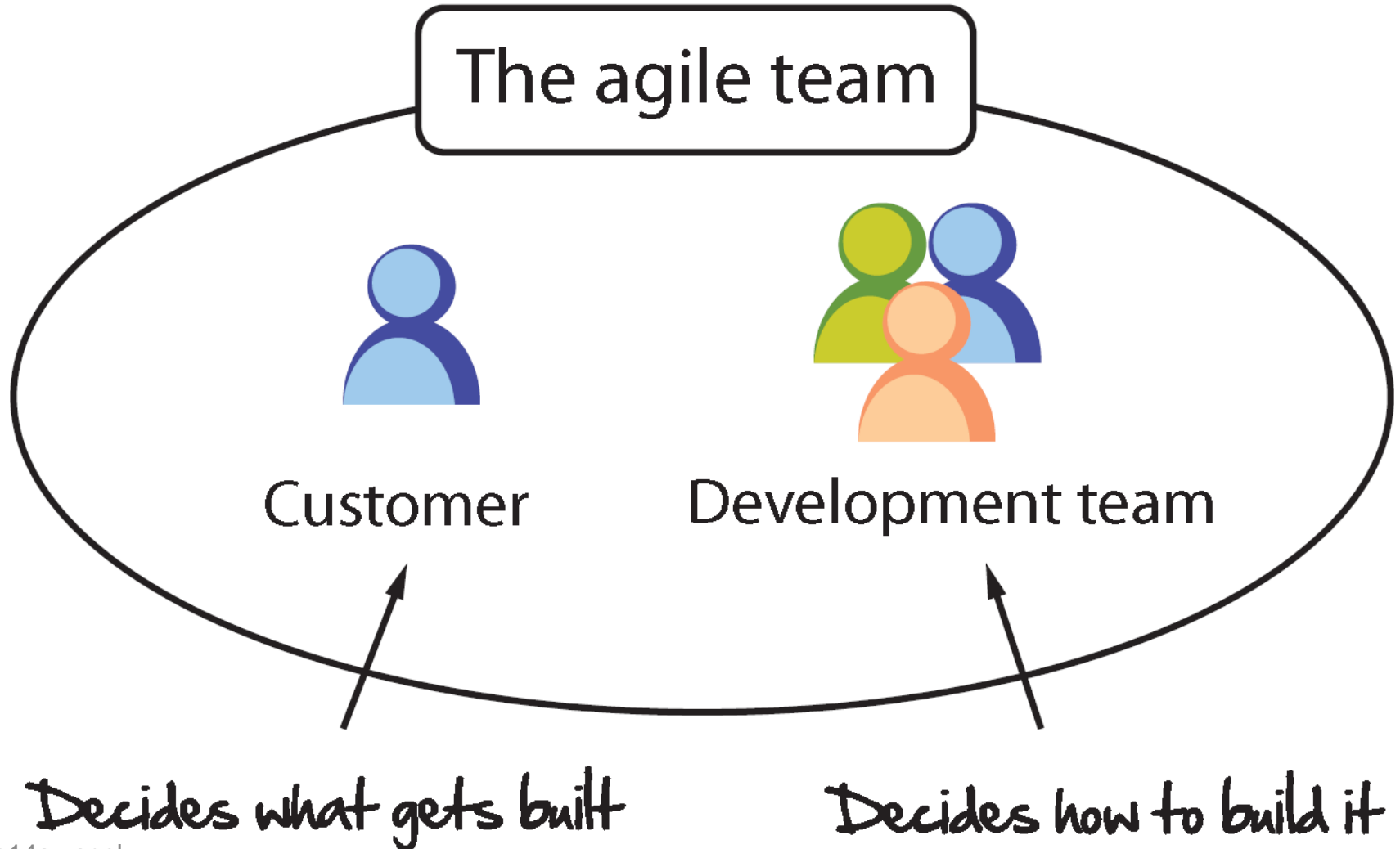


Other Agile

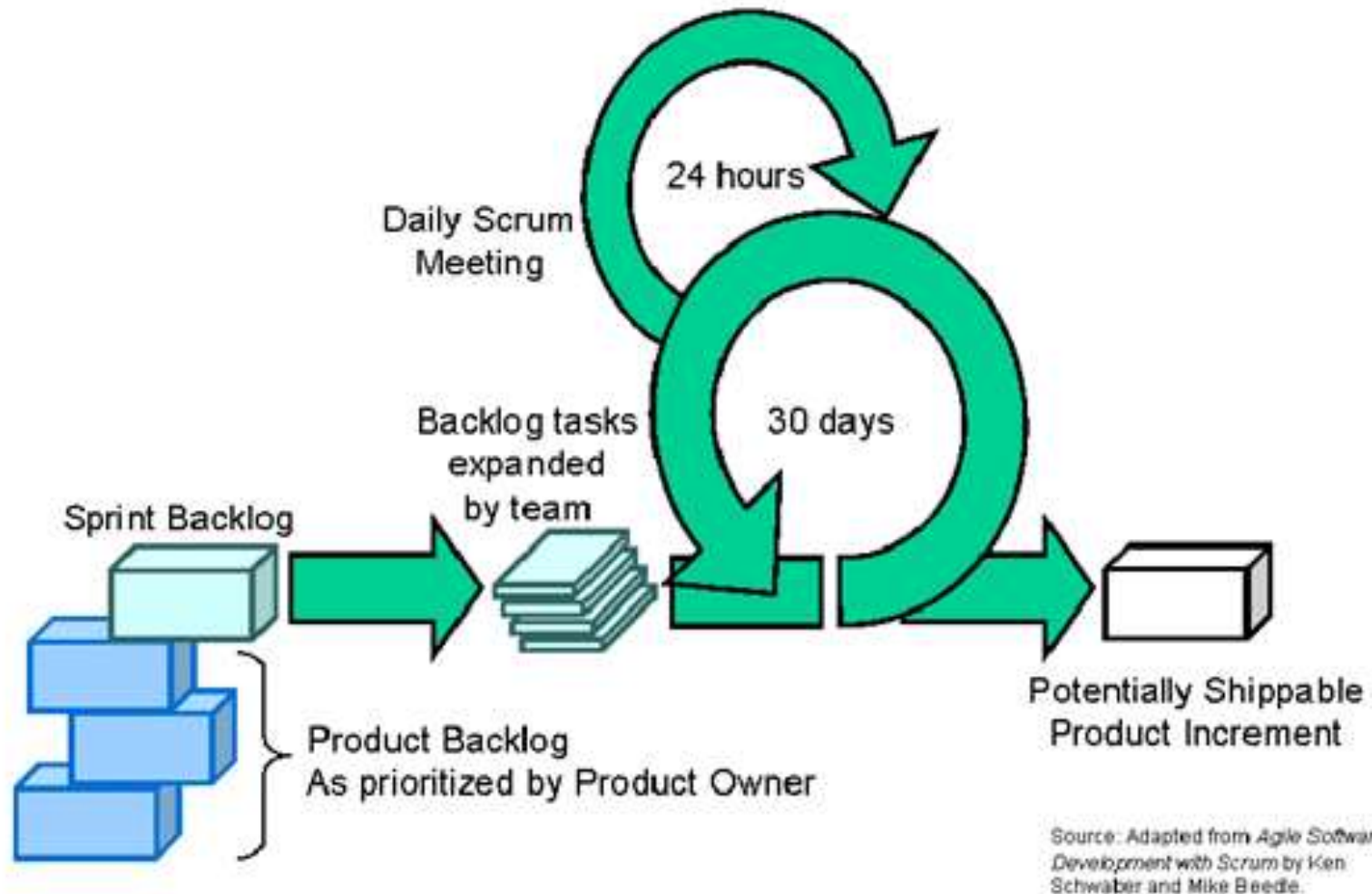
• SCRUM

- מבוסס על Takeuchi&Nonaka'86
- גישה הוליסטית לעבודת צוות לפיתוח מוצרים חדשים
- בשימוש נרחב מאד (גוגל, מיקרוסופט, ...)
- [מחקר פורסטר](#) (2010) רוב החברות כבר עברו
- עוד: Lean Startup, MSF, DSDM, Crystal Clear
שילוב? שיטה משלכם?
- העיקרון המוביל: התאמה למשוב בזמן-אמת
- נחזור ונבקר בהמשך הקורס, עקרונות שונים
- [מצגת בעברית](#)

תפקידים ב-Agile



Scrum Lifecycle

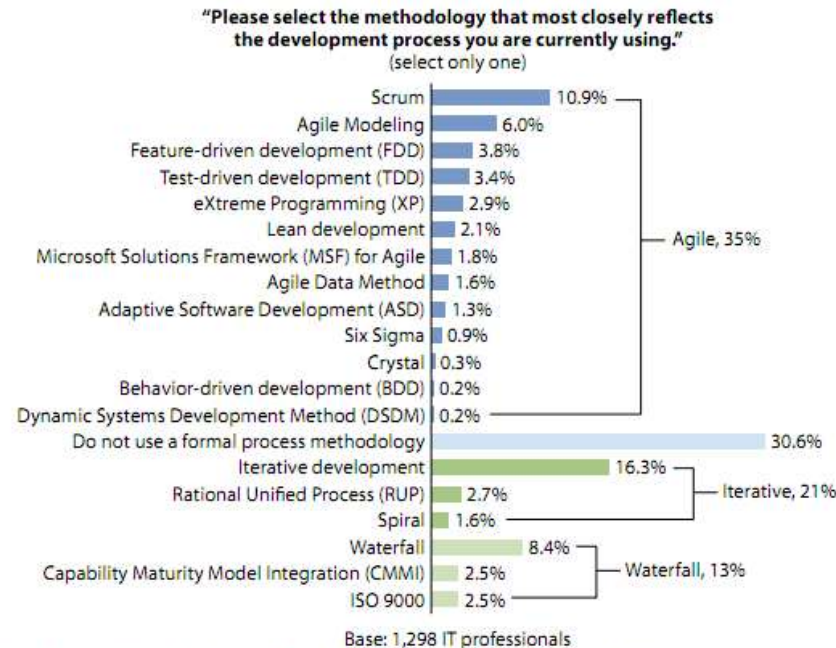


Lean / Kanban

- הפעם אנחנו לומדים מעקרונות ייצור בתעשייה
- http://en.wikipedia.org/wiki/Lean_software_development
- [http://en.wikipedia.org/wiki/Kanban_\(development\)](http://en.wikipedia.org/wiki/Kanban_(development))

Agile Adoption: Forester 2009

Figure 1 Agile Adoption Has Reached Mainstream Proportions



Source: Forrester/Dr. Dobb's Global Developer Technographics® Survey, Q3 2009

56100

Source: Forrester Research, Inc.

- 2011:
http://www.versionone.com/state_of_agile_development_survey/11/

מתי להשתמש ב-Agile

- פרויקט דינמי עם קבוצה קטנה ובעלת מוטיבציה
- דרישות משתנות

- יש לקוח מעוניין וזמין

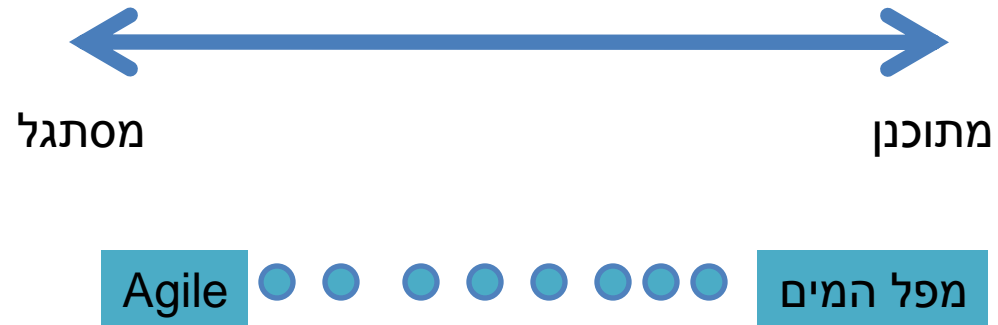


Copyright © 2003 United Feature Syndicate, Inc.

חסרונות

- לא מתאים לכל המקרים והארגונים (מי הזיז את הגבינה שלי?)
- מה קורה עם מאמצים רק חלק מההרגלים?
- שינויים מאוחרים עדיין יכולים לעלות הרבה

זריזים מול אחרים



מדוע יש כ"כ הרבה מודלים

- בחירה זה טוב, אבל...

- מה הכי מתאים?

- It Depends, אף פרויקט אינו דומה לאחר

- תפירת מודל אישיו\קבוצת\חברה\פרויקט

- שינוי – הובלה\עלויות\...

- האם באמת משתמשים במודלים?

Beck: “put in what
you need when
you need it”

Clements '86]

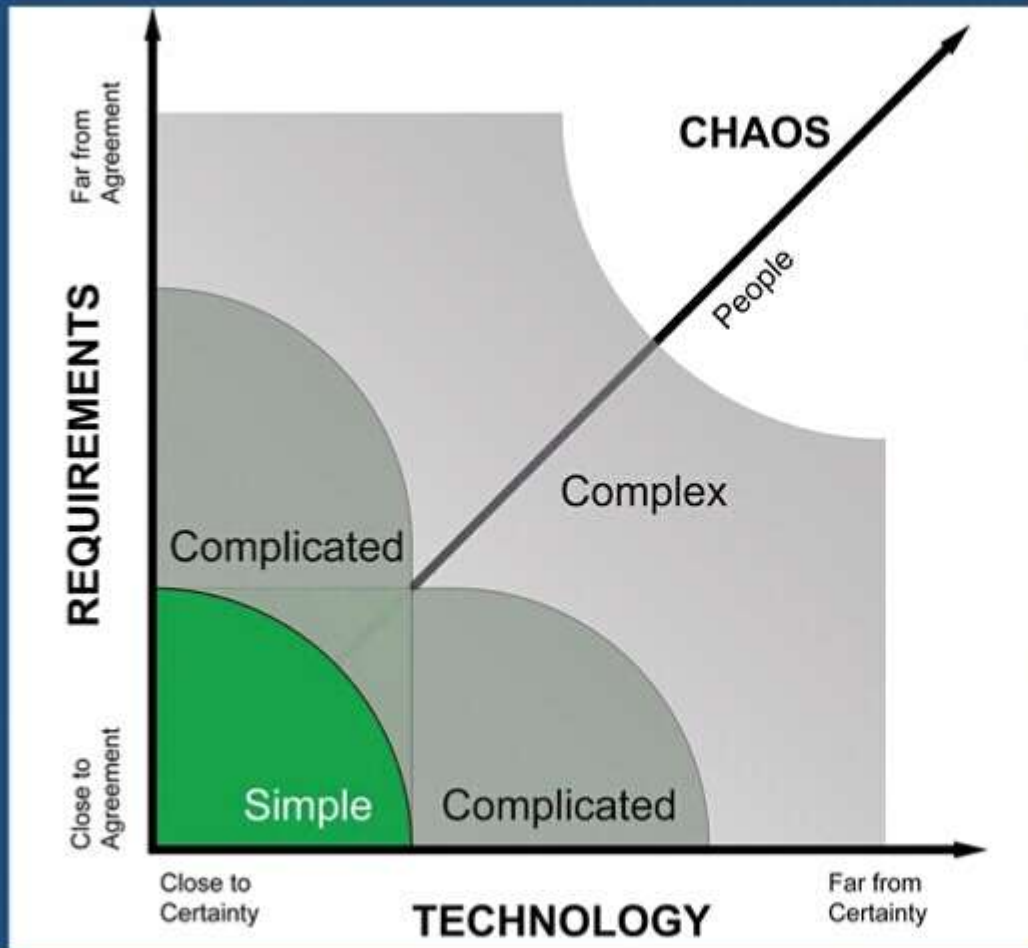
A rational design process: how and why to fake it

- Brooks, “No Silver Bullet”.

Fowler, “Cannot Measure Productivity”

- “We must destroy these methodologies that get in the way of...Programming, XXX.”

- מה מתאים לפרויקט הקבוצתי?



Source: Ralph Stacey, University of Hertfordshire

Simple

Everything is known

Complicated

More is known than unknown

Complex

More is unknown than known

Chaotic

Very little is know

“The Inevitable Pain of Software Development” [Berry]

- “Each method, if followed religiously, **works**. Each method provides the programmer a way to manage complexity and change so as to delay and moderate the B-L upswing. However, each method has a **catch**, a fatal flaw, at least one step that is a real pain to do, that people put off. People put off this painful step in their haste to get the software done and shipped out or to do more interesting things, like write more new code. Consequently, the software tends to decay no matter what. The B-L upswing is inevitable.”

Bugs Found Per Release

se14a-yagel

Release Number

תקנים והסמכות

- תקני US DoD
 - MIL-STD-498 , DOD-STD-2167A
 - נוהל מפת"ח
- תקני הנדסה ודרישות איכות
 - IEEE/EIA 12207
 - ISO 9000-3
- מודל בשלות CMMI (CMU-SEI)
- האם בשימוש? שילוב Agile

פיתוח בשיטת מפל המים הכי מתאים ל:

1. אתר אינטרנט למשרד כרטיסים
2. מערכת הפעלה לטלפון חדש (Scrum@Nokia)
3. רכיב דפדפן עבור צ'אט מרובה משתמשים
4. סטרטאפ שמחפש לבנות מוצר חדשני

בפעם הבאה

- דרישות

- סדנה

- אתחול פרויקט קבוצתי

- משימת קריאה 2:

~~Assaf Stove, [Why Should I INVEST in User Stories?](#)~~

- שאלה: מה הקשר בין דרישות לבדיקות? תן דוגמה מהפרויקט שלך.

לסיכום

- האם תהליך יכול לעזור לפיתוח תוכנה?
([Cargo Cult SE](#))
- מהו המודל המתאים ביותר?
("No One Size Fits All", "No Silver Bullet")
- ישנם מודלים נוספים (למשל של קוד פתוח,
דור5?)
- <http://theleanstartup.com/> Build-Measure-Learn –
בפרויקט?
- מהו השלב הבא?

Constantine: If you don't know what you're going to do before you do it, you don't know what you're doing.

If you spend all your time figuring out what you're doing, you're doing nothing.

משחק המרשמלו

- <http://marshmallowchallenge.com>
- סרטון תוצאות [TED](#) – מי הכי מצליח?