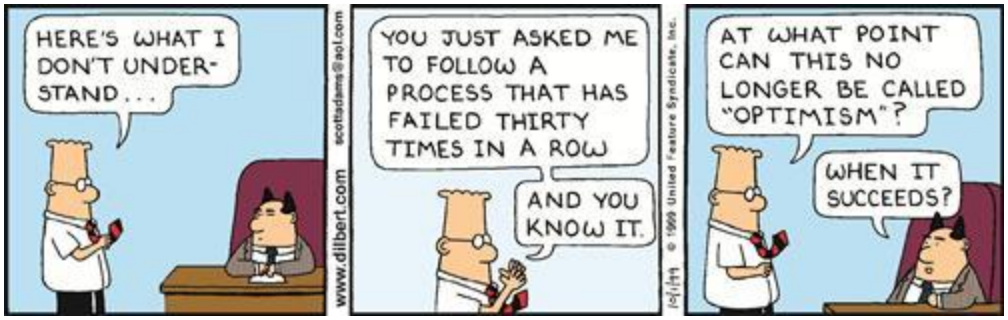
"אם ה**שפה** אינה נכונה, אזי הדברים הנאמרים אינם משקפים את ה**כוונה**;
אם מה שנאמר אינו משקף את הכוונה, אזי מה שיש לעשות **אינו נעשה**."
-קונפוציוס

# הנדסת תוכנה

## 2. צוות (ותהליכים)



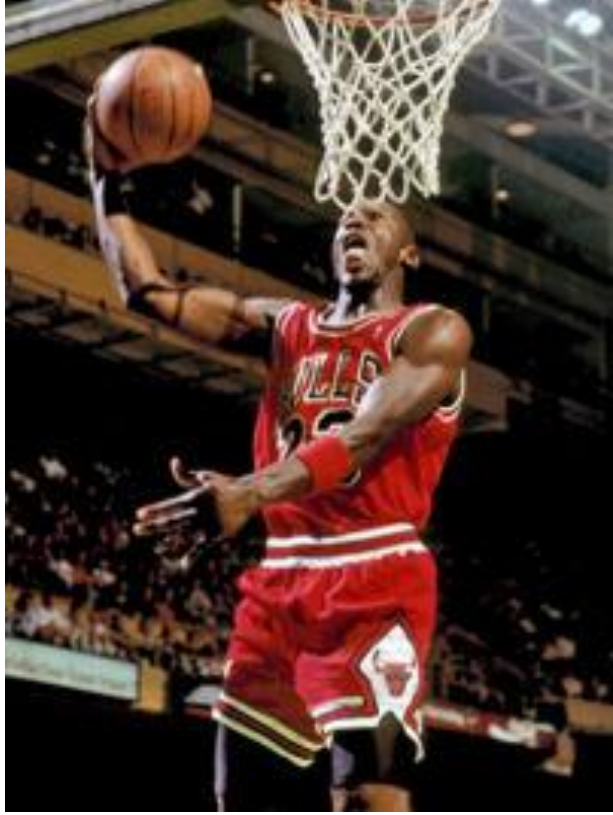

# מה השבוע?

- 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)
- (הרצאת טכנולוגיה למיזמים חברתיים)
- משימות
  - היום: מצגות הצעות פרויקטים (הרעיון, מימוש וישימות, שאלות)
  - אישי– אפליקציית ווב: צד לקוח
  - בפרויקט: יצירת קבוצות, אתחול הפרויקט והאתר שלו, סקר ציבורי 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.

# התפוח הרקוב ועוד

- W. Felps, “How, when, and why bad apples spoil the barrel: Negative group members and dysfunctional groups.”
- K. Matsudaira, The Paradox of Autonomy and Recognition - Thoughts on trust and merit in software team culture
- The Drucker Exercise
- J. Spolsky, How to be a program manager
- מה לעשות כשחברי הקבוצה אינם מסתדרים?
- O'Brien, People the missing ingredient (talk)

# "Overcoming The Five Dysfunctions of a Team", P. Lencioni

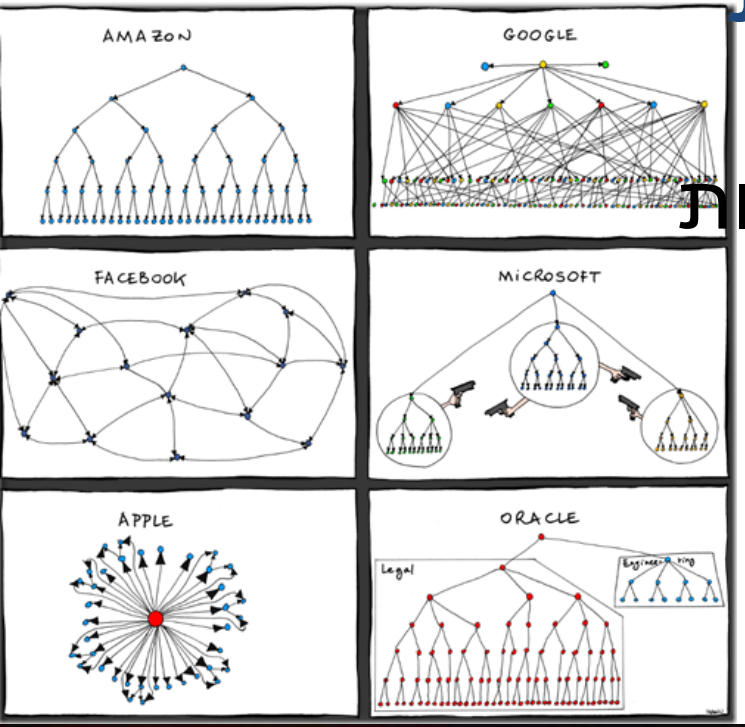
# Diversity שונות

- "Why Some Teams Are Smarter Than Others? It’s about listening, empathy and having more women“
  - http://www.nytimes.com/2015/01/18/opinion/sunday/why-some-teams-are-smarter-than-others.html?_r=0


# תרגיל צוות

- הצלחה מגיעה בין השאר מחזון משותף
- תרגיל אישי,( 2 דקות) לחשוב ולרשום:
  - 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)
- <u>צוותי קוד פתוח</u>


- מה המבנה שלכם?
  - שימו לב ל Conway's law

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

# כלים

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

- כלי ניהול פרויקט (github)
- דיון ורשימה בפרק רברסים


Sometimes Social Media looks like this
problogger.net

# פתרון קונפליקטים

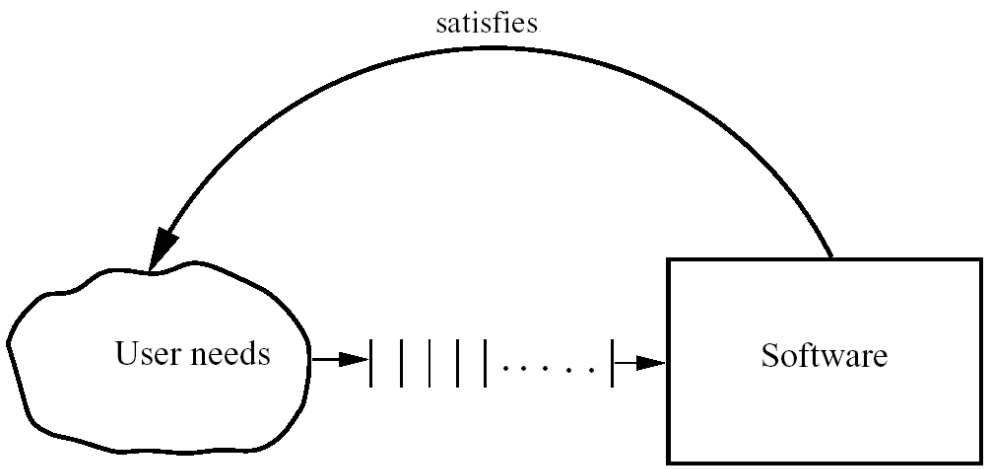
- למשל באיזה כיוון טכני לבחור
- רשימה של הדברים שמסכימים עליהם
  - אולי אנחנו בעצם מסכימים
- כל אחד חוזר על הטיעונים של השני
  - אולי בכלל הוויכוח רק על מינוח
- Constructive confrontation (Intel)
  - לא לפחד להעלות בעיות
- Disagree and commit (Intel)
  - אם החלטנו כולם מחויבים
- שימושי לחיים בכלל

# 2. תהליכי פיתוח תוכנה - מקורות

- Pressman, Chap. 2+3
- Meyer, Agile! : The Good, the Hype and the Ugly
- Laplante, Software Engineering
- Jalote, Chap. 2
- Wikipedia: Software Development Processes http://www.ambysoft.com/essays/agileLifecycle.html#Iteration0
- שובל, תכנון ניתוח ועיצוב מערכות מידע, (פרק בנושא)

# 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?

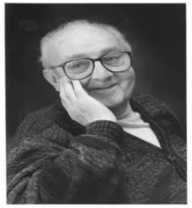
- הגדרה: אוסף צעדים \ שלבים, שבאמצעותם מוצר תוכנה מיוצר
  - משלב הרעיון עד ליציאה משימוש
  - יכול להימשך על פני חודשים או שנים
- מטרות לכל שלב
  - ציון ברור של הצעדים הדרושים לביצוע
  - תוצר ברור המאפשר סקירת העבודה
  - הגדרת פעולות לביצוע לשלב הבא
- לפעמים נקרא גם מתודולוגיה\מודל פיתוח\מחזור חיים ועוד

# 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

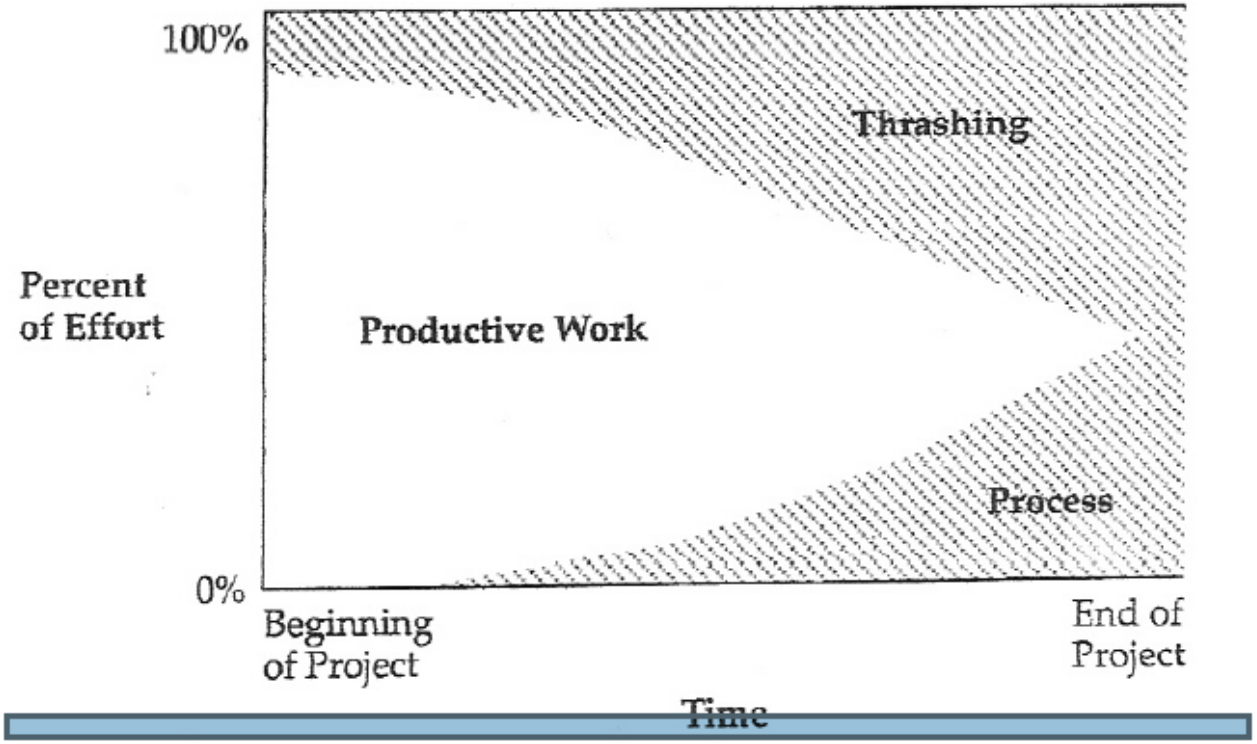


# מהם היתרונות לשימוש במחזור חיים?

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

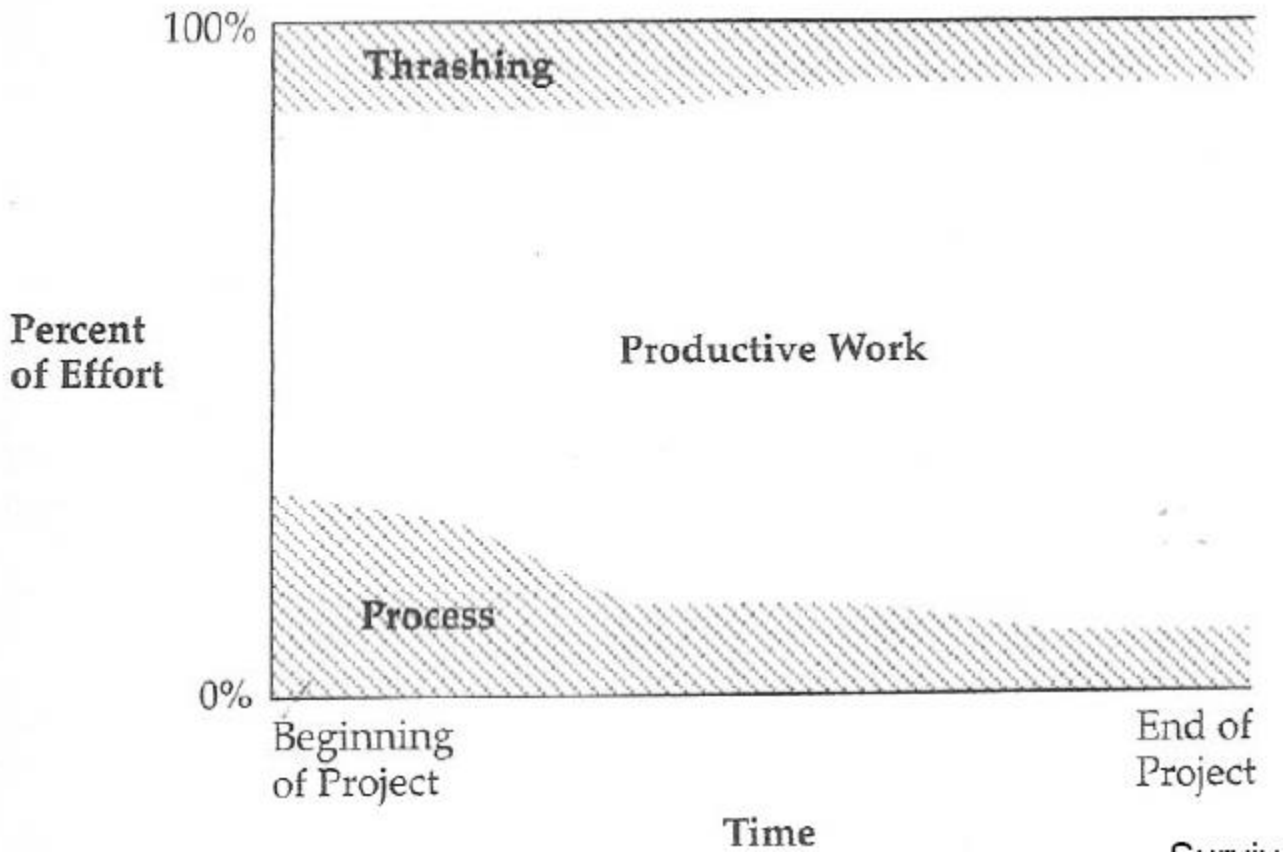
חסרונות?

# 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

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

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

הקמה / בדיקה

תכנון / פירוט

Open Main Window

# תוכנה בונים כמו בית...

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

**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

**בניה:**
קידוד
הידור (קומפילציה)
קישור
שילוב

הקמה / בדיקה

תכנון / פירוט

**האם?**

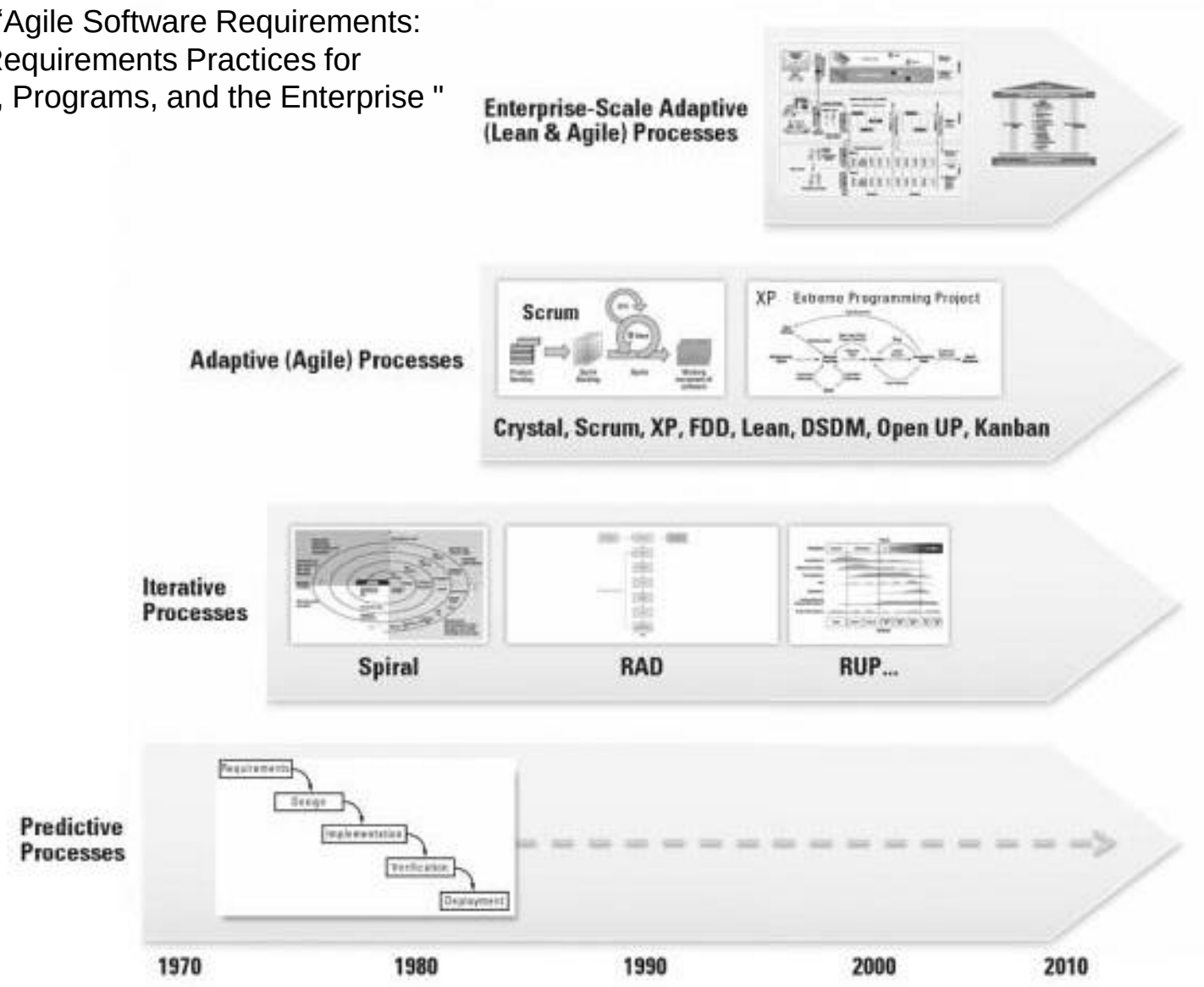
# מה יכול להיות ההבדל העיקרי בין הנדסת בניין לתוכנה?

1. לתוכנה אין ארכיטקטורה
2. עלות שינויים מאוחרים
3. פיתוח תוכנה אינו מצריך שלבים שונים ומופרדים
4. אין הבדל ממשי

# 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: ?

From: “Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise "

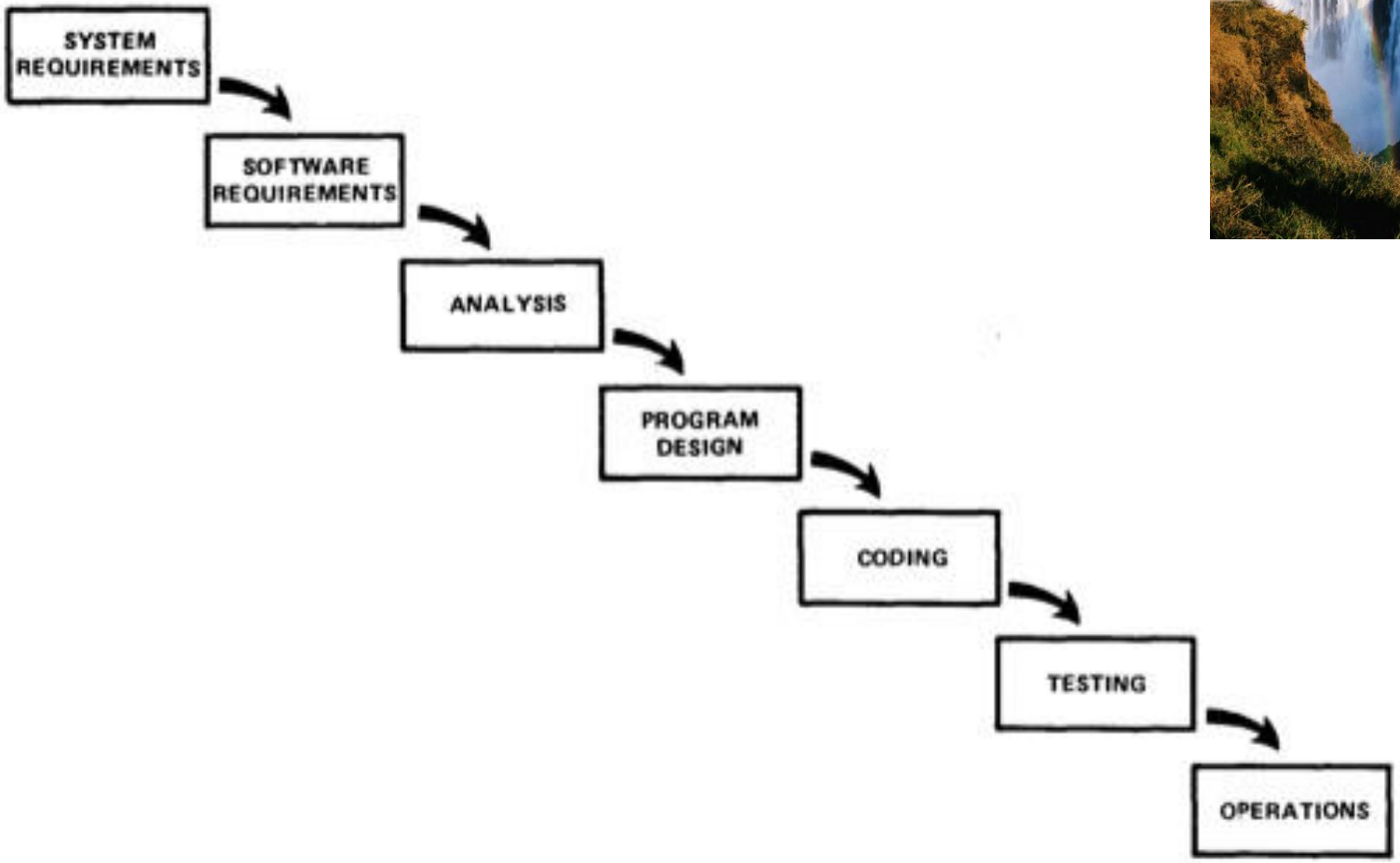


# מודל "קודד ותקן" (~אד-הוק)

# "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 בתכן
  - לא ברור מתי מסיימים, תכולות, תמיכה וכו'

# דור II - 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

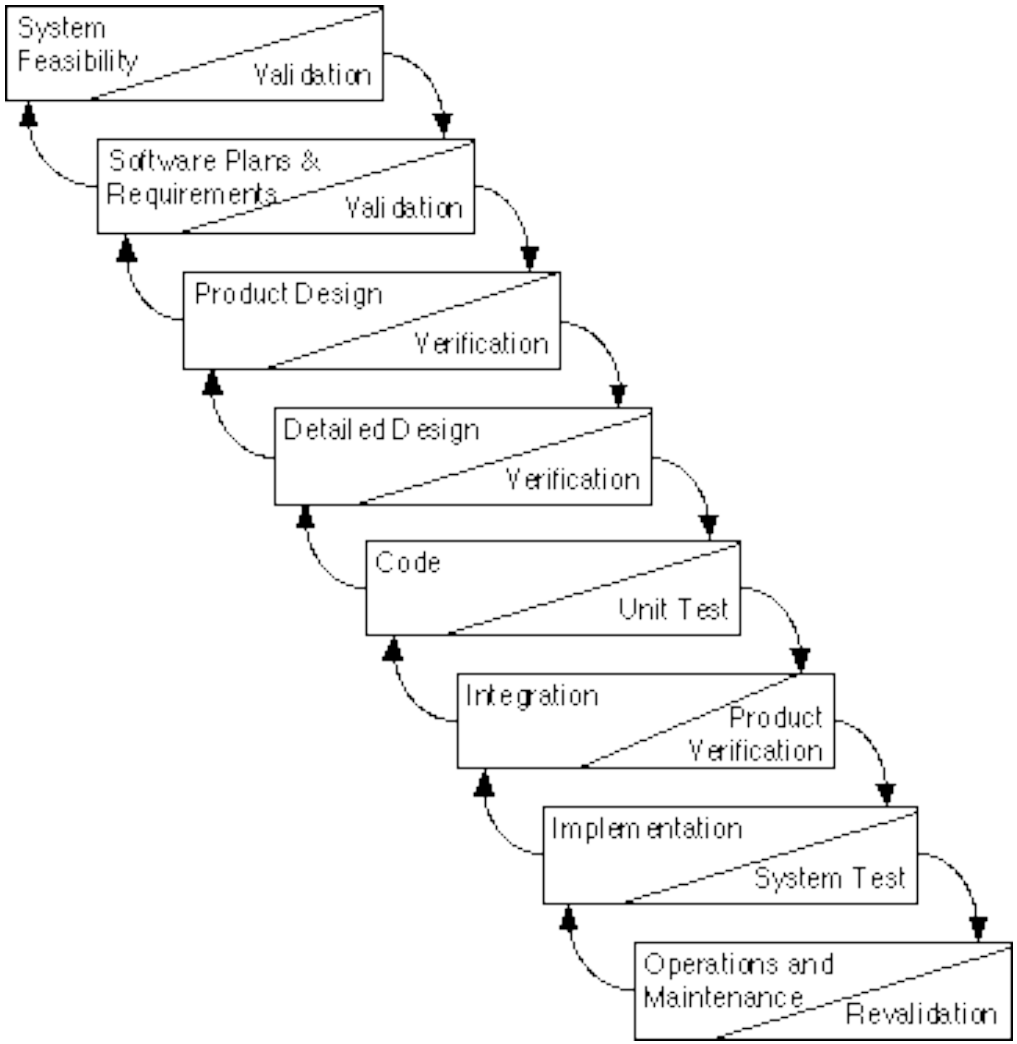
# מפל המים - יתרונות

- 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

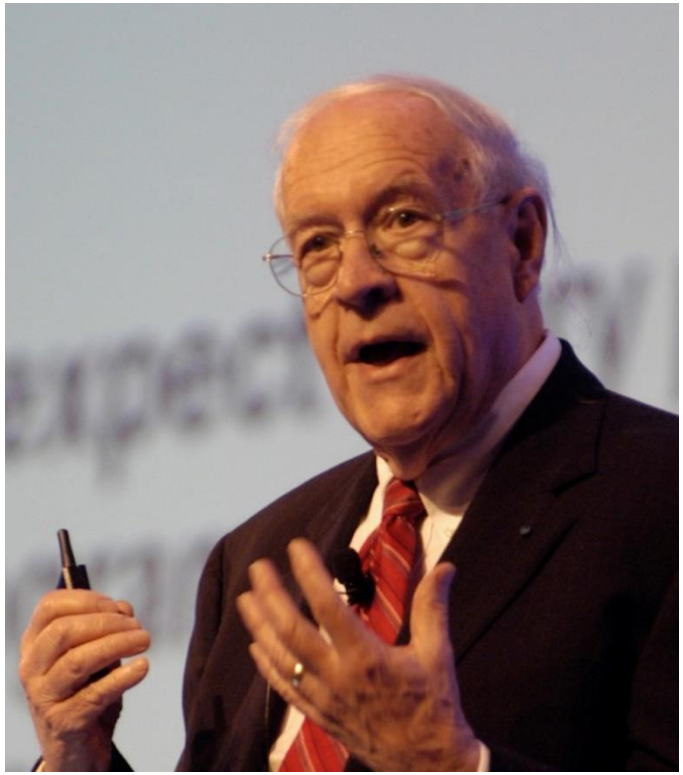
- בד"כ אי-אפשר לחשוב על הכל מראש
- אין הרגשת התקדמות לפני הסוף
- אינטגרציה מתבצעת רק בסוף
  - נוגד את מה שכבר מוסכם שצריך לשלב מוקדם ובקביעות
  - המוצר יוצא קשיח בהעדר משוב במהלך הדרך
  - ייתכן שבסוף התהליך יצא מוצר שהלקוח אינו מעונין בו
- עלול לגרום לחוסר נצילות בצוות
- חוסר גמישות\פתיחות לשינויים

# מפל המים – עם משוב (הרחבה V)



# How well does Waterfall work?

- *"Plan to throw one [implementation] away; you will, anyhow*."
  - Fred Brooks, Jr.
    (1999 Turing Award winner)
- Often after build first one, developers learn right way they should have built it



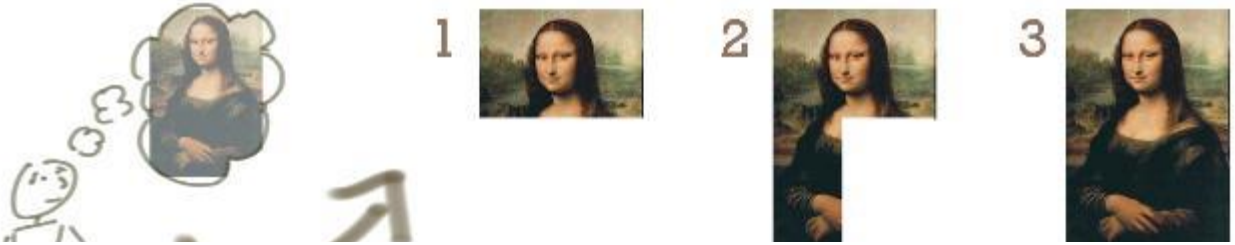
(Photo by Carola Lauber of SD&M www.sdm.de. Used by permission under CC-BY-SA-3.0.)

# דור III – איטרטיבי \ אינקרמנטלי

- ספיראלי
- אב טיפוס
- RAD
- שחרור בשלבים
- RUP

# מודלים איטרטיבים\אינקרמנטליים

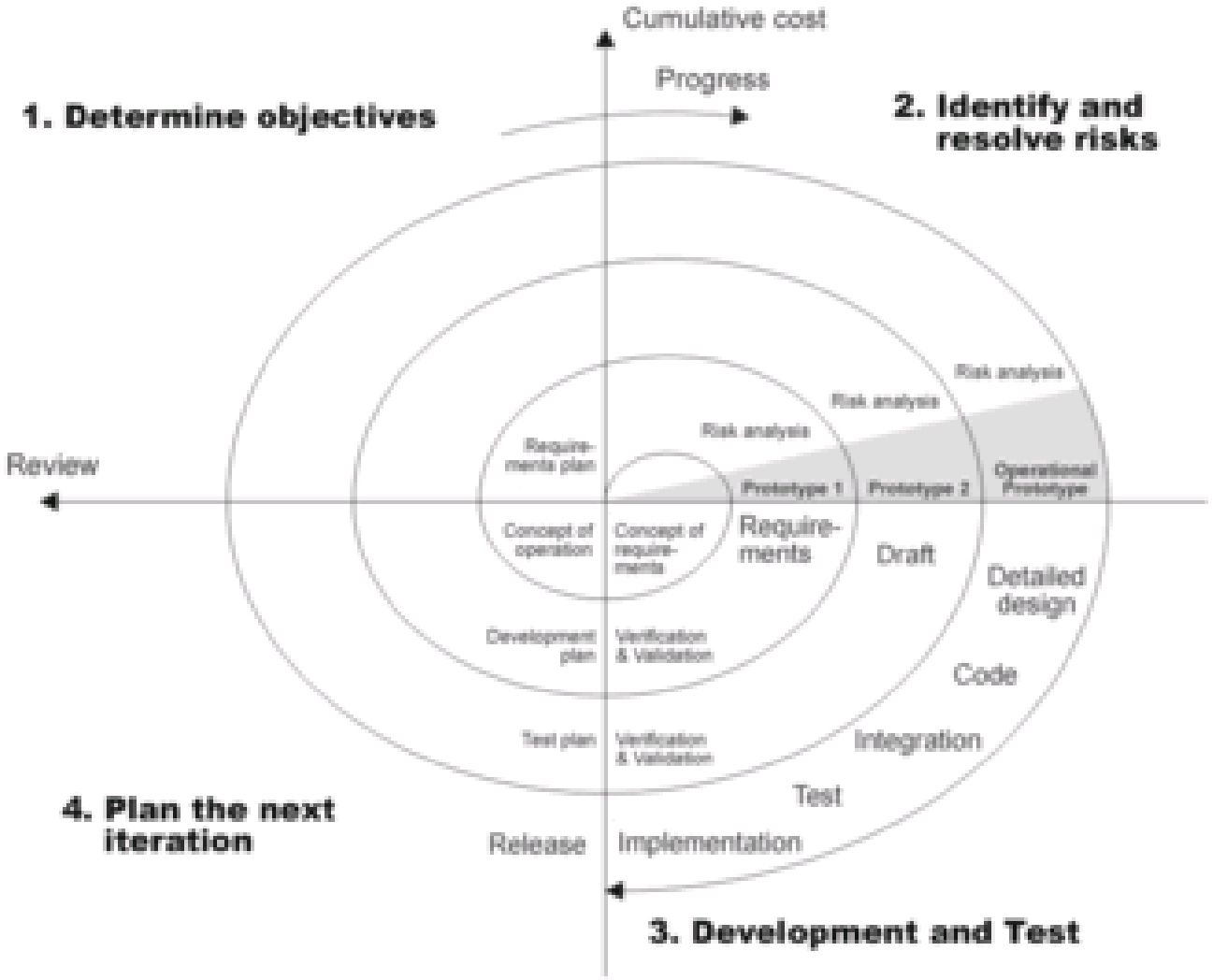
- סבבים (קצרים בד"כ) של
  - הוספת פונקציונליות באופן אינקרימנטלי
  - תזמון לפי איטרציות - מסגרת זמן (שבוע\חודש)
  - אפשר לשלב ביניהם
  - אפשר לתכנן ולממש חלקים במקביל
- יתרונות
  - כל התקדמות קלה להבנה ולניהול
  - מימוש ראשוני של פונקציונלית עיקרית ואח"כ לפי משוב מהלקוח
    - הורדת סיכונים




# המודל הספיראלי \ אב-טיפוס

- 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



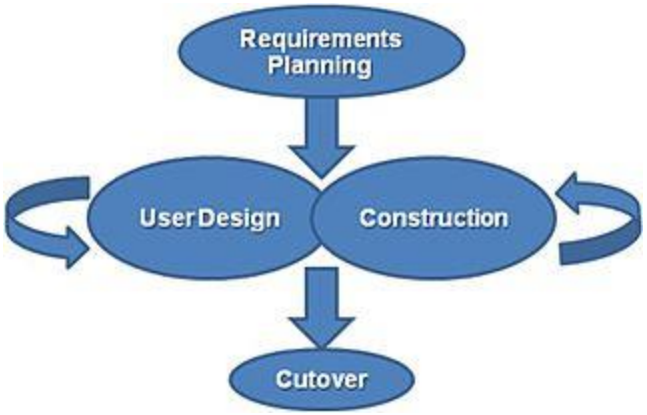
# המודל הספיראלי - יתרונות

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

# וריאציות

# Rapid Application Development

- תכנון מועט בכדי להגיע ל**אב-טיפוס** מהיר
- אוסף שיטות המסתמכות גם על כלי פיתוח מתאימים

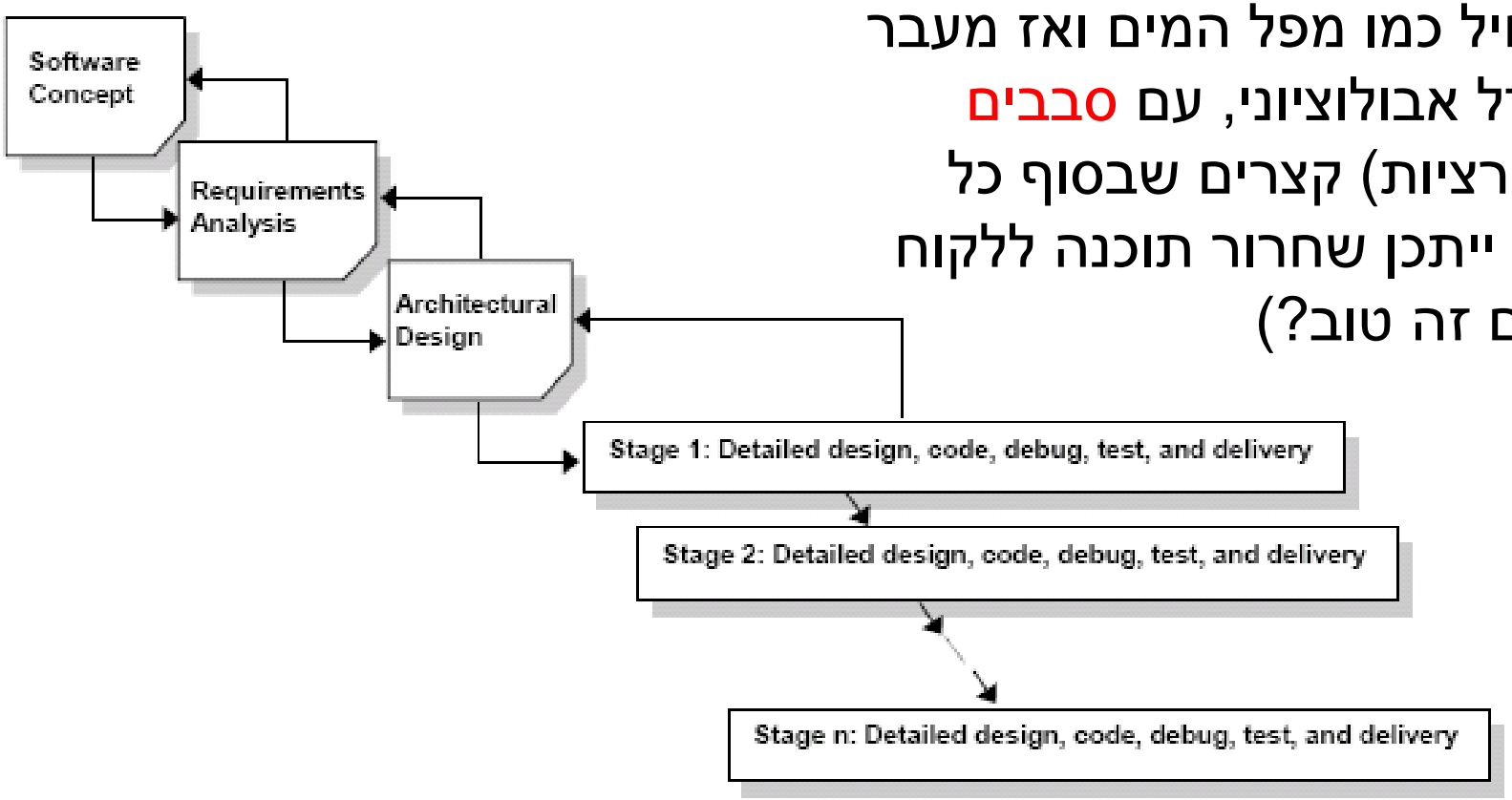


# שחרור בשלבים Staged Delivery

- 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 מזוהות מוקדם
- חסרונות
  - עלות שחרור\אינטגרציה
  - האם תמיד ניתן לחלק?
- היה בשימוש די נרחב

# מודל "שחרור בשלבים"
# Staged/Rapid Delivery

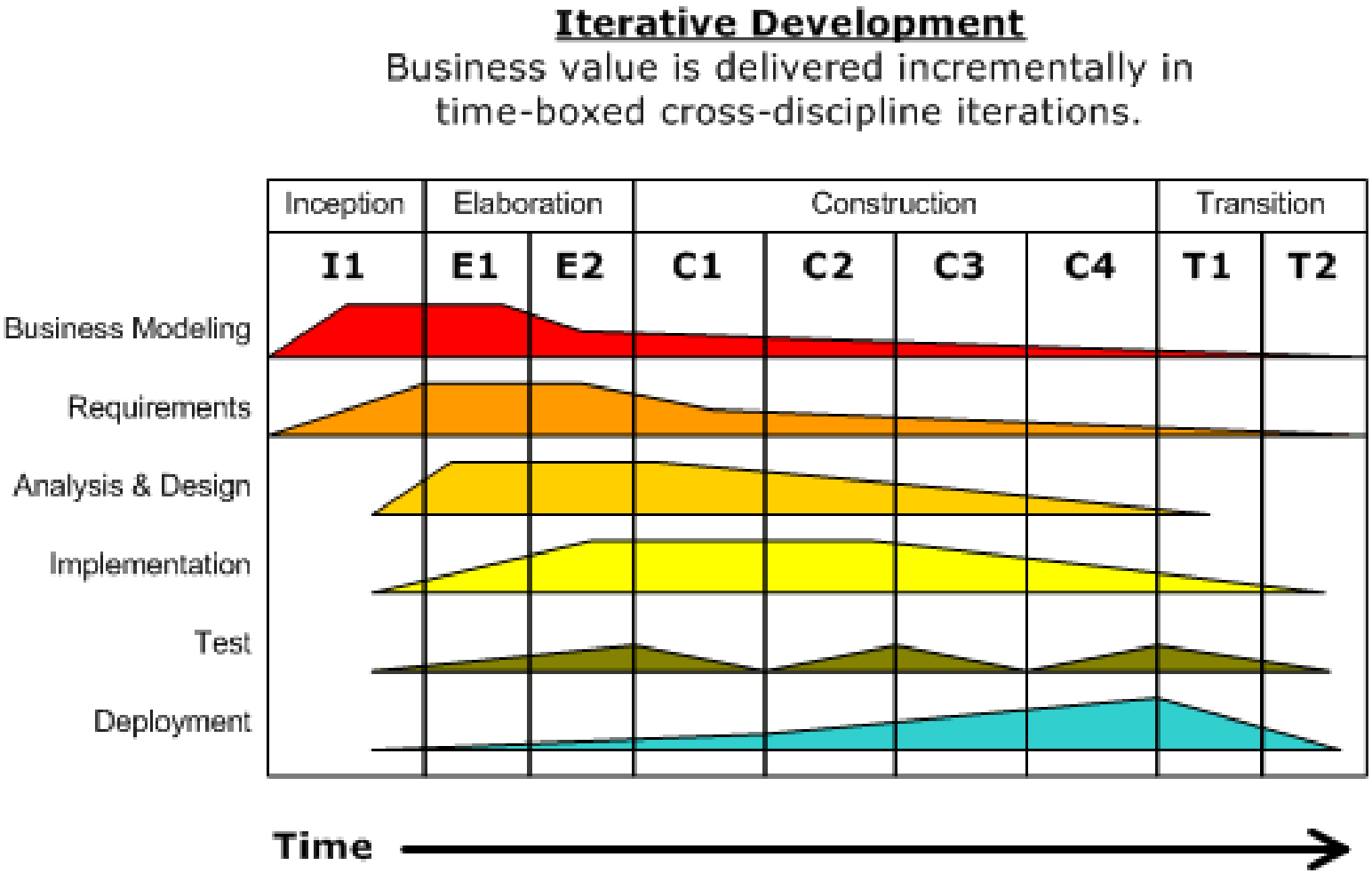
מתחיל כמו מפל המים ואז מעבר למודל אבולוציוני, עם סבבים (איטרציות) קצרים שבסוף כל אחד ייתכן שחרור תוכנה ללקוח (האם זה טוב?)



# Unified Process

- התפתח מתוך המודל הספיראלי
- Rumbauch, Booch, Jacobson 1999
- מעטפת\מסגרת תהליכים לפיתוח מונחה עצמים
  - מיועד להתאמה לפרויקט המסוים
  - UML פותחה כחלק מהתהליך (בהמשך)
- RTC <- IBM/Rational UP
- מתאים לפרויקטים ענקיים
- התפתח ל- Agile RUP, OpenUP

# Unified Process



# 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

# Unified Process

- יתרונות
  - קשר בין ההיבט העסקיים לפיתוחיים
  - כלים מתקדמים
- חסרונות
  - כלים יקרים בד"כ
  - יותר מדי אפשרויות...
  - מתאים בעיקר לפרויקטים גדולים
  - תלות גבוהה במנהלי פרויקטים

# מה אינו נכון בקשר למודלים העיקריים שראינו (מפל המים, ספראלי, RUP)?

1. כולם סומכים על תכנון מפורט ומודדים התקדמות לפי התכנון
2. כולם סומכים על מסמכים מפורטים
3. כולם סומכים על מנהל שיהיה אחראי על הפרויקט
4. כולם משתמשים באיטרציות ואב-טיפוס

# האם יש חלופה?

- שם חלופי למודלים הקודמים – תכנן ותעד
  - האם הם מספקים עמידה בעלויות, זמנים ואיכות?
- האם אפשר לבנות תוכנה ללא תכנון, מסמכים וניהול? או לפחות ללא הפרדה ברורה בין השלבים
- לשמר את מה שהוכיח את עצמו?
- אבל עדיין איך לא לשאר רק ברמת ההאקרים?

# 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**

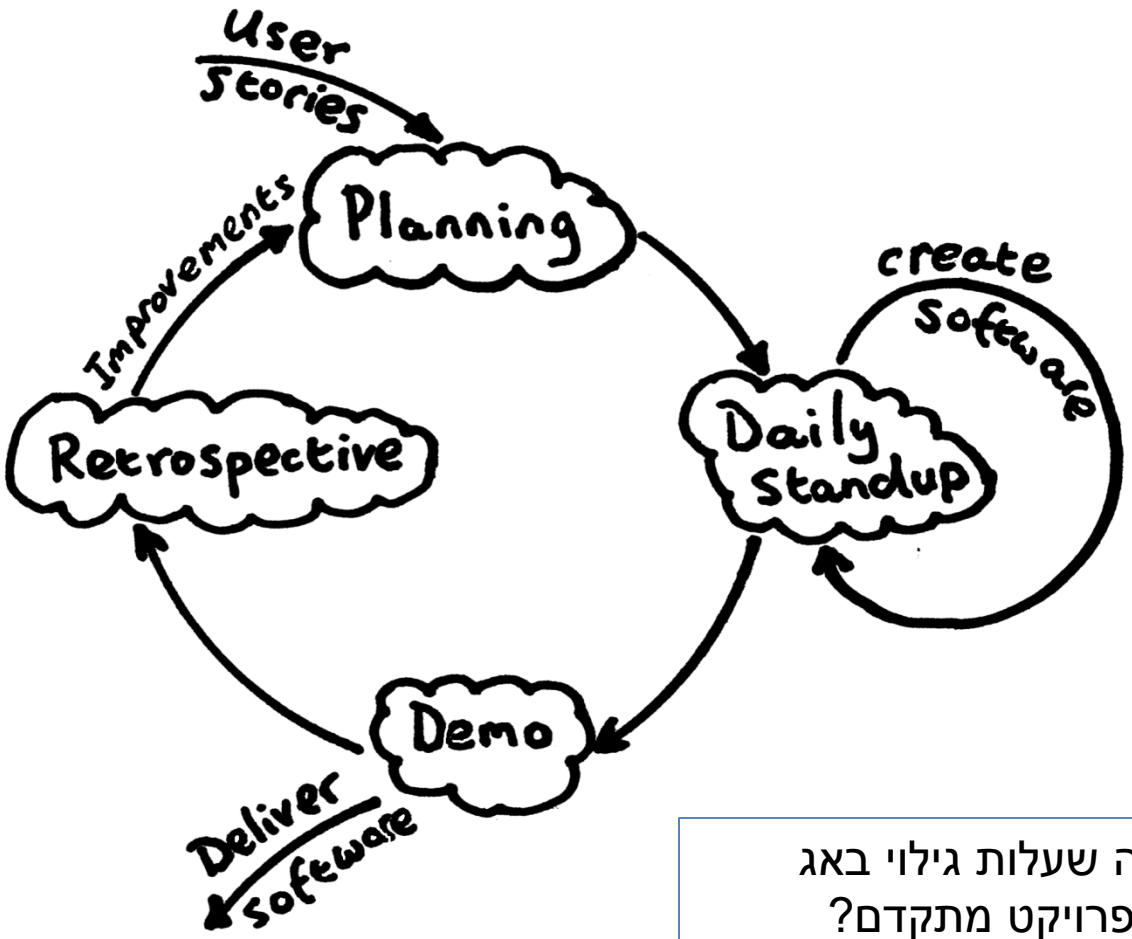
**Customer collaboration over contract negotiation**

**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 (ביקורת נגדית, antiagilemanifesto)

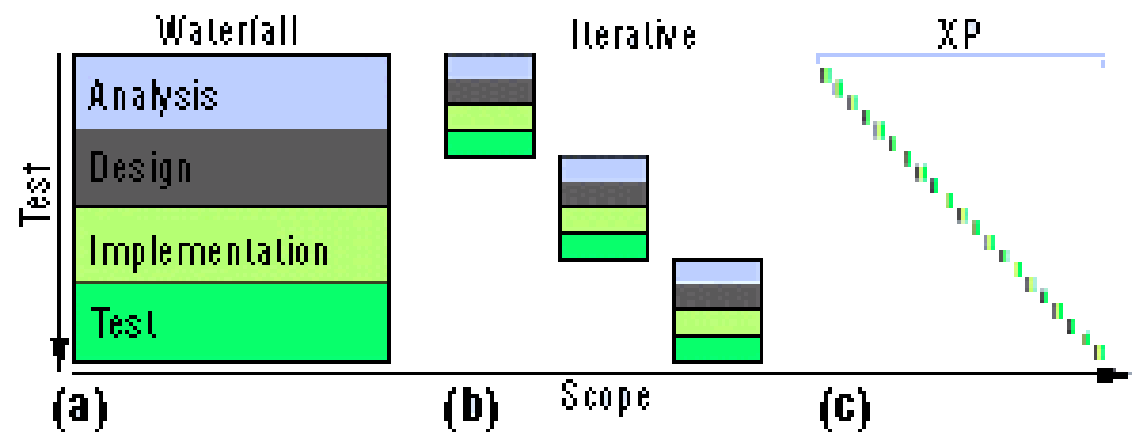
# מודל למחזור החיים העיקרי ב- Agile




מה עם הטענה שעלות גילוי באג
עולה ככל שהפרויקט מתקדם?

# XP – Extreme Programming ‘99

- אם איטרציות הן רעיון טוב, בואו נקצר אותן יותר
- אם פשטות זה טוב, בואו ניתן את הפתרון הפשוט ביותר
- אם בדיקות הן רעיון טוב, בואו נבדוק כל הזמן ואפילו לפני הקוד
- אם סקרי קוד הם רעיון טוב, נעשה אותם כל הזמן ע"י תכנות בזוגות




Kent Beck, “Embracing Change with eXtreme Programming”

# שתים עשרה המיומנויות של XP

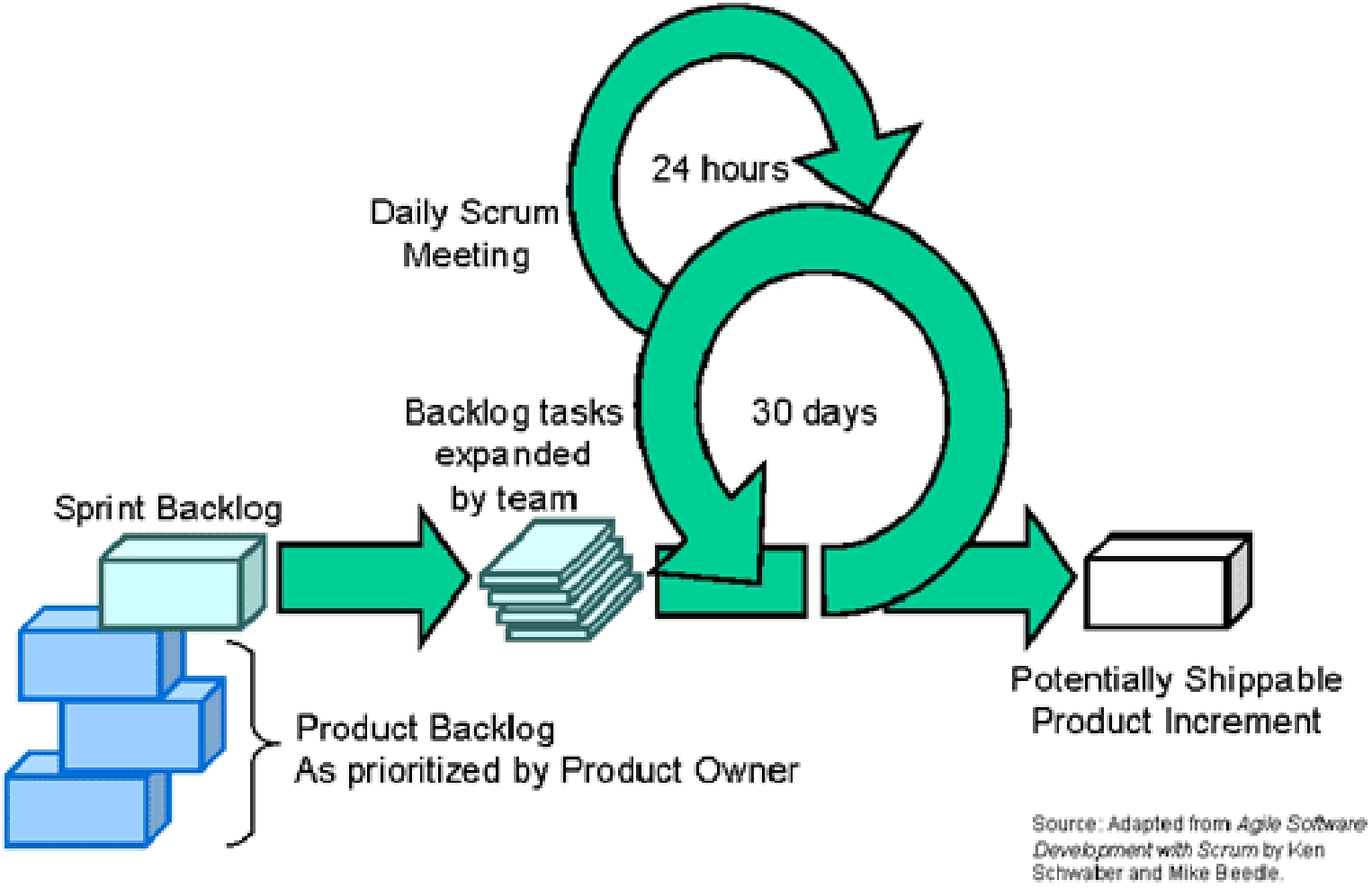
- "משחק" התכנון
- שחרור גרסאות קטנות
- מטאפורה
- תכן פשוט
- בדיקות רצופות
- עיצוב מחדש
- תכנות בזוגות
- בעלות משותפת
- שילובים רצופים
- שבוע עבודה בן 40 שעות
- הלקוח זמין באתר הפיתוח
- כללי קידוד

כיצד באים לידי ביטוי ערכי Agile?

# 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: Crystal Clear, DSDM, MSF, Lean Startup שילוב? שיטה משלכם?
- העיקרון המוביל: התאמה למשוב בזמן-אמת
- נחזור ונבקר בהמשך הקורס, עקרונות שונים
  - מצגת בעברית

# Scrum Lifecycle




מה משיגים בשחרור תכוף?

# אחרים

- Lean
- Kanban
- PSP/TSP
- GROWS

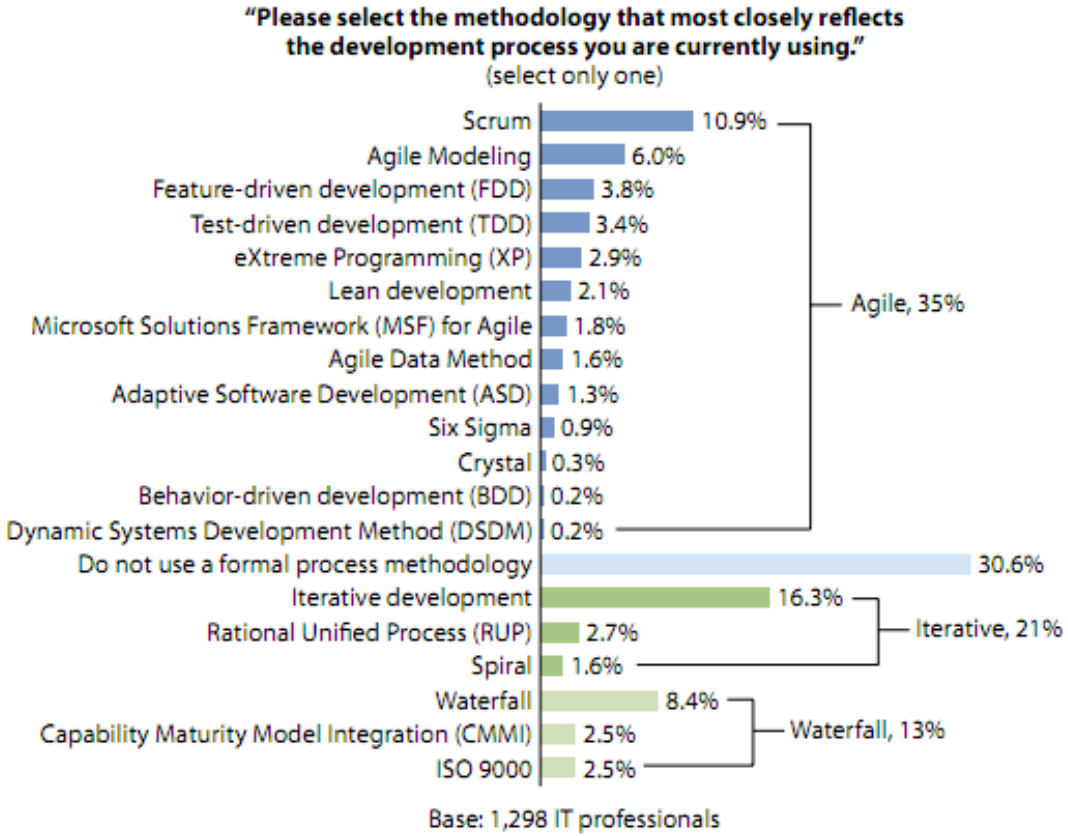
# האם בשימוש?

- Controversial in 2001
  - "… yet another attempt to undermine the discipline of software engineering… nothing more than an attempt to legitimize hacker behavior."
  - Steven Ratkin, "Manifesto Elicits Cynicism,"

  *IEEE Computer*, 2001
- Accepted in 2014
  - 2012 study of 66 projects found majority using Agile, even for distributed teams
- Is it new? 1974 - Lehman's laws of software evolution: 1. Continuing Change — An E-type system must be continually adapted or it becomes progressively less satisfactory

# Agile Adoption: Forester 2009

**Figure 1** Agile Adoption Has Reached Mainstream Proportions



# VersionOne Survey

# מתי להשתמש ב- Agile

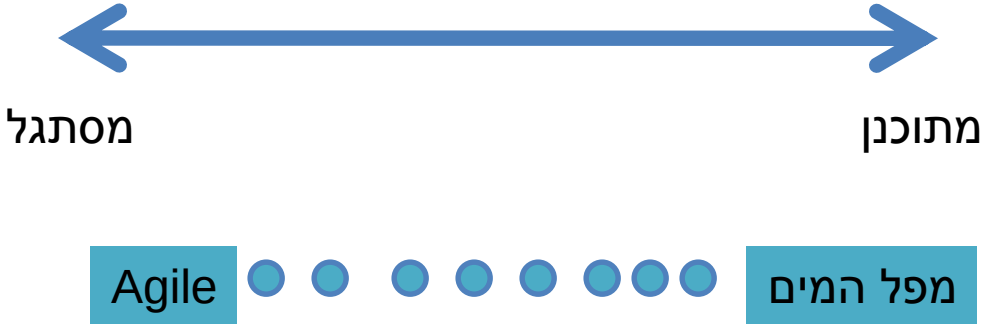
- פרויקט דינמי עם קבוצה קטנה ובעלת מוטיבציה
- דרישות משתנות
- יש לקוח מעונין וזמין




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



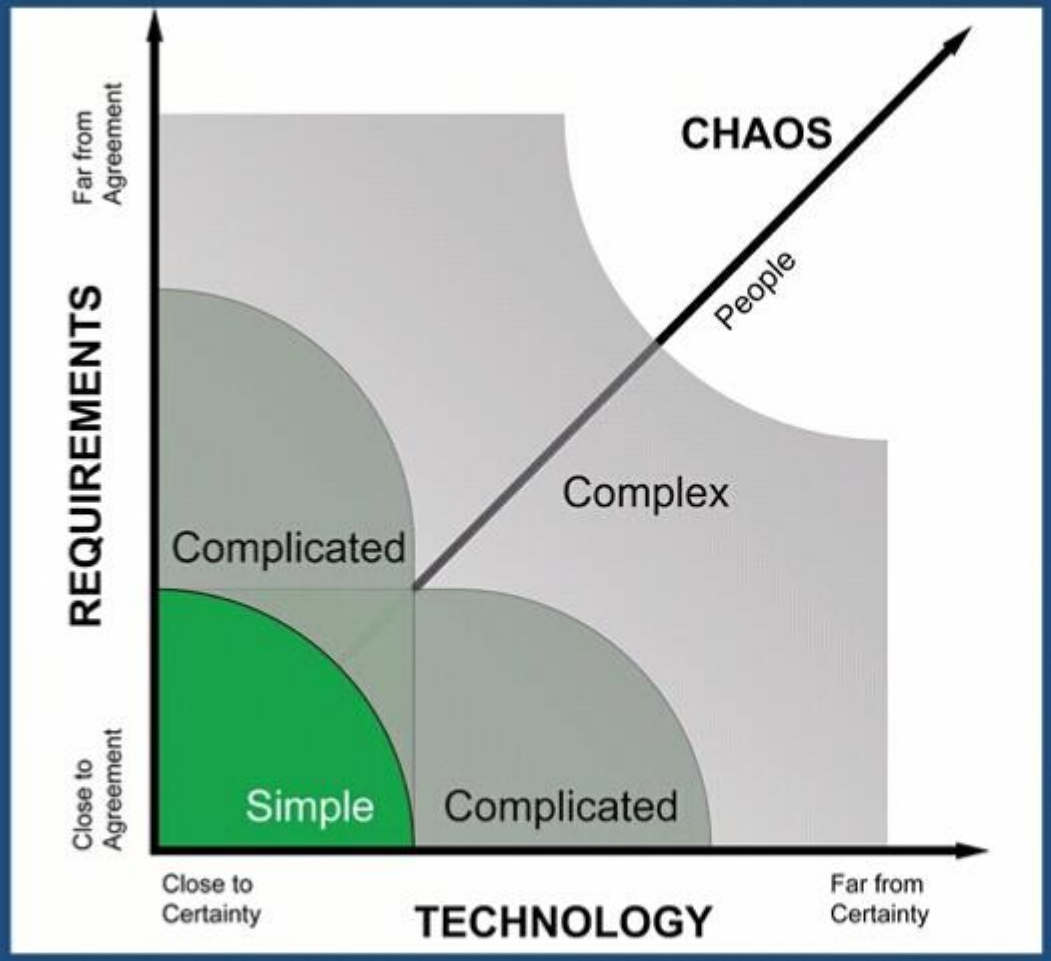
# מדוע יש כ"כ הרבה מודלים?

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

- אף פרויקט אינו דומה לאחר
- תפירת מודל אישי\קבוצתי\חברה\לקוח\בעיה\סיבוכיות\פרויקט
- מתי לבחור? מה קורה אם צריך לשנות?
- האם באמת משתמשים במודלים?

- [Parnas & Clements ‘86]
  A rational design process: how and why to fake it
- Brooks, “No Silver Bullet”.
  Fowler, “CannotMeasureProductivity”
- “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

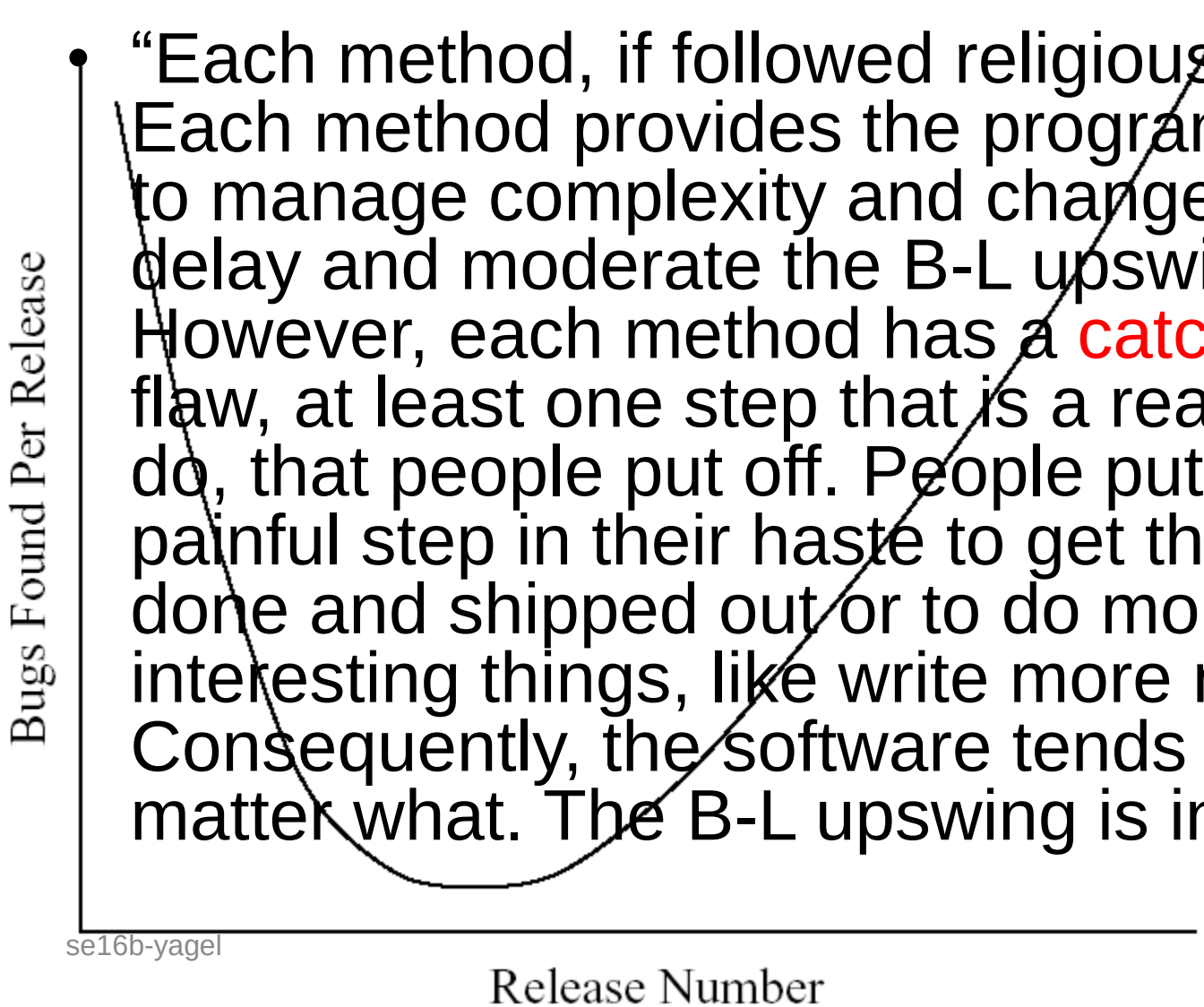
## Complicated
More is known than unknown

## 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.”



# תקנים והסמכות

- תקני US DoD
  - MIL-STD-498 ,DOD-STD-2167A
  - נוהל מפת"ח (תאוריית הקיפודים ופרקטיקת השועלים - גישת הנדסת תוכנה זריזה וישומה בארגון צבאי)
- 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
- קבוצתי: אתחול פרויקט
- אישי: צד לקוח

# עוד על עבודת צוות ותהליכים – משחק המרשמלו

- http://marshmallowchallenge.com
- סרטון תוצאות TED – מי הכי מצליח?

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.

# לסיכום

- האם מודל לתהליך יכול לעזור לפיתוח פרויקט תוכנה? (Cargo Cult SE)
- מהו המודל המתאים ביותר?
("No One Size Fits All", "No Silver Bullet")
- ישנם מודלים נוספים (למשל של קוד פתוח, דור5?)
  - **Build-Measure-Learn** – **http://theleanstartup.com/**
- בפרויקט?
- החלקים שנלמד בהמשך יכולים בד"כ להשתלב ללא קשר למודל הפיתוח הספציפי שנבחר?