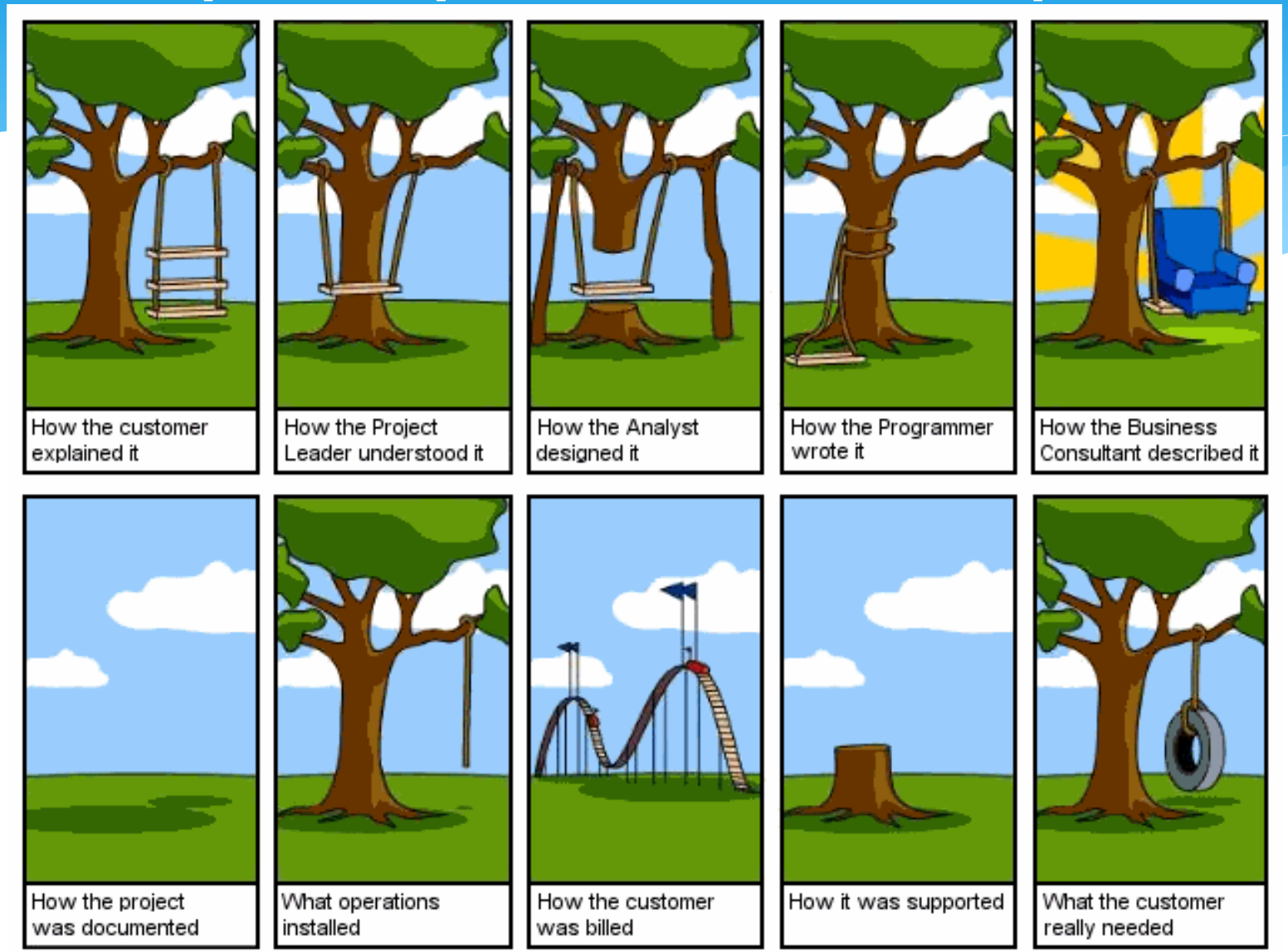
מתודולוגיות בדיקה

QA MASTERS

נושאי השיעור

* הקדמה- למה צריך בדיקות ?
* מערכת
* בדיקות תוכנה
* תכנון בדיקות
* סביבות עבודה
* מודלים של פיתוח תוכנה ומחזור חיים של פרויקט הבדיקות

הקדמה – למה צריך בדיקות?



למה צריך בדיקות?

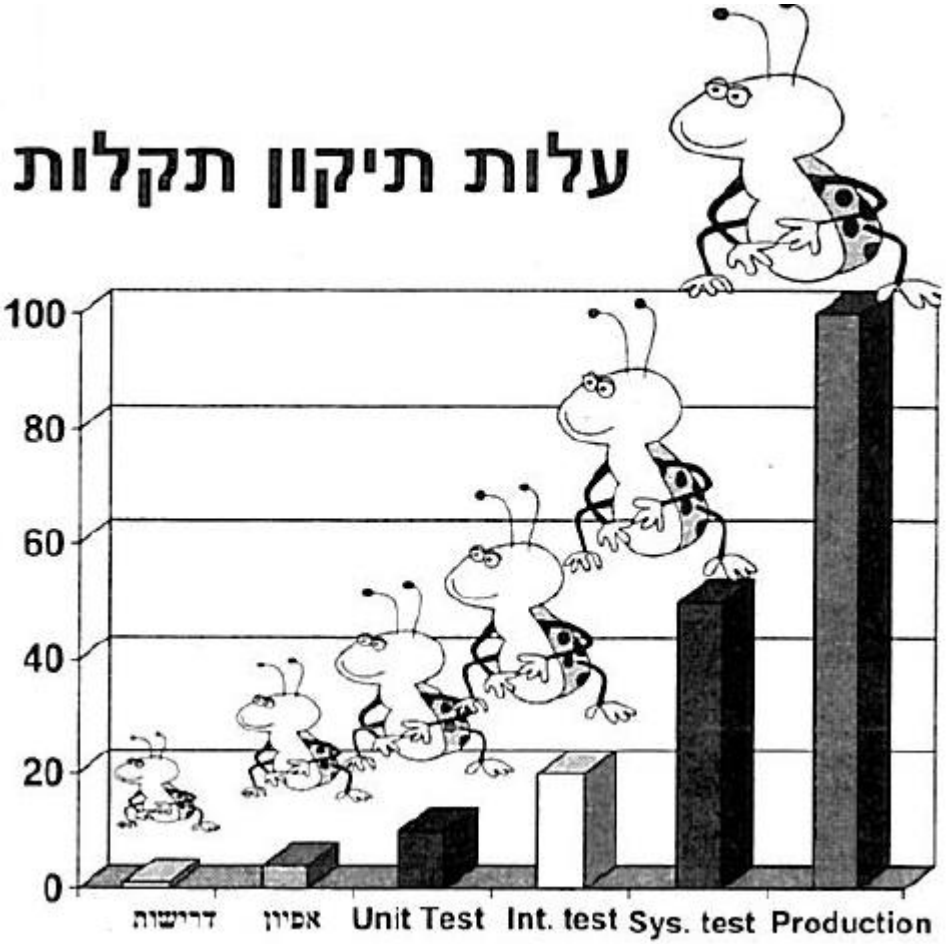
* סטטיסטיקה מראה שישנה גלישה של 100%-200% בזמני הפיתוח ובתקציב הפיתוח.
* מפתחים משקיעים כמעט 30% מזמנם בתיקוני תקלות
* 40% מהפרויקטים מבוטלים או נעצרים ללא עליה לאויר
* טכנולוגיות מורכבות יותר, כלי פיתוח דינאמיים ודרישה של הלקוחות לחדשנות ולאיכות

למה צריך בדיקות?

Quality Assurance= QA

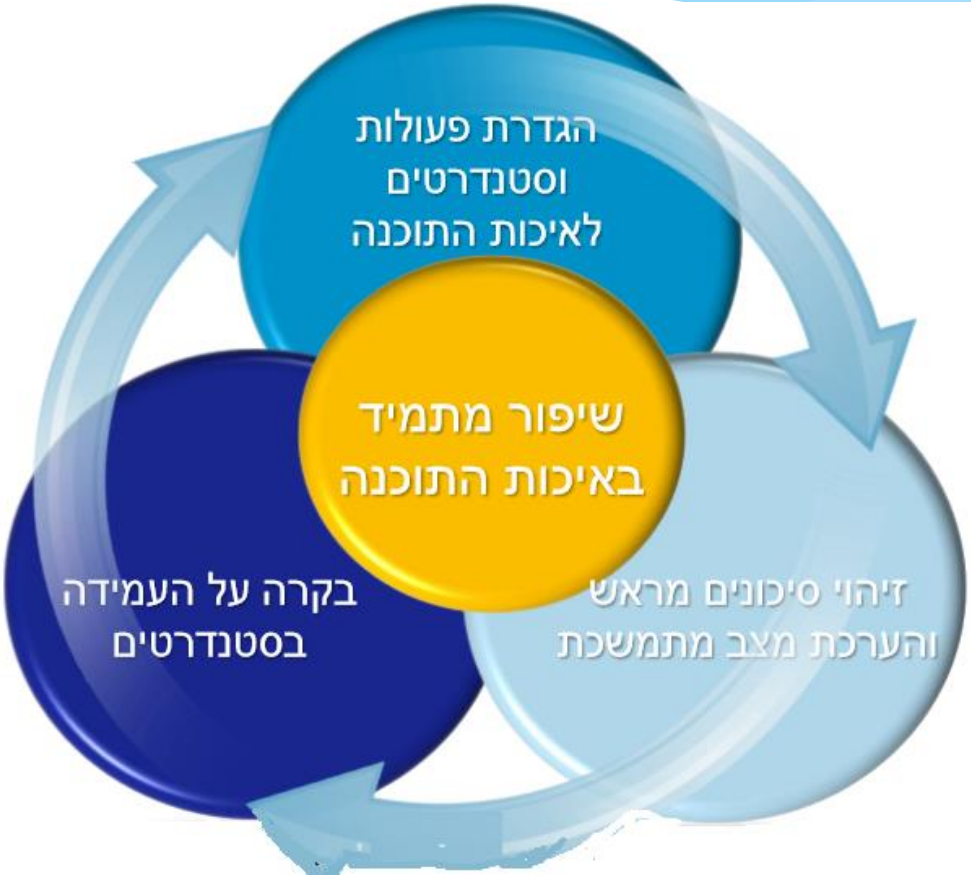
* הבטחת איכות תוכנה (QA) הינו מכלול התהליכים בארגון הבאים להבטיח את איכות תפקוד הארגון ומוצריו.
* מיועד לצמצם נזקים כבר בשלבים הראשוניים.

למה צריך בדיקות?



ככל שמציאת התקלות
תעשה מוקדם יותר כך
העלות תהיה קטנה יותר

למה צריך בדיקות



מערכת

* אוסף של תוכניות מחשב בשילוב עם הגדרת מאגרי נתונים מתאימים.
* מערכת איכותית :
  - מספקת את דרישות הלקוח
  - עונה על צרכיו העסקיים ופשוטה לתחזוקה
  - אין בה תקלות חמורות

בדיקות תוכנה
Software Testing

* תהליך פורמאלי של הפעלת רכיב תוכנה לפי תוכנית שנכתבה ואושרה על סמך דרישות המערכת במטרה לאתר מקסימום שגיאות מוקדם ככל האפשר.
* התהליך נועד לאפשר לבעלי העניין לקבל מדד לאיכות המוצר ולעמידתו בדרישות ולאפשר קבלת החלטות לגבי המשך תהליך הפיתוח/ העלאה לייצור.
* תהליך הבדיקות הינו פרויקט מתוכנן מראש ואנשי הבדיקות הינם הסמכות המקצועית,לכן עליהם להכיר מתודולוגיות, נהלים כלים וטכנולוגיות מתקדמות.

בדיקות תוכנה- אימות ותוקף

Verification= אימות

האם אנו מפתחים נכון את המוצר ?

אימות ע"י בחינה ועדויות תוצאתיות שדרישות ספציפיות אכן מולאו. וריפיקציה מתבצעת כחלק מהבדיקות הסטטיות כלומר האימות מבוצע כנגד מסמכים ואין צורך בהרצת בדיקות. (התאמה לתכנון).

Validition = תוקף

האם אנו מפתחים את המוצר הנכון ?

ווידוא ע"י בחינה ועדויות תוצאתיות שהדרישות לשימוש מסוים או יישום מסוים אכן מולאו. ולדיציה מתבצעת כחלק מהבדיקות הדינאמיות, כלומר ניתן לבצע ולידציה רק ע"י הרצת הבדיקות (התאמה לדרישות).

בדיקות תוכנה - סטטיות ודינאמיות

בדיקות סטטיות

סקירת הקוד, מסמכי האפיון וכדומה מבלי להשתמש במערכת הנבדקת.

(Sut= system under test)

איתור: חריגה מסטנדרטים בקוד, כשלים במסמכי האפיון ובתכנון, חוסר כיסוי דרישות

בדיקות דינאמיות

בדיקות HANDS ON

הרצת התוכנה הנבדקת

השוואת התוצאות המתקבלות לתוצאות הצפויות

תכנון הבדיקות

* בכדי לתכנן את מערך הבדיקות בצורה איכותיות עלינו להתייחס למספר נתונים:
  - **סוג מערכת**
  - **דגשים עיקריים : דגשי המערכת, צרכי הארגון והלקוח**
  - **הגדרות ומושגים בתכנון הבדיקות**
  - **הגדרות ומושגים בביצוע הבדיקות**
  - **גישות לבדיקות תוכנה**

תכנון הבדיקות – סוג המערכת

סוג המערכת	הסבר	איך נבדוק
IT- Information Technology - מערכות מידע/עיבוד נתונים	מבוססות על נתונים לצורך שליפה,הצגת מידע,עדכון נתונים. האפיון נעשה בראיה מרחבית להתאמה למס ארגונים (כספים, משאבי אנוש). **או מערכות מידע מוכללות ERP= Enterprise Resource Planning)**	ההתמקדות היא בדיקת התהליכים העסקיים כפי שהוגדרו ע" הארגון ופחות בדיקת התוכנה עצמה. במקום דרישות ואפיון קיים שלב המגדיר את התהליכים העיסקיים (המסמך נקרא Blue Print).
Real-time\Embedded מערכות זמן אמת/משובצות	מערכות מחשב שיש בהן דרישות לביצועים במגבלות זמן. תכונות מיוחדות: אינן מבוססות על מאגרי מידע אלא על מכשור, ציוד יקר ולעיתים מסוכן. לדוגמא: מערכת השולטת על טילים וכו'.	הכנות מורכבות , הוספת קוד זמני למערכת שיוציא פלט חשוב לצורך בקרה, בניית סימולטורים
Command & Control מערכות שו"ב – שליטה ובקרה	מערכות המאפשרות לדרג הניהולי: קבלת מידע ממקורות שונים, הצגת תמונת מצב אחודה של המערכת בכל זמן נתון, פעילות השרתים, תקינות מסד הנתונים. שליטה על פעולות של הרכיבים השונים: הפעלת שרת, שליחת הודעות וכו' דוגמא: מערכת לניטור שרתים	בניית סביבת עבודה הכוללת את כל הרכיבים הנדרשים. התמקדות בתקשורת בין הרכיבים ואמינות המידע המשתקף
תוכנות מדף	תוכנה המופצת לכל רוכש כפי שהיא. אין לקוח מוגדר לדוגמא: מערכת ניתוב שיחות.	יש לשים דגש מיוחד על: בדיקות התקנה, בדיקות תאימות
מערכות בתחזוקה	מערכת ששלב הפיתוח שלה הסתיים והיא הועברה לשלב הייצור ושימוש שוטף ע"י המשתמשים	שינוים ותוספות, בדיקת תקלות שתוקנו, בדיקת חלקים לוודא שלא הושפעו

תכנון הבדיקות – דגשים עיקריים

כדי לתכנן את הבדיקות נידרש לענות על מספר שאלות בינהן :

* האם מערכת חדשה – האם ישנה הסבה?
* האם מערכת לשיפור מערכת קיימת/ הטמעת חבילת תוכנה?
* האם היא מתממשקת למערכות אחרות (on-line או Batch)?
* האם מערכת on-line (מערכת בורסאית) /Batch (אצווה) – ריצת לילה
* האם מערכת אל- כשל / החייבת הגנה מקסימאלית/ מחייבית זמני תגובה קצרים
* האם מיועדת לשימוש לטווח ארוך/ קצר
* מיהם הלקוחות – מנהלים / עובדים
* מהי כמות ואיכות המשתמשים
* מהם לוחות הזמנים
* האם ישנן מגבלות של כוח אדם, האם נדרש כוח אדם מנוסה

תכנון הבדיקות – הגדרות תכנון

* **דרישה (Requirement)**

תיאור של תכונה או תפקוד הנדרש מהמערכת המפותחת למימוש פעילות עיסקית של הארגון. **לדוגמא** : תוספת של אופציית כניסה מהירה באפליקציית בנק דיסקונט.

https://www.youtube.com/watch?v=_uGYbimtzXI&feature=youtu.be

* **מבחן (TEST) או מקרה בדיקה (TEST CASE)**

אוסף צעדים וערכים שמטרתם לתכנן ולממש מטרת בדיקה מסוימת. **לדוגמא:** אוסף הצעדים להפעלת כניסה מהירה באפליקצייה עם קלט חיובי ושלילי ובדיקת התוצאות

* **אתר הבדיקה (SITE)**

הסביבה בה מריצים את תרחישי הבדיקות. לדוגמא: סביבת בדיקות וסביבת pre-production

תכנון הבדיקות – הגדרות ביצוע

* **מנת הרצה (Test Set)**

אוסף של מקרי בדיקה בעלי מכנה משותף הרצים בזה אחר זה בסבב בדקות אחד

* **סבב בדיקה (Test Cycle)**

אוסף מנות הרצה המורצות על גרסת מערכת אחת בטווח תאריכים מוגדר

* **סבב בדיקה חוזר (Re-testing)**

סבב שמטרתו לבדוק את המערכת לאחר שתוקנו בה ליקויים שאותרו בסבבים קודמים

תכנון הבדיקות – גישות לבדיקות

* **קופסא לבנה White Box**
* **מיקוד בקוד ובמהלך עיבוד הנתונים : בדיקת מסלולים , תנאים , זרימת מידע ולולאות בתוך הקוד.**
* **ניתוח חוקיות בשפת הפיתוח, כיסוי לוגי Flow Graph דרך כל קטעי הקוד, בדיקת איכות הכתיבה ע"י Code Review.**

יתרונות	חסרונות
כיסוי מקסימלי ואופטימלי של הקוד באמצעות מקרי הבדיקה	מורכב וממושך מאוד, קשה לפיענוח מיושם ע"י תוכניתנים או בודקים עם ניסיון בכתיבת קוד.

תכנון הבדיקות – גישות לבדיקות

* **קופסא שחורה Black Box**
* **תפקוד התוכנה קלט לעומת פלט (ללא התייחסות לקוד)**
* **הפעלת המערכת כפי שתופעל ע"י המשתמשים על בסיס פעולה ותגובה**
* **בדיקות ביצועים**
* **בדיקות כשל והתאוששות**
* **בדיקות MTBF = משך הזמן שעובר בין תקלה אחת לבאה. = מדד לאיכות.**

יתרונות	חסרונות
פשוט לביצוע ידנית ואוטומטית. מופעל ע"י קלט ובדיקת הפלט בלבד	אינו בודק את יעילות ונכונות הקוד מקרי הבדיקה תלויים בערכי הקלט

סביבות עבודה

* סביבת הפיתוח (Development Environment)
* סביבת הבדיקות (Testing Enviroment)
* סביבה תואמת ייצור (Pre – Production)
* סביבת הייצור (Production Enviroment)

	פיתוח	בדיקות	ייצור
פעילות	מתכנתים	בודקים	משתמשים
תכולה	תוכנה בפיתוח	תוכנה נבדקת ונתונים לבדיקה	תוכנת אמת נתונים "חיים"

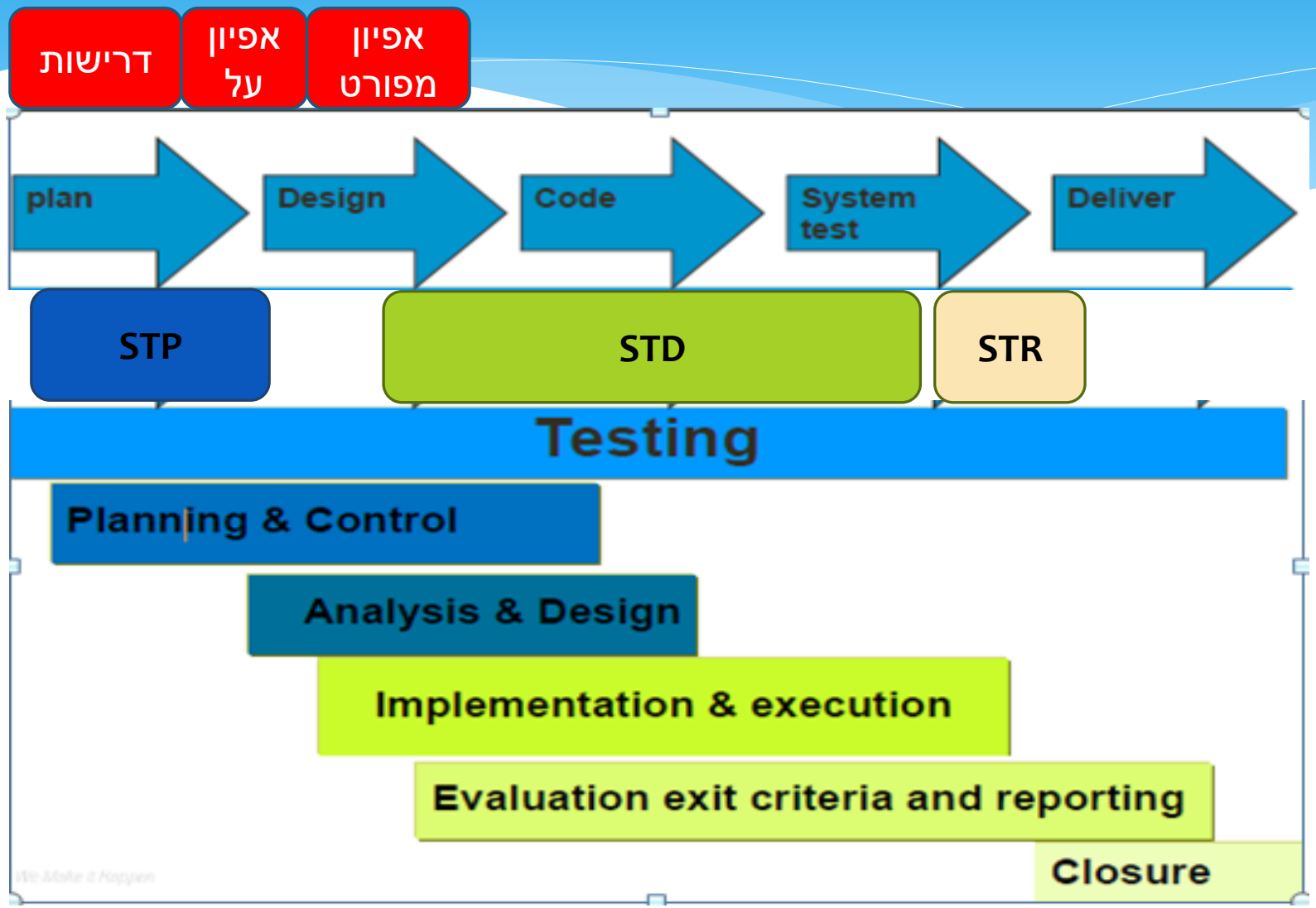
קידום = PROMOTION

הנחתה = Demotion

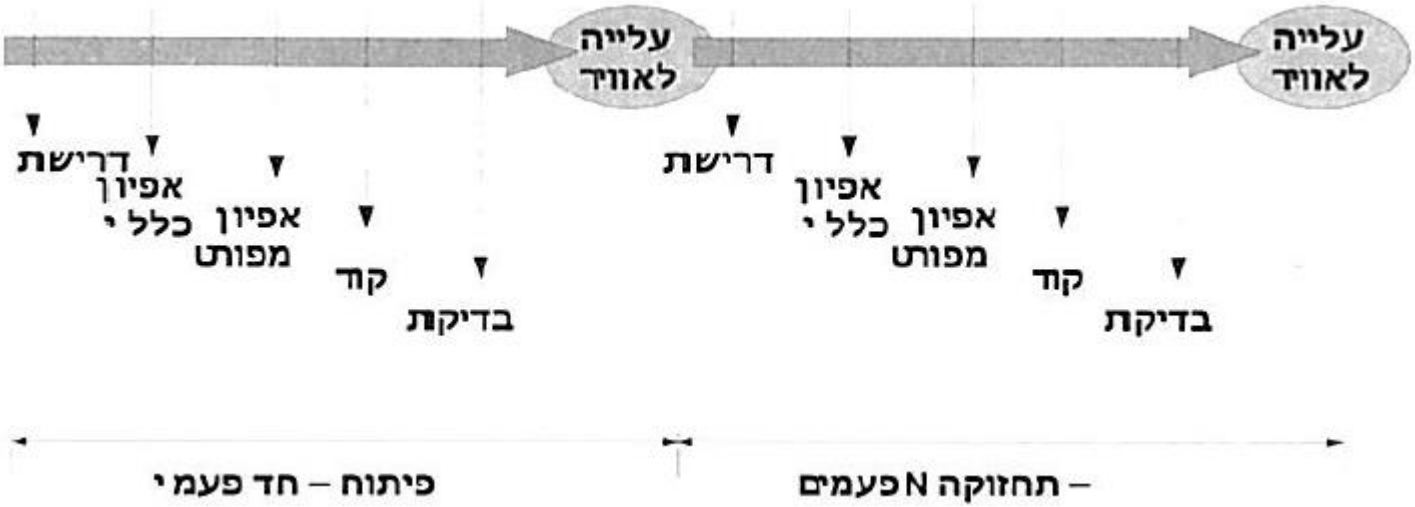
קידום = PROMOTION

הנחתה = Demotion

מחזור חיי המערכת



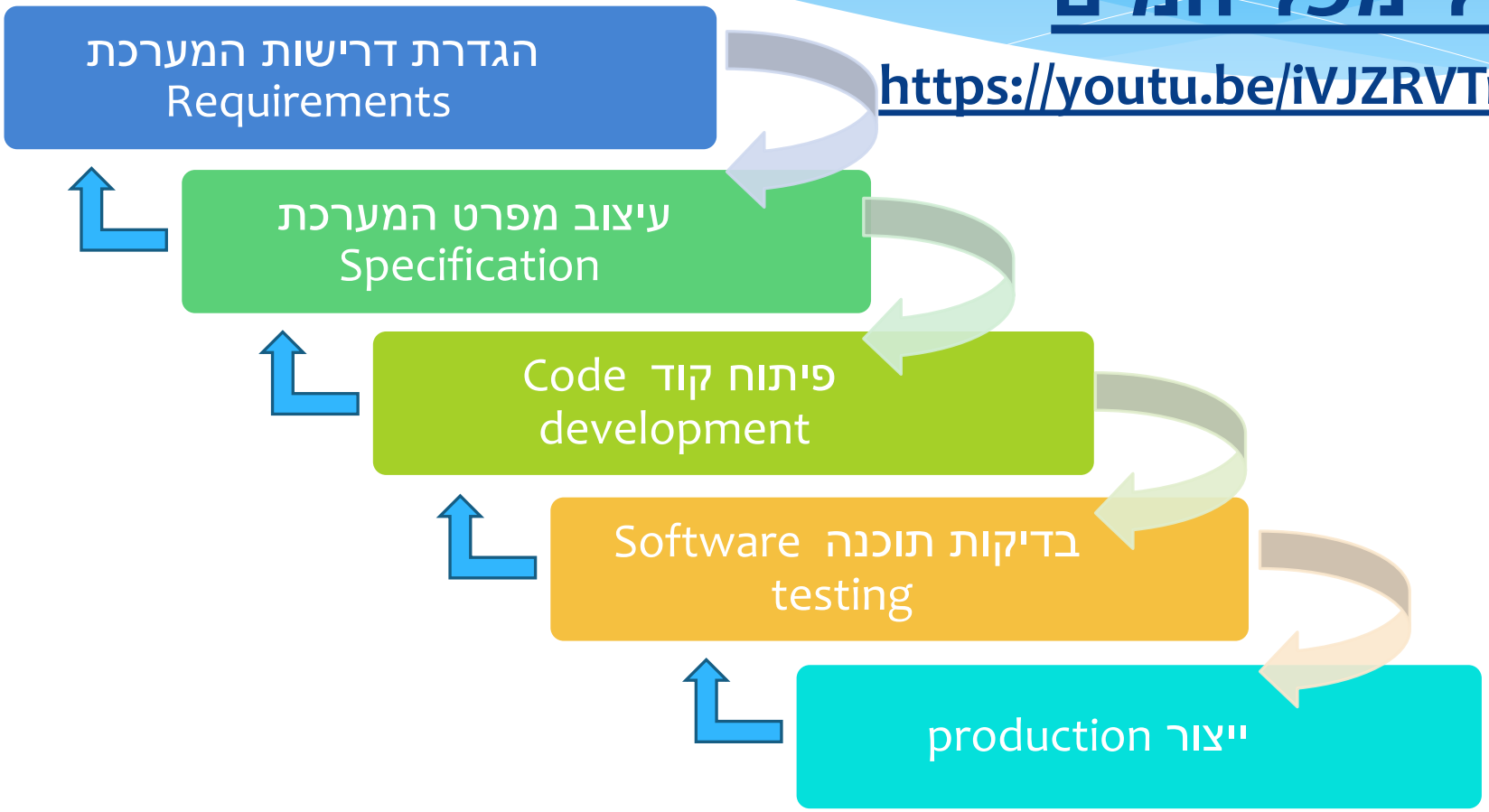
מחזור חיי המערכת



מודלים של פרויקט

מודל מפל המים

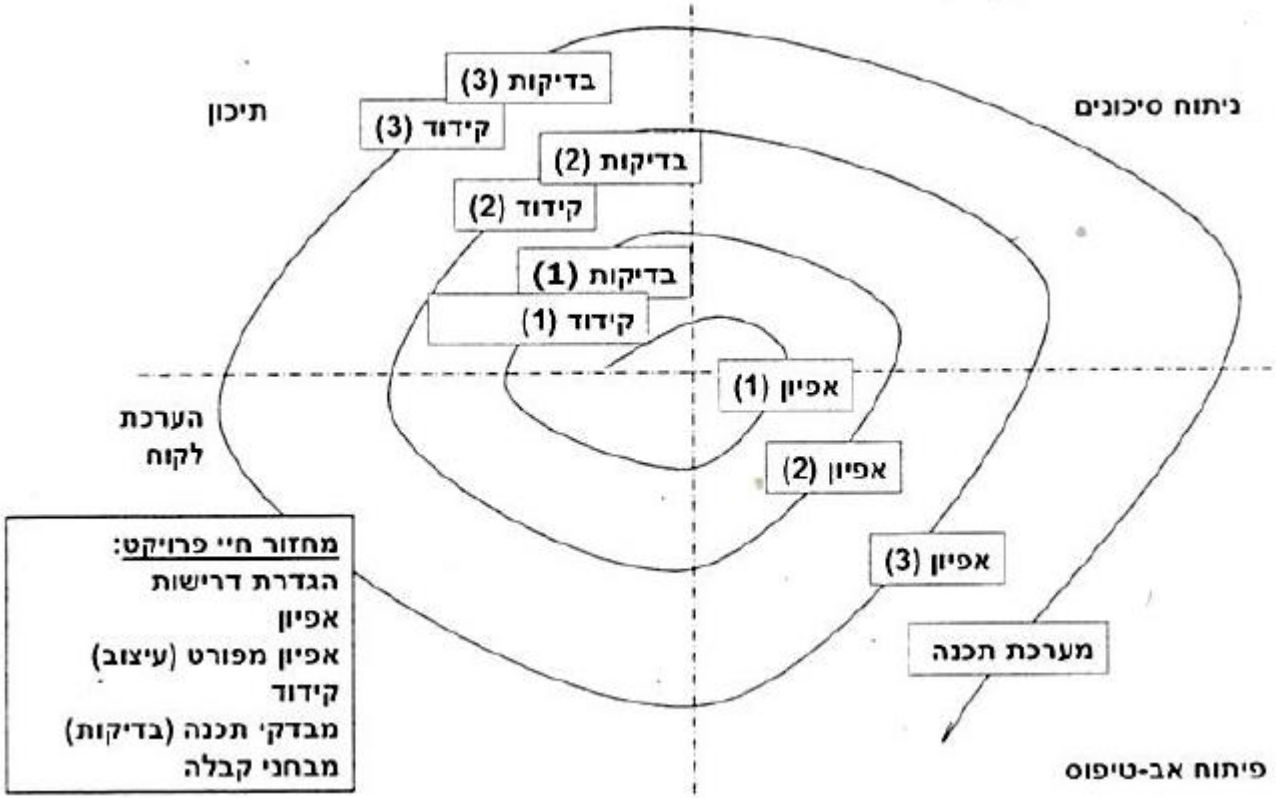
https://youtu.be/iVJZRVTrpu4



מודלים של פרויקט

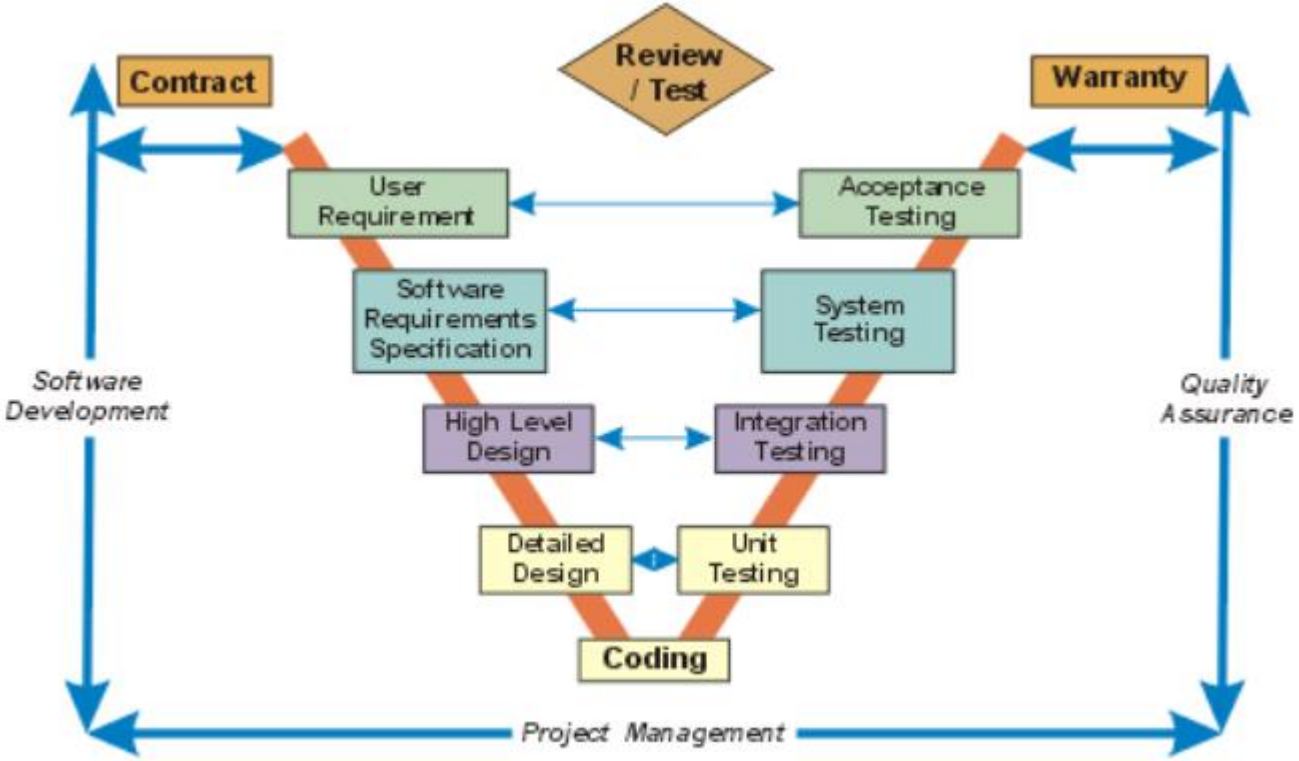
המודל הספירלי

שילוב מפל המים ואב טיפוס



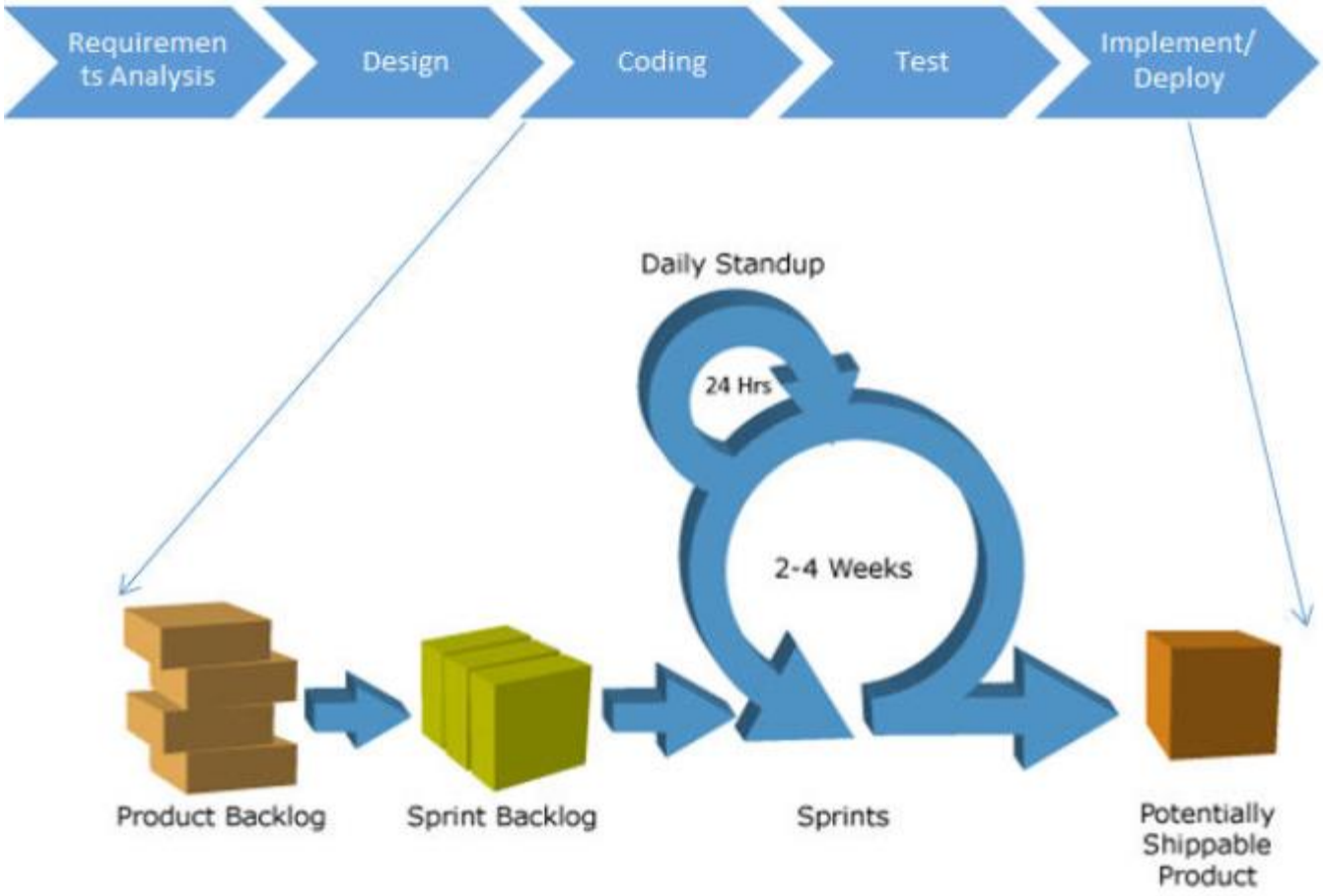
מודלים של פרויקט

מודל ה- V



מודלים של פרויקט

AGILE



מודלים של פרויקט

AGILE

* על פי מודל זה הבדיקות הן המובילות. דורש אנשי בדיקות באיכות גבוהה
* מעורבות צוותי QA לכל אורך התהליך
* אינטגרציה גבוהה בין הלקוח ל- QA ולפיתוח
* המטרה במודל זה הינה לקדם תקשורת בין אנשים תוך דגש על קוד עובד (יותר מאשר תיעוד) כמו כן הלקוח הינו שותף פעיל (לא בהכרח רק על פי החוזה) וכל שינוי הינו מבורך ← גמיש וזריז ← זמיש

מודלים של פרויקט

AGILE - מונחים חשובים :

* SPRINT= איטרציית פיתוח מהירה וממוקדת
* Extreme Programming / SCRUM = טכניקות נפוצות לשימוש
* TDD = Test Driven Development – תכנות ע"י יצירת בדיקות, וכתיבת קוד ע"מ שהן יעברו
* JIT= Just In Time ← לא כותבים קוד תשתתי למען העתיד אלא רק מה שנדרש לאיטרציה הנוכחית.
* User Stories = התחליף של Agile לאפיון מפורט
* Cowboy Stories = מתודולוגיה בה למפתחים יש אוטונומיה בתהליך הפיתוח- בדיוק מה ש- Agile **הוא לא**

מודלים של פרויקט

AGILE - יתרונות וחסרונות:

יתרונות	חסרונות
הפרויקט לא מאבד מיקוד הסיכוי לבאגים חמורים בפריסה הסופית קטן אפקטיביות גבוהה מיומנות הצוות עולה שביעות רצון הלקוח עולה תוצאות טובות	ארגונים גדולים מתקשים במימוש של Agile לעיתים זה רק תירוץ להתנהלות לקויה מקשה על התנהלות החוזית מול הלקוח לא מתאים לפרויקטים תשתיתיים עבודה בלחץ גבוה לאורך זמן עלולה "לשרוף" את הצוות.

מחזור חיים - סיכום

מחזור חיים
פיתוח תוכנה

מחזור חיים
בדיקות תוכנה