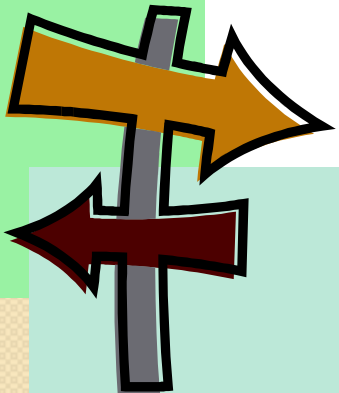


רמות וסוגי בדיקה

70	• רמות בדיקה
72	◦ Unit Test
75	◦ Integration Test
82	◦ System Test
85	◦ Acceptance Test
91	• סוגי בדיקה
93	◦ בדיקות פונקציונאליות
96	◦ בדיקות לא-פונקציונאליות
107	◦ בדיקות תלויות מבנה (בדיקות קופסא לבנה)
109	◦ בדיקות שינויים
112	◦ בדיקות תחזוקה

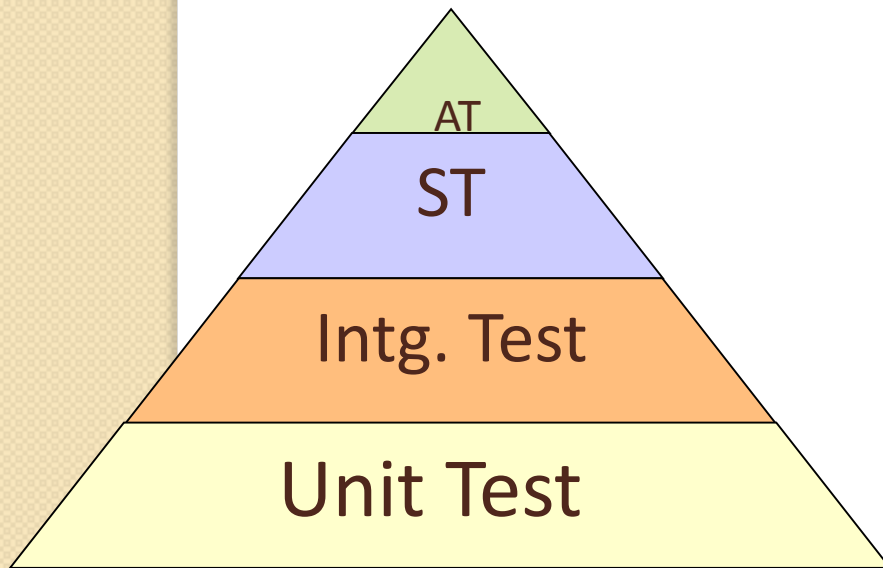


רמות בדיקה

- **רמת בדיקה:** קבוצה של פעילויות בדיקה המאורגנות ומנהלות ביחד, עם אחריות מוגדרת בפרויקט

- **רמות הבדיקה:**

- Unit Test
- Integration Test
- System Test
- Acceptance Test



רמות בדיקה

- כל רמת בדיקה כוללת:

- Test objective - מטרות לרמה זו

- Test basis - מסמכי הבסיס מהם גוזרים את תסריטי הבדיקות

- Test object – הרכיב הנבדק

- תקלות אופייניות הנמצאות ברמה זו

- Test Environment - סביבת הבדיקות

- גישות ואחריות ספציפיות לרמה

Unit Test / Component Test

בדיקות יחידה

- בדיקת רכיב תוכנה בודד, הרכיב הקטן ביותר הניתן לבדיקה

Unit, Class, Object, Component, Program, Data
conversion program, Database model

- מקרי הבדיקה נבנים על סמך מבנה הקוד:

- קופסא שחורה – פונקציונאליות: מה עושה הרכיב

- קופסא לבנה – מבנה הרכיב: כיצד הוא עובד

- בדיקת מאפיינים לא-פונקציונאליים ספציפיים

- בדיקות עמידות, דליפות זיכרון ושימוש במשאבי מחשב

- תקלות מתוקנות "על המקום" ע"י המפתח, ואינן מתועדות

Unit Test / Component Test

- Test basis:

- Component Requirements

- מסמכי Detailed Design של הרכיב ומבנה הנתונים (Data model)

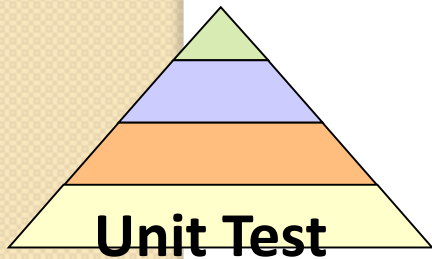
- קוד

- אחריות: המפתחים

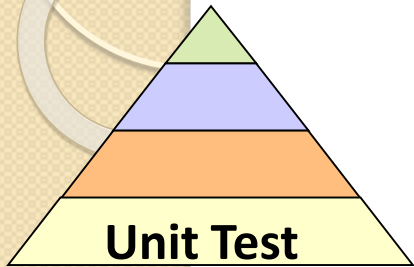
- Test harness - סביבת הבדיקות: סביבת הפיתוח הכוללת:

- Driver – קטע קוד המדמה קריאה/הפעלת הרכיב הנבדק

- Stub - קטע קוד המדמה פונקציה/קוד המופעל ע"י הרכיב הנבדק



Unit Test / Component Test



- 2 גישות עיקריות:
 - הגישה הקלאסית:



- Test-Driven Development – TDD (First):



Integration testing

בדיקות אינטגרציה

- בדיקת האינטגרציה בין רכיבי המערכת ובין מערכות:

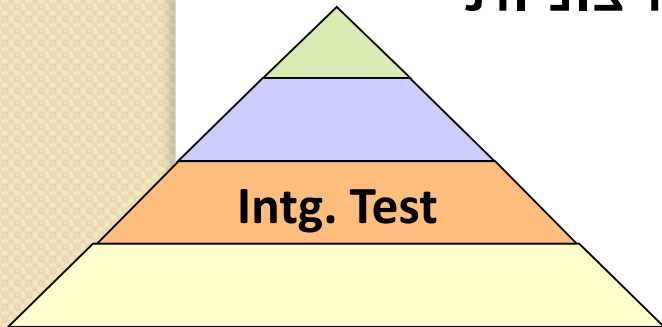
- **Component Integration** – בדיקת האינטגרציה

הפנימית, בין רכיבי המערכת

- Link Testing נקרא גם

- **System Integration** - בדיקת האינטגרציה

החיצונית, בין המערכת למערכות חיצוניות

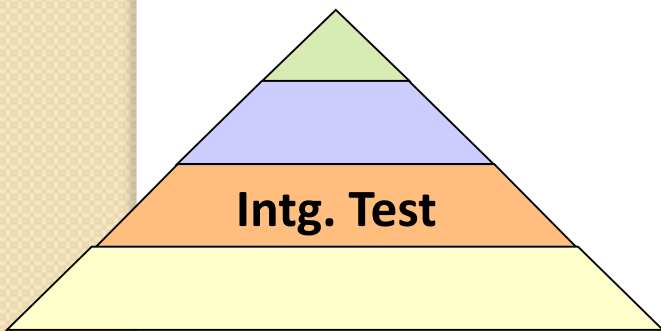


Integration testing

בדיקות אינטגרציה

- הרכיבים הנבדקים:

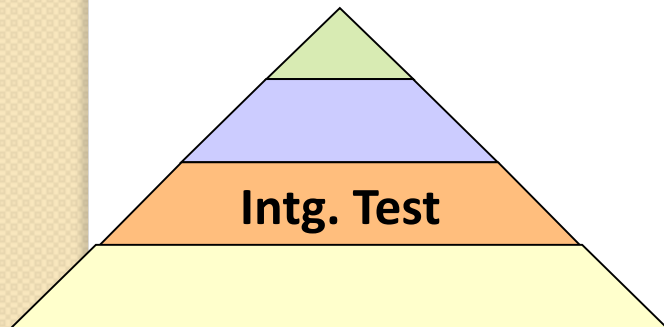
- Database implementation מודל הנתונים של כל תת מערכת
- Configuration data נתוני קונפיגורציה
- תשתית
- ממשקים (פנימיים וחיצוניים)



Integration testing

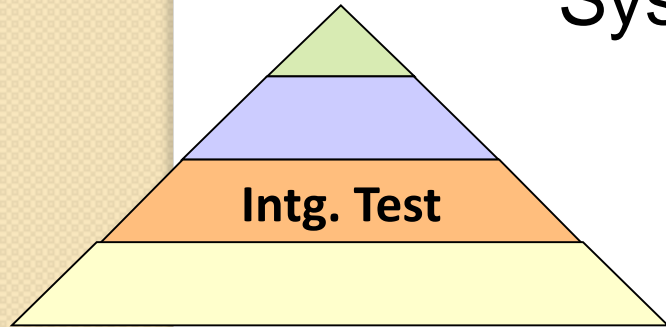
בדיקות אינטגרציה

- מתבצעות בשלבים שונים של הפרויקט:
 - Component Integration – לאחר שלב ה-Unit Test
 - System Integration – לאחר שלב ה-System Test
- ככל שמידת האינטגרציה עולה קשה יותר לשייך את התקלה לרכיב / מערכת מסוימים ורמת הסיכון עולה



Integration testing

בדיקות אינטגרציה



• Test basis:

- ממכי ה-System Detailed Design

- Architecture Design

- Workflows

- Use Cases

- באחריות המפתחים או צוות הבדיקות

- סביבת הבדיקות: סביבת בדיקות ייעודית

- בדיקות קופסא שחורה

Integration testing

בדיקות אינטגרציה

- גישות לבדיקות אינטגרציה:

- גישת ה-Big Bang – בדיקות האינטגרציה של כל המערכת לאחר שכל הממשקים מוכנים לבדיקה

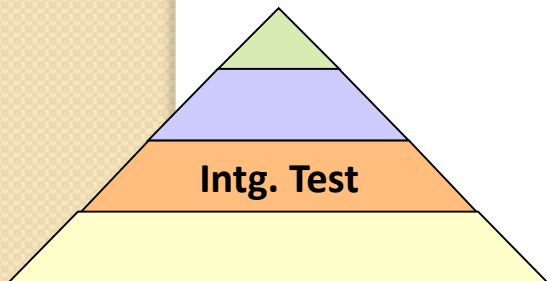
- בדיקות בשלב מאוחר של תהליך הפיתוח, לא נספיק לתקן את התקלות

- קשה לבודד כל תקלה ולמצוא את הגורם לה

- הגישה ההדרגתית Incremental – בדיקות כל ממשק ברגע שהוא מוכן לבדיקה, ללא תלות בממשקים האחרים

- בדיקות בשלב מוקדם יותר בתהליך הפיתוח

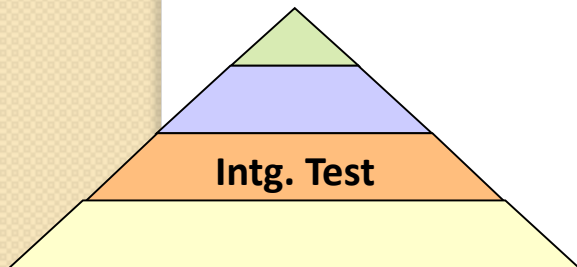
- ניתן לבודד את התקלות יותר בקלות



Integration testing

בדיקות אינטגרציה

- גישת Bottom-UP – בדיקת הממשק מהרמה הנמוכה ביותר ועד לרמה הגבוהה:
 - קישוריות בין המערכות (למשל שליחת Ping)
 - שליחת נתונים בין המערכות (למשל העברת קובץ)
 - תהליך יצירת הנתונים במערכת השולחת
 - תהליך קבלת הנתונים במערכת המקבלת
 - תזמון הממשק
- בדיקה ברמת קצה-לקצה E2E (End-to-End) – פונקציונאליות (תפקוד הממשק) ונכונות הנתונים



Integration testing

בדיקות אינטגרציה

- גישת Top-down – בדיקת הממשק ברמת קצה-לקצה E2E ורק אם יש תקלה – בדיקת הרמות הנמוכות יותר על מנת לבודד את התקלה
- סוגי בדיקות:
 - פונקציונאליות
 - לא-פונקציונאליות רלוונטיות (כגון עומסים)
- מיקוד הבדיקות באינטגרציה בין הרכיבים / מערכות ולא בפונקציונאליות של כל רכיב / מערכת

System Testing

בדיקות מערכת

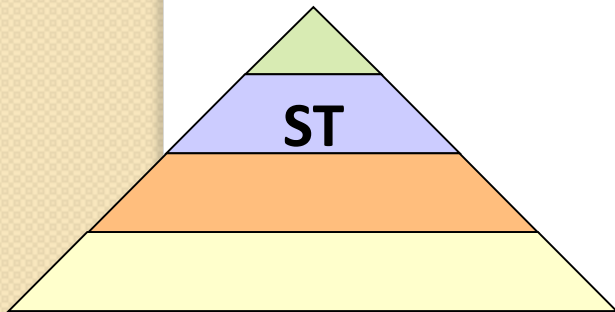
- בדיקות התנהגות המוצר / המערכת כשלם
- מטרות: לוודא עמידה בדרישות הפונקציונליות והלא-פונקציונליות של המערכת
- סביבת הבדיקות: סביבת בדיקות ייעודית המדמה את סביבת ה-Production
- הרכיבים הנבדקים:
 - המערכת
 - מדריכים למשתמש System, user and operation manuals
 - קונפיגורציה של המערכת System configuration
 - נתוני קונפיגורציה

System Testing

בדיקות מערכת

• Test Basis:

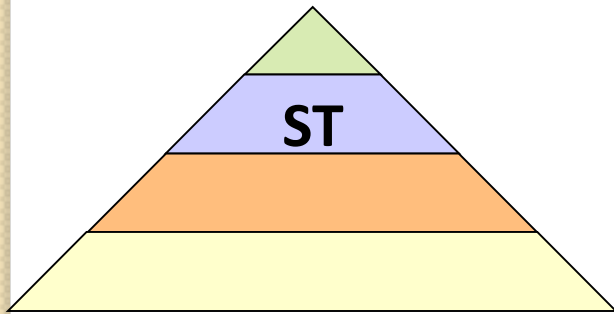
- מסמכי הדרישות System and software requirement specification
- תהליכים עסקיים Use Cases
- מסמכי איפיון Functional specification
- סיכונים Risk analysis reports



System Testing

בדיקות מערכת

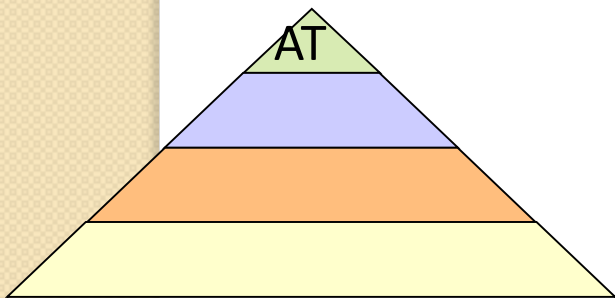
- שימוש בטכניקות בדיקות קופסא שחורה
לכתיבת תסריטי בדיקה יעילים וכיסוי
הדרישות
- ניתן להשתמש גם בטכניקות קופסא לבנה
כשרלוונטי (למשל בדיקת מבנה תפריט, ניווט
באתר אינטרנט)
- באחריות צוות בדיקות ייעודי



Acceptance Testing

בדיקות קבלה

- הערכת מוכנות המערכת לכניסה לתפעול שוטף (Production)
- מטרה: קבלת בטחון במערכת
 - האם עומדת בדרישות?
 - האם נבנתה המערכת הנכונה?
- מקרי הבדיקה נגזרים ממסמכי הדרישות
- באחריות הלקוח
- סביבת הבדיקות: סביבת בדיקות של הלקוח
- בדיקות קופסא שחורה

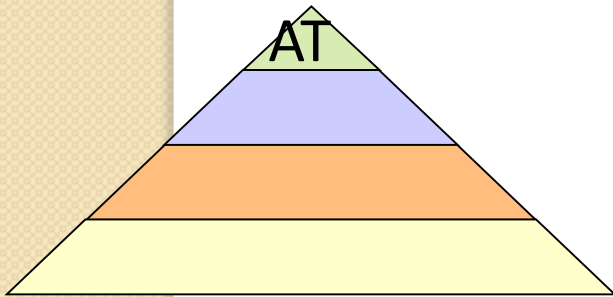


Acceptance Testing

בדיקות קבלה

• Test Basis:

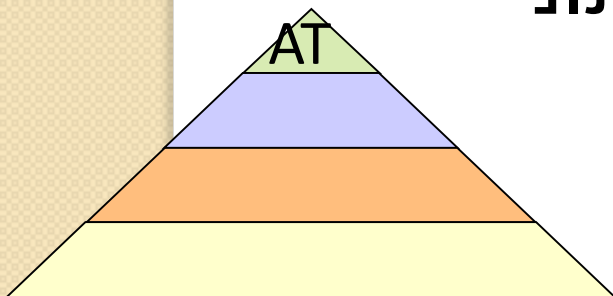
- דרישות המשתמש User requirements
- דרישות המערכת System requirements
- תהליכים עסקיים Use Cases, Business processes
- סיכונים Risk analysis reports



Acceptance Testing

בדיקות קבלה

- רכיבים נבדקים:
 - תהליכים עסקיים על המערכת עם כל רכיביה וממשקיה
 - תהליכי תפעול ותחזוקה
 - תהליכי משתמש User procedures
 - טפסים Forms
 - דוחות Reports
- הבדיקות מתבצעות תוך שימוש בנתוני קונפיגורציה אמיתיים

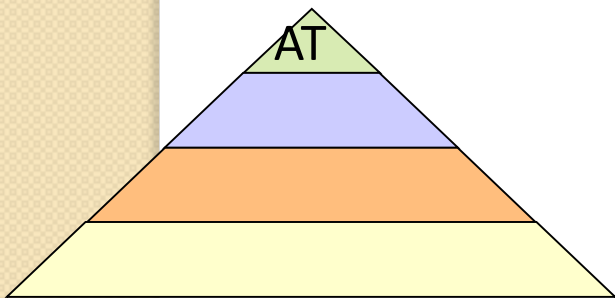


Acceptance Testing

בדיקות קבלה

- סוגי בדיקות קבלה:

- User acceptance testing (UAT) – בדיקות הנעשות ע"י משתמשי המערכת (Business users)
- Operational (acceptance) testing – בדיקות הנעשות ע"י מפעילי המערכת (System administrators):
 - בדיקת מנגנון השחזור והגיבוי (Backup & Restore)
 - התאוששות מאסון (Disaster recovery)
 - ניהול משתמשים
 - פעולות תחזוקה
 - בדיקות תקופתיות לאבטחת המידע



Acceptance Testing

בדיקות קבלה

- Contract and regulation acceptance testing

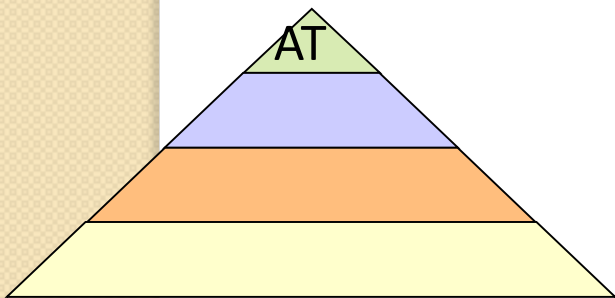
בדיקות עמידה בחוזה, עמידה בחוקים ותקנות מחייבות בתחום

- למשל Safety-Critical systems

- Alpha testing

(ספק התוכנה) ע"י צוות בדיקות חיצוני או משתמשי קצה, במטרה לקבלת משוב על הלקוחות הפוטנציאליים / קיימים בשוק לפני שחרור המוצר

- בד"כ מתבצע בפיתוח תוכנות מדף



Acceptance Testing

בדיקות קבלה

- Beta (field) testing – בדיקות קבלה הנערכות באתר הלקוח **הסופי**, ע"י משתמשי הקצה של המערכת, על מנת לקבל משוב ממשתמשי הקצה לגבי המוצר

- בעיקר בתוכנות מדף

• מושגים נוספים:

- FAT (Factory Acceptance Test) – בדיקות הנערכות ע"י הארגון המפתח, לפני מסירה ללקוח
- SAT (Site Acceptance Test) – בדיקות קבלה הנערכות באתר הלקוח, ע"י אנשי הלקוח

סוגי בדיקה

- קבוצה של פעילויות בדיקה המכוונות לבדיקת רכיב או מערכת וממוקדים במטרת בדיקות ספציפית
- סוג בדיקה כלשהוא יכול להתבצע ברמת בדיקות או שלב בדיקות אחד או יותר



סוגי בדיקה

- בדיקות פונקציונליות
- בדיקת מאפיין איכות מסוים של המערכת
(כגון שימושיות, ביצועים, אמינות, ועוד)
- בדיקת מבנה התוכנה (הקוד)
- בדיקות הקשורות לשינויים בתוכנה (כגון
(Regression



בדיקות פונקציונליות

Functional Test

- מטרתם לבדוק מה עושה המערכת, לפי המוגדר בדרישות המערכת
- תסריטי בדיקה ותוצאות צפויות נגזרים ממסמכי המערכת המתארים את הפונקציונליות של המערכת:
 - מסמכי הדרישות
 - high level design - HLD
 - Detailed design - DD
- בכל רמות הבדיקות



בדיקות פונקציונאליות

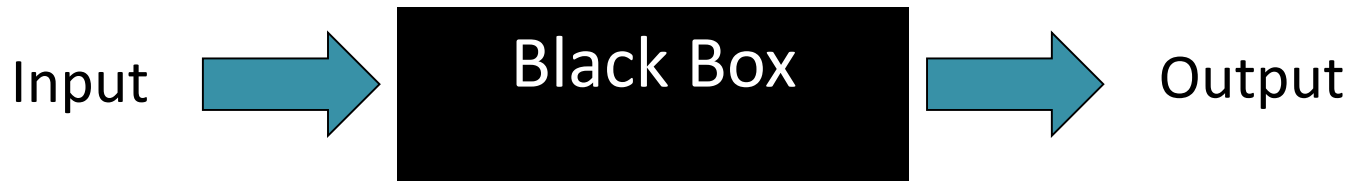
Functional Test

- מאפייני בדיקות פונקציונאליות (על פי ISO 9126):
 - התאמה לדרישות ודיוק
 - בדיקות ממשקים Interoperability Testing -
בדיקת יכולת המערכת / רכיב לעבוד עם רכיב /
מערכת אחרת אחת או יותר
 - בדיקות Security – בדיקת יכולת המערכת למנוע
גישה שאינה מורשית למידע ולקוד
- בדיקות קופסא שחורה
 - שימוש בטכניקות קופסא שחורה לכתובת תסריטי
הבדיקות

בדיקות קופסא שחורה

Black Box Testing

- בדיקת הרכיב / מערכת כקופסא שחורה:
המבנה הפנימי של המערכת / הקוד אינו ידוע
/ רלוונטי
- תסריטי הבדיקות נגזרים מהמסמכים
המגדירים את דרישות המערכת



בדיקות לא-פונקציונאליות

Non-Functional Test (NFT)

- מטרתם לבדוק כמה טוב המערכת עובדת
- ניתן לבדוק פרמטרים מדידים בלבד
- ישנם סוגים רבים של בדיקות לא-פונקציונאליות, לדוגמא:

- Efficiency
- Usability
- Portability

- Maintainability
- Reliability

- סוגים שונים יכולים לעודד להיבדק ברמות בדיקה שונות

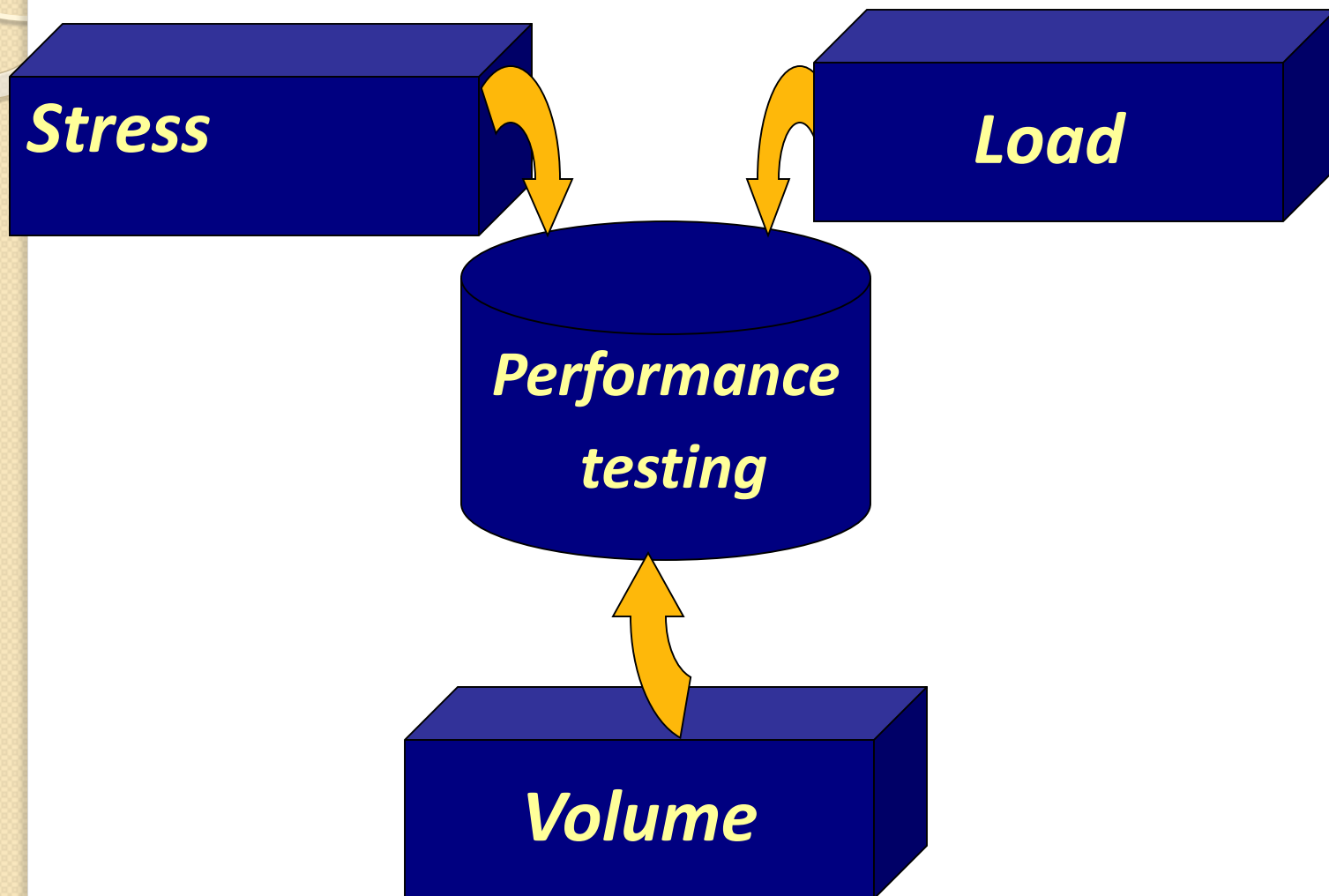
Efficiency Testing

בדיקות ביצועים

- בדיקת המערכת מתוך מטרה לוודא כי היא מתפקדת כנדרש במצבי עומס שכיחים וקיצוניים
- ישנם כמה מישורים לבדיקת ביצועים:
 - זמן תגובה Response Time – מדידת זמן התגובה של המערכת / הרכיב מבחינת מסכים, הרצת תהליכים
 - כמות נתונים – מדידת יכולת וביצועי המערכת כאשר בסיס הנתונים שלה עמוס בנתונים
 - צריכת משאבי מחשב
- על פי ISO 9126



סוגי בדיקות ביצועים





בדיקות ביצועים

Load Testing

- בדיקות עומסים - אימות יכולת התגובה של צד השרת במערכות שרת/לקוח בהן צפויים משתמשים רבים בו זמנית
- בדיקות אלו מתמקדות במדידת זמני התגובה של המערכת בזמן עומס
- בודקות:
 - זמני תגובה כאשר מספר "משתמשים וירטואלים" הינו כצפוי בשעת העומס
 - זמני תגובה כאשר מספר הטרנזקציות ופעילויות המתרחשות במערכת הינו כצפוי בשעת העומס

בדיקות ביצועים

Stress Testing



- בדיקת זמני התגובה של המערכת בעומס רב מהצפוי, מציאת גבולות המערכת (מהו צפי הגידול של המערכת עד אליו היא תעבוד בצורה תקינה)
- בדיקת התמודדות המערכת עם עומס רב מהצפוי לאורך זמן:
 - יציבות המערכת - האם היא קורסת ובאילו מצבים (דליפות זיכרון, File System Full)
 - התאוששות המערכת לאחר קריסה (אם קרסה)
 - זמינות המערכת וזמני תגובה
- מספר משתמשים בו זמנית – רב מהצפוי (עולה בהדרגה)
- מספר טרנזאקציות הצפוי בשעת העומס או רב יותר

בדיקות ביצועים

Volume Testing

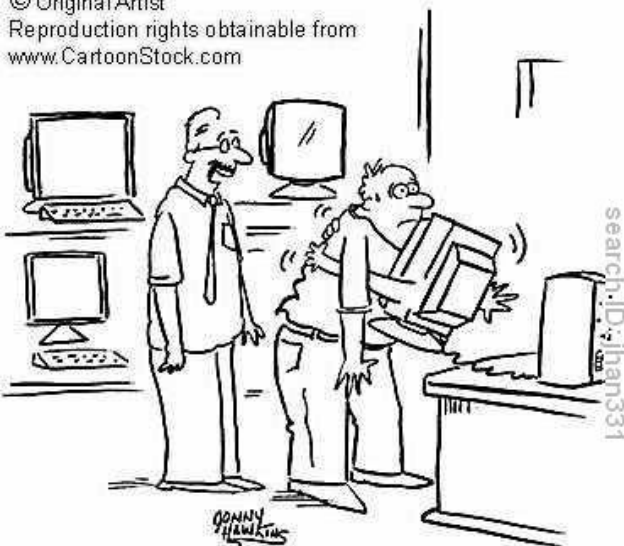
- בדיקת זמני תגובת המערכת כאשר בסיס הנתונים מלא (80% ומעלה)
- מספר משתמשים בו-זמנית – לפי הצפוי בשעת העומס
- מספר טרנזאקציות הצפוי בשעת העומס



בדיקות שימושיות Usability Testing

- בדיקת מידת השימושיות של המערכת מבחינת היותה ברורה, מובנת, ניתנת ללמידה, שימושית וחברותית למשתמש
 - שימוש בסטנדרטים של GUI
 - בדיקות תפעול Operability Testing

© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com



עפ"י הגדרה של ISO 9126

"It's a very user-friendly model."

בדיקות תאימות

Portability Testing

- בדיקות מידת התאימות של המערכת לסביבות חומרה ותוכנה שונות:

- מערכות הפעלה שונות (Win 2000, Win XP, Win (2007, Vista, Unix
- Databases שונים (Access ,SQL Server ,Oracle)
- Web Browser שונים (IE ,Firefox ,Chrome , ועוד)
- מחשבים שונים (Mainframe, PC, Mac , וכו')



עפ"י הגדרה של ISO 9126

בדיקות תחזוקתיות

Maintainability Testing

- בדיקת מידת הקלות בה ניתן לתחזק את התוכנה: לתקן תקלות, להכניס שינויים בעקבות דרישות חדשות, להתאימה לשינויים בסביבה ולבדקה



- הערות בקוד
- קוד הכתוב לפי סטנדרטים
- קוד הכתוב בצורה ברורה ושאינה מורכבת
- האם ניתנת לבדיקה?

עפ"י הגדרה של ISO 9126

בדיקות לא-פונקציונליות

Non-Functional Tests

- **בדיקת אמינות Reliability Testing**
בדיקות אמינות המערכת בביצוע הפעילויות העסקיות לפי הדרישות לאורך זמן מוגדר
- **בדיקות התאוששות Recoverability Testing**
– בדיקת יכולת המערכת להתאושש מאסון מבחינת זמן הגעה לתפקוד מלא ושחזור מידע לנפגע/אבד כתוצאה מהאסון

עפ"י הגדרה של ISO 9126



רמות בדיקה וסוגי בדיקה

סוגי בדיקות שונים נדרשים ברמות בדיקה שונות

Test Type/Test Level	Comp. Test	Intg Test	ST	AT
Functional	✓	✓	✓	✓
Performance	✓	✓	✓	✓
Reliability			✓	✓
Backup & Recovery				✓
Security	✓	✓	✓	✓
Usability	✓	✓	✓	✓
Documentation			✓	✓
Interoperability/Interface Testing		✓	✓	✓
Portability		✓	✓	✓
Maintainability	✓	✓	✓	✓

בדיקות מבניות

Structural Testing

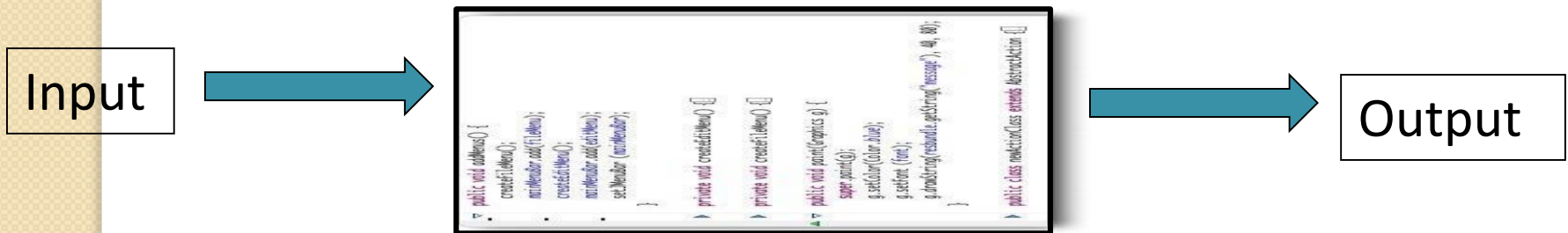
- בדיקות המבוססות על מבנה הקוד (White Box)
- בעיקר ב-Unit Test, אך ניתן להשתמש גם ברמות אחרות
 - למשל לבדיקת תפריטים ומודלים עסקיים
- Code Coverage - מדידת הכיסוי של הקוד ע"י הבדיקות

- ישנן טכניקות שונות למדידת הכיסוי
- קיימים כלים תומכים

```
1 _root.onEnterFrame=function(){
2   for(i=1;i<6;i+=1){
3     _root["item"+i].onMouseMove=function(){
4       d = Math.sqrt(Math.abs(_xmouse-this._x)*
5         Math.abs(_xmouse-this._x)+
6         Math.abs(_ymouse-this._y)*
7         Math.abs(_ymouse-this._y));
8       dx = 70;
9       if(d < 70){
10        this._xscale = 100+(dx-d);
11        this._yscale = 100+(dx-d);
12      }else{
13        this._xscale = 100;
14        this._yscale = 100;
15      }
16    }
17  }
18 }
```

בדיקות קופסא לבנה White Box Testing

- בדיקת רכיב המערכת בהתבסס על המבנה הפנימי של הרכיב (מבנה הקוד)
- הבדק צריך להכיר שפות קוד ומבנה הרכיב צריך להיות גלוי לבודק
- תסריטי הבדיקות נגזרים מהמבנה הפנימי של הרכיב או ממסמכי ה-Detailed Design
- נעשה ע"י המפתח



בדיקות הקשורות לשינויים

- בדיקות המבוצעות לאחר שבוצעו שינויים בתוכנה, כגון:

- תיקון תקלות

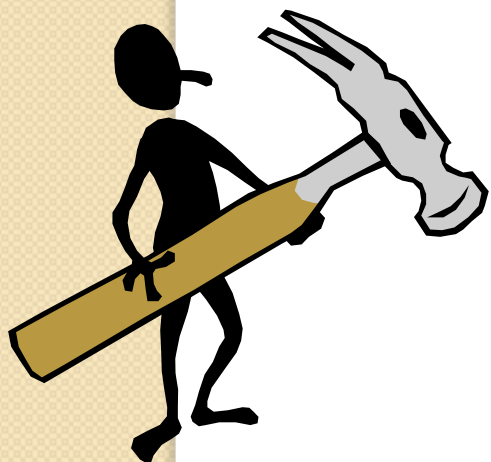
- פיתוח פונקציונאליות חדשה

- שינויים לפונקציונאליות קיימת

- שינויים בסביבה

- Confirmation Testing בדיקות אישור –

בדיקת תיקוני תקלות על מנת לוודא שהתיקון פתר את התקלה בצורה מלאה



בדיקות הקשורות לשינויים

- Regressions Testing בדיקות רגרסיה –
בדיקות חוזרות של אזורים שנבדקו כבר
בהצלחה בעבר, על מנת לוודא שלא נכנסו
תקלות חדשות בעקבות שינויים שנעשו
במוצר

- כמות הבדיקות שנריץ מבוססת של הסיכון של אי
מציאת תקלות בתוכנה שעבדה קודם
- בכל הרמות ועבור כל סוגי הבדיקה



בדיקות הקשורות לשינויים

- שימוש חוזר (Reuse) של תסריטי הבדיקות - ב- Regression Testing ו- Confirmation Testing
ישנה הרצה חוזרת של תסריטי בדיקות שהורצו בעבר:

- לשם בדיקת תיקון תקלה – נריץ את התסריט שגילה אותה
- לשם בדיקות רגרסיה – נריץ תסריטי בדיקות שהורצו בעבר ועברו בהצלחה



בדיקות תחזוקה

Maintenance Testing

- לאחר התקנת הגרסה / התוכנה ב-Production – היא יכולה לעבוד ב"עולם האמיתי" במשך שנים רבות
- במשך עבודתה ב-Production ישנם שינויים שונים שצריכים להיעשות על התוכנה ו/או סביבה:
 - שינויי תוכנה כגון תיקוני תקלות, פיתוח דרישות חדשות (New functionality), התאמות לתוכנה צד ג'
 - שינויים לסביבה כגון upgrade לחומרה חדשה יותר
 - הסבת נתונים ממערכת קודמת למערכת החדשה

בדיקות תחזוקה Maintenance Testing

- בדיקות תחזוקה נדרשות כאשר ישנם שינויים כלשהם במערכת הנמצאת בשלב

Production

- בדיקות תחזוקה מתבצעות על סביבת בדיקות זהה לסביבת ה-Production (Production-Like Environment)



סוגי שינויים למערכת ב-Production



- Modifications – התאמות:
 - גרסה חדשה (מתוכננת מראש):
 - פיתוח פונקציונאליות חדשה
 - תיקוני תקלות (רגילות ודחופות)
 - שינויים לסביבה:
- כגון שדרוג מערכת ההפעלה או גרסת ה-DB
- תיקונים דחופים לפרצות אבטחה

סוגי שינויים

למערכת ב-Production

- Migration – העברת המערכת מפלטפורמה אחת לאחרת:

- שינויי תוכנה לתמיכה בפלטפורמה החדשה (כגון מערכת הפעלה אחרת, מחשבים שונים)
- Operational Testing (בדיקות תפעוליות) לבדיקת תפקוד המערכת על הפלטפורמה החדשה



סוגי שינויים

למערכת ב-Production

• Retirement – פרישה – החלפת המערכת
הישנה במערכת חדשה:

◦ בדיקות הסבות נתונים - data migration
(מהמערכת הישנה לחדשה)

◦ בדיקות תהליך אחסון הנתונים archiving –
במידה וצריך לשמור על הנתונים לפרק זמן ארוך



בדיקות תחזוקה

- בנוסף לבדיקת השינויים, בדיקות תחזוקה כוללות בדיקות רגרסיה נרחבות על חלקי מערכת שלא עברו שינוי
- Breadth Test – בדיקת כל הפונקציונליות, אך לא בצורה מעמיקה
- Depth Test – בדיקת פונקציונליות מסוימת בצורה מעמיקה



בדיקות תחזוקה

- היקף בדיקות התחזוקה והרגרסיה נקבע על סמך הסיכון של השינויים, גודל המערכת הקיימת והיקף השינויים
- בהתאם לאופי השינויים שנעשו - יקבעו אילו רמות וסוגי בדיקות יבוצעו



בדיקות תחזוקה

- Impact Analysis – מסמך המפרט כיצד משפיעים השינויים על המערכת הקיימת ועל הפונקציונאליות הקיימת
- מסמך זה עוזר בקביעת כמה בדיקות רגרסיה נדרשות, ואילו אזורים במערכת כדאי לבדוק (כדי לוודא שלא נפגעו מהשינויים)



בדיקות תחזוקה

- בעיה נפוצה בבדיקות תחזוקה – מחסור במסמכי מערכת או שהמסמכים הקיימים אינם מעודכנים (המתארים את המערכת)
- Test Oracle – כאשר חסרים מסמכי מערכת מעודכנים – נחפש מקורות אחרים למידע (לצורך כתיבת התוצאות הצפויות בתסריטי הבדיקות שאנו כותבים):
 - המערכת הקודמת
 - מסמך מדריך למשתמש User manual
 - משתמש מומחה המכיר טוב את המערכת
 - בכל מקרה – לא הקוד של התוכנה אותה אנו בודקים



תרגיל

- תרגיל 2 בחוברת תרגילים: רמות וסוגי בדיקה



1. Which of the following is not a quality characteristic listed in ISO 9126 Standard?

- a) Functionality
- b) Usability
- c) Supportability
- d) Maintainability



2. What test can be conducted for off-the-shelf software to get market feedback?

- A. *Beta testing*
- B. *Usability testing*
- C. *Alpha testing*
- D. *COTS testing*

3. Which of the following statements is MOST OFTEN true?

- a) Source-code inspections are often used in component testing.
- b) Component testing searches for defects in programs that are separately testable.
- c) Component testing is an important part of user acceptance testing.
- d) Component testing aims to expose problems in the interactions between software and hardware components.

4. Which of the following defines the scope of maintenance testing?

- A. The coverage of the current regression pack.
- B. The size and risk of any change(s) to the system.
- C. The time since the last change was made to the system.
- D. Defects found at the last regression test run.


5. Which of following statements is true? Select ALL correct options Regression testing should be performed:

- i once a month*
- ii when a defect has been fixed*
- iii when the test environment has changed*
- iv when the software has changed*

- A ii and iv.
- B ii, iii and iv.
- C i, ii and iii.
- D i and iii.

6. Which of the following uses Impact Analysis most?

- a) Component testing
- b) Non-functional system testing
- c) User acceptance testing
- d) Maintenance testing



7. Repeated Testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software component:

- a) Re Testing .
- b) Confirmation Testing
- c) Regression Testing
- d) Negative Testing

8. Impact Analysis helps to decide :-

- a) How much regression testing should be done.
- b) Exit Criteria
- c) How many more test cases need to written.
- d) Different Tools to perform Regression Testing

9. Functional system testing is:

- a) testing that the system functions with other systems
- b) testing that the components that comprise the system function together
- c) testing the end to end functionality of the system as a whole
- d) testing the system performs functions within specified response times

10. The selection of test cases for regression testing (Testing artifacts)

- a) Requires knowledge on the bug fixes and how it affect the system
- b) Includes the area of frequent defects
- c) Includes the area which has undergone many/recent code changes
- d) All of the above

11. *A type of integration testing in which software elements, hardware elements, or both are combined all at once into a component or an overall system, rather than in stages.*

- A. System Testing
- B. Big-Bang Testing
- C. Integration Testing
- D. Unit Testing

12. *Big bang approach is related to*

- A. Regression testing
- B. Inter system testing
- C. Re-testing
- D. Integration testing

13. A test harness is a:

- a) A high level document describing the principles, approach and major objectives of the organization regarding testing*
- b) A distance set of test activities collected into a manageable phase of a project*
- c) A test environment comprised of stubs and drives needed to conduct a test*
- d) A set of several test cases for a component or system under test*

14. System test can begin when?

- i. The test team completes a three day smoke test and reports on the results to the system test phase entry meeting*
- ii. The development team provides software to the test team 3 business days prior to starting of the system testing*
- iii. All components are under formal, automated configuration and release management control*

- A. I and II only*
- B. II and III only*
- C. I and III only*
- D. I, II and III*

תשובות:

שאלה	A	B	C	D
1			✓	
2	✓			
3		✓		
4		✓		
5		✓		
6				✓
7			✓	
8	✓			
9			✓	
10				✓
11		✓		
12				✓
13			✓	
14		✓		

תהליך הבדיקות

129

130

132

134

140

142

• תהליך הבדיקות

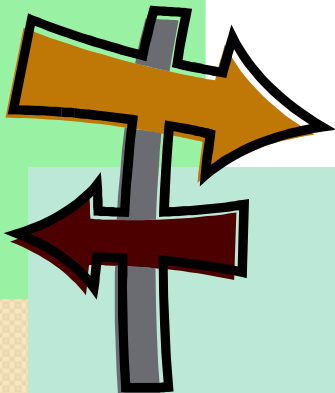
Planning & Control ◦

Analysis & Design ◦

Implementation & execution ◦

Evaluation exit criteria and reporting ◦

Closure ◦



תהליך הבדיקות הבסיסי

Testing

Planning & Control

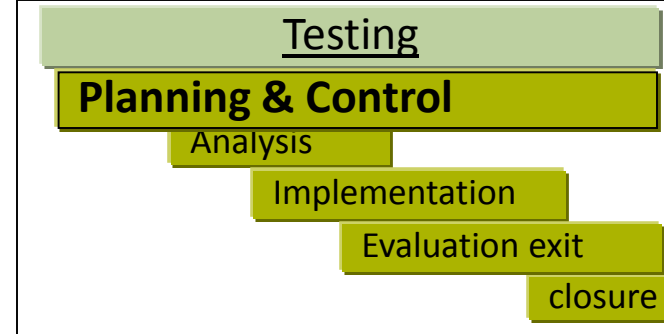
Analysis & Design

Implementation & execution

Evaluation exit criteria and reporting

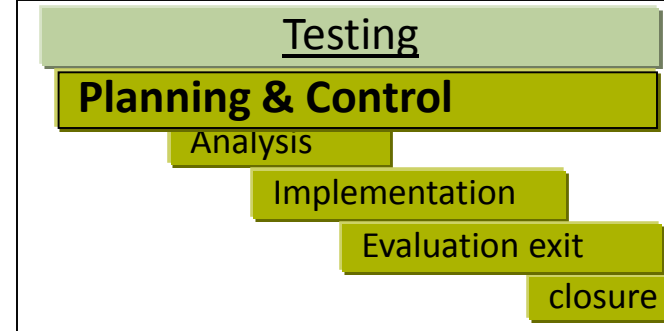
Closure

Planning and control



- קביעת מטרות הבדיקות והפעולות להשגתן
- קביעת תכולת הבדיקות והסיכונים
- זיהוי גבולות המערכת הנבדקת
- קביעת גישת הבדיקות, מדיניות ואסטרטגית הבדיקות
- קביעת משאבים: כ"א, סביבות, כלים
- קביעת לוחות זמנים לפעילויות הבדיקות
- קביעת קריטריונים לכניסה ויציאה עבור רמות הבדיקה
- תוצר: מסמך STP - SW Test Plan
- באחריות מנהל הבדיקות

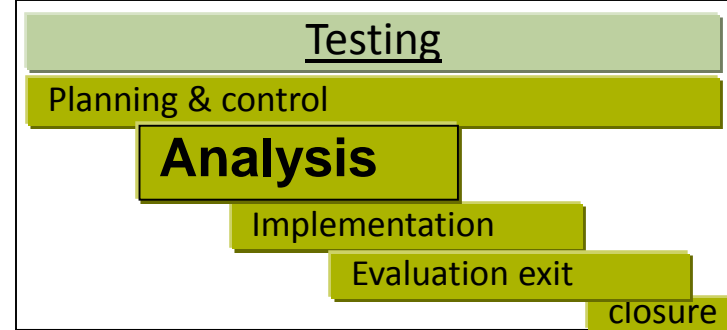
Planning and control



- השוואת ההתקדמות במשימות השונות בפועל לעומת התכנון
- פעילות מתמשכת לאורך כל הפרויקט
- דיווח הסטטוס וסטיות מהתוכנית למנהלים
- נקיטת פעולות הכרחיות להשגת מטרות הפרויקט
- עדכון התכנון בהתאם לתוצאות הבקרה
- שימוש במדדים, דוחות
- באחריות מנהל הבדיקות



Analysis & Design

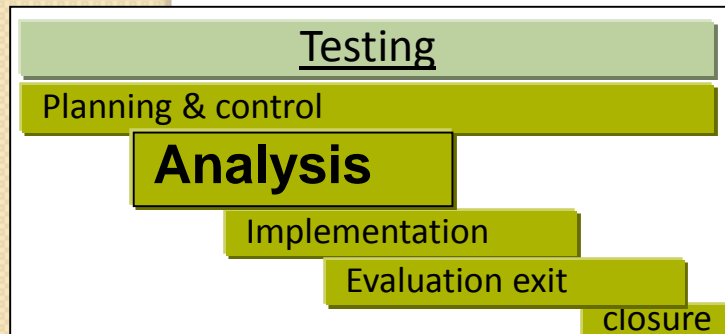


- קריאה וניתוח מסמכי ה-Test basis:
 - Testability - האם הדרישות השונות ניתנות לבדיקה?
- תרגום מסמכי ה-Test basis לחוקי בדיקה Test
 - Rules / Test Conditions
- **Test Condition** = פריט או מאורע ברכיב או מערכת, שניתן לוודא אותו באמצעות תסריט בדיקה אחד או יותר
 - למשל פונקציה, טרנזקציה, שדה במסך

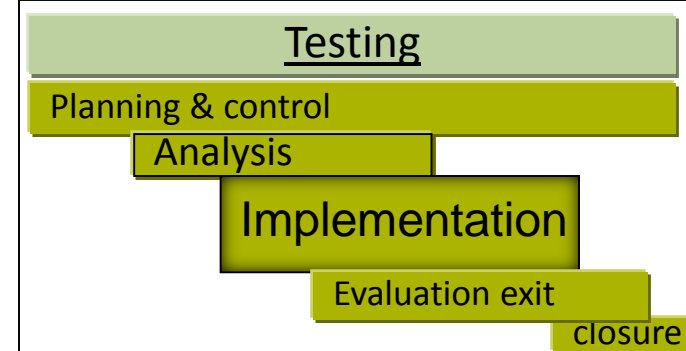
Analysis & Design

ניתוח ועיצוב

- תעדוף חוקי הבדיקה
- פירוק פונקציונאלי של חוקי הבדיקה לתסריטי בדיקה (Test Cases) – ברמה גבוהה
- זיהוי נתוני בדיקה הכרחיים לוידוא חוקי הבדיקה
- תכנון סביבת הבדיקות, תשתית וכלים הנדרשים להרצת הבדיקות
- Traceability – שמירת עקיבות דו-צדדית בין ה-Test Test Conditions ל-Basis

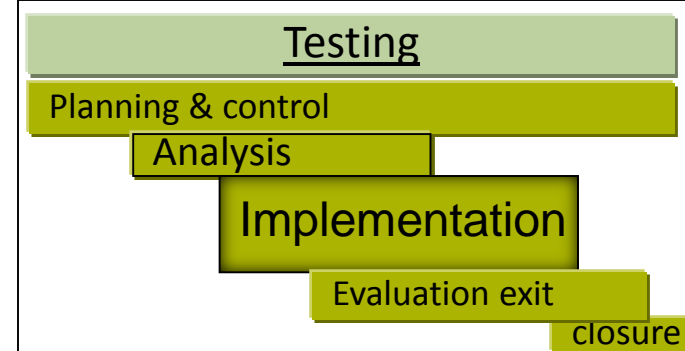


Implementation & execution



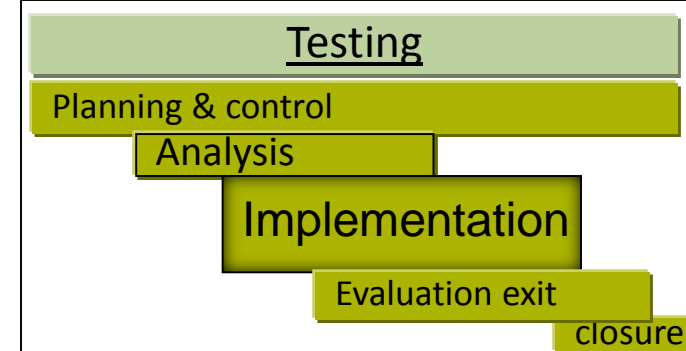
- כתיבת תסריטי הבדיקה – Test Cases כולל צעדים (Test Procedure)
- Test Case = סט של ערכי קלט, תנאי קדם, תוצאות צפויות ותנאים לאחר הרצה, הבאים לוודא חוק בדיקה מסוים
- Test Procedure = הפעולות והצעדים אותם צריך לבצע על מנת להריץ תסריט בדיקה מסוים
- תעודף תסריטי הבדיקות
- Traceability – שמירת עקיבות דו-צדדית בין ה-Test Basis ל-Test Cases

Implementation & execution



- הכנת נתוני הבדיקות Test Data
- הכנת תסריטי בדיקות אוטומטיות – אופציונאלי
- יצירת מנות הרצה Test Suites
- Test Suite = אוסף של תסריטי בדיקה המסודרים לפי סדר הרצה מסוים
 - מצב המערכת בסוף תסריט אחד משמש כתנאי קדם להרצת התסריט הבא
- בדיקת מוכנות סביבת הבדיקות
 - למשל ע"י ריצת בדיקת Sanity

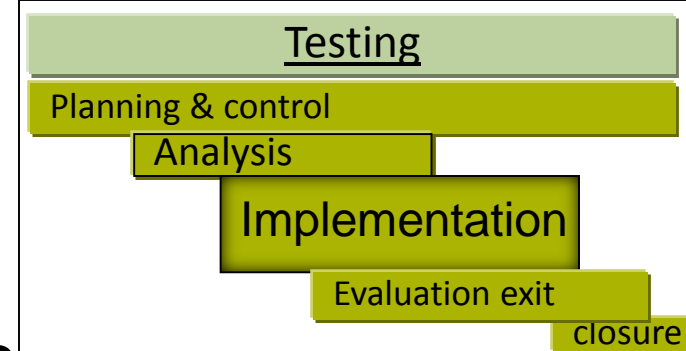
Implementation & execution



• בדיקת Sanity:

- מנת הרצה (Test Suite) הכוללת תסריטי בדיקות (Test Cases) הבודקים ברמה בסיסית את הפונקציונאליות העיקרית והחשובה ביותר (תהליכים עסקיים) של המערכת
- המטרה: לוודא תקינות ה-build (הגרסה) ברמה בסיסית לפני תחילת הרצת הבדיקות המעמיקות
- הרצת בדיקת Sanity מתבצעת על כל Build שמגיע לבדיקות
- משך ההרצה: מספר שעות

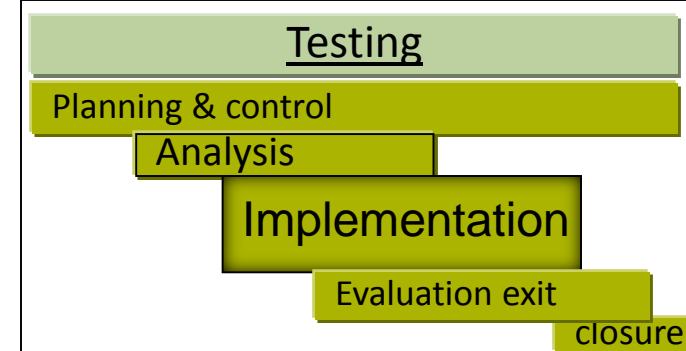
Implementation & execution



לפני תחילת הדקות יש לוודא מוכנות להרצה:

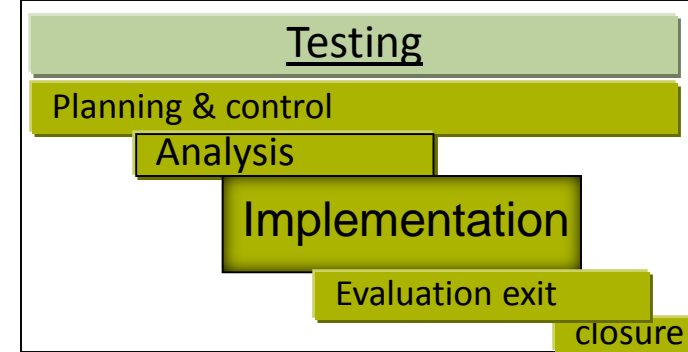
- האם כל הדרישות כוסו ע"י תסריטי בדיקה?
- האם כל תסריטי הבדיקה אושרו כטובים?
- האם הנתונים לבדיקות Test Data – מוכנים?
- האם הסביבות מוכנות?
- האם התוכנה מוכנה לבדיקות (ה-build)?
- האם ישנה תמיכת אנשי פיתוח לכל תקופת הבדיקות?
- האם ישנה תמיכת אנשי System Administrators לכל תקופת הבדיקות?
- האם ישנו נוהל וכלי לפתיחת תקלות וטיפול בהן?
- האם קיימים רישיונות לתוכנות צד ג' לכל תקופת הבדיקות?

Implementation & execution



- הרצת מנות ההרצה ותסריטי הבדיקה
- הרצת הבדיקות האוטומטיות – אופציונאלי
- תיעוד תוצאות ההרצה (נכשל/עבר/לא הושלם/לא רץ)
- השוואת תוצאות בפועל לתוצאות הצפויות
- ניתוח ודיווח התקלות שנמצאו

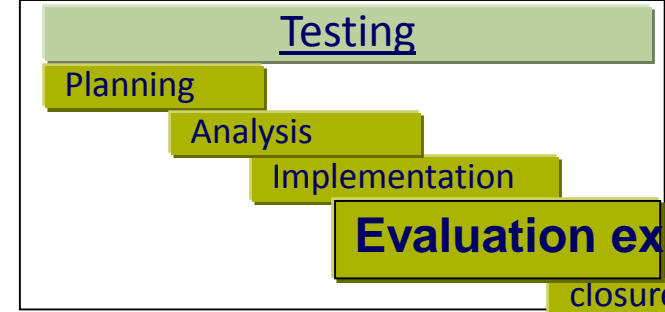
Implementation & execution



- Re-Testing – ווידוא תיקוני תקלות
- Regression Testing – הרצת בדיקות רגרסיה



Evaluating exit criteria & reporting



- הערכת הגעה לקריטריון היציאה מרמת הבדיקות
- ההערכה נעשית ע"י ניתוח תוצאות ההרצה לעומת קריטריון היציאה לפי המוגדרים ב-STP
- הערכה האם צריך להמשיך להריץ או לעדכן את קריטריון היציאה
- נעשה עבור כל רמת בדיקות
- כתיבת דוח סיכום הבדיקות STR לאחר הגעה לקריטריון היציאה



דוח סיכום הבדיקות

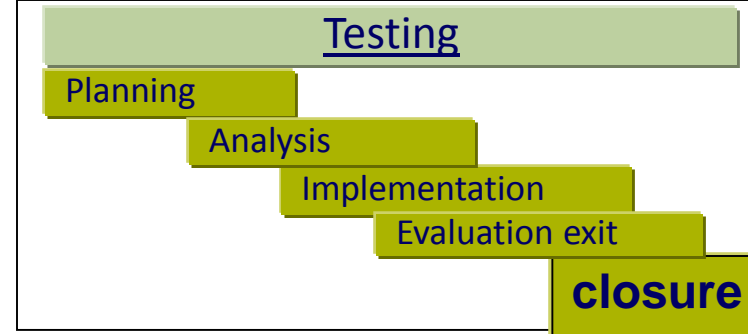
Test Summary Report - STR

- מטרת הדוח - לסכם את כל פרויקט הבדיקות עבור הגרסה
- ניתוח מידע ומדדים והמלצה על פעילויות עתידיות, כגון:
 - סיכום פעילויות הבדיקות שנעשו
 - הבדלים בין התכנון המקורי לביצוע בפועל של הבדיקות
 - הערכת התקלות הפתוחות
 - סיכונים בולטים
 - מידת הביטחון במוצר שנבדק
- באחריות מנהל הבדיקות

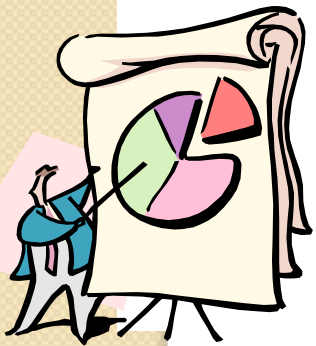


Test Closure

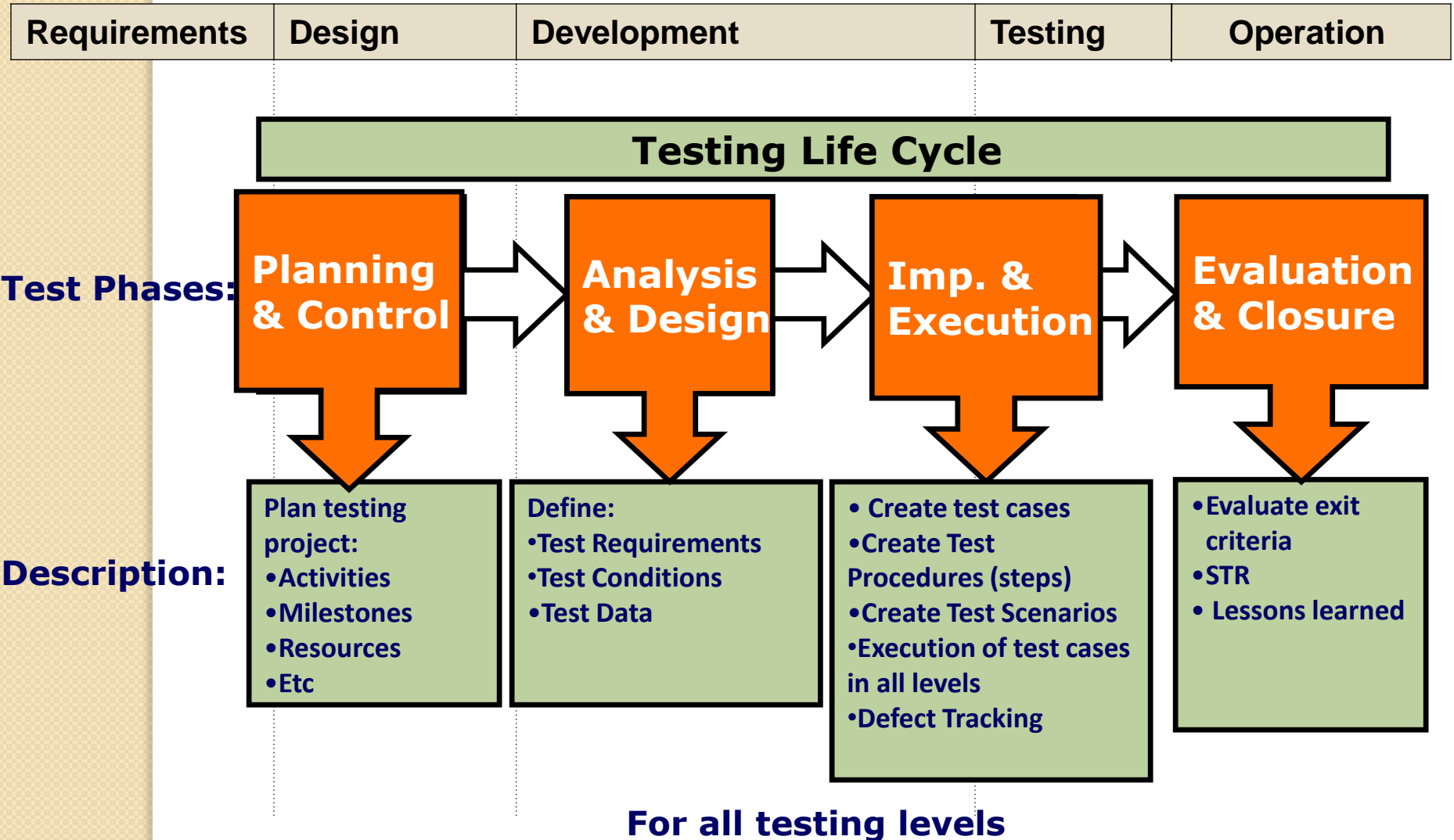
פעילויות סיכום



- בדיקה האם נמסרו כל תוצרי הבדיקות, לפי ההתחייבויות וה-STP
- וידוא סגירת כל התקלות וטיפול בתקלות הפתוחות – ע"י תיקון או העברתם לגרסה הבאה
- תיקון כנדרש ואחסון כל ה-Testware: חוקי הבדיקה, תסריטי הבדיקה, נתונים, סביבת הבדיקות – לשימוש עתידי
- העברת ה-Testware לצוות בדיקות תחזוקה
- ניתוח והסקת מסקנות מפרויקט הבדיקות שהסתיים לשיפור תהליך הבדיקות בגרסאות עתידיות



תהליך הבדיקות - סיכום



פעילויות הפיתוח והבדיקות

סיכום

