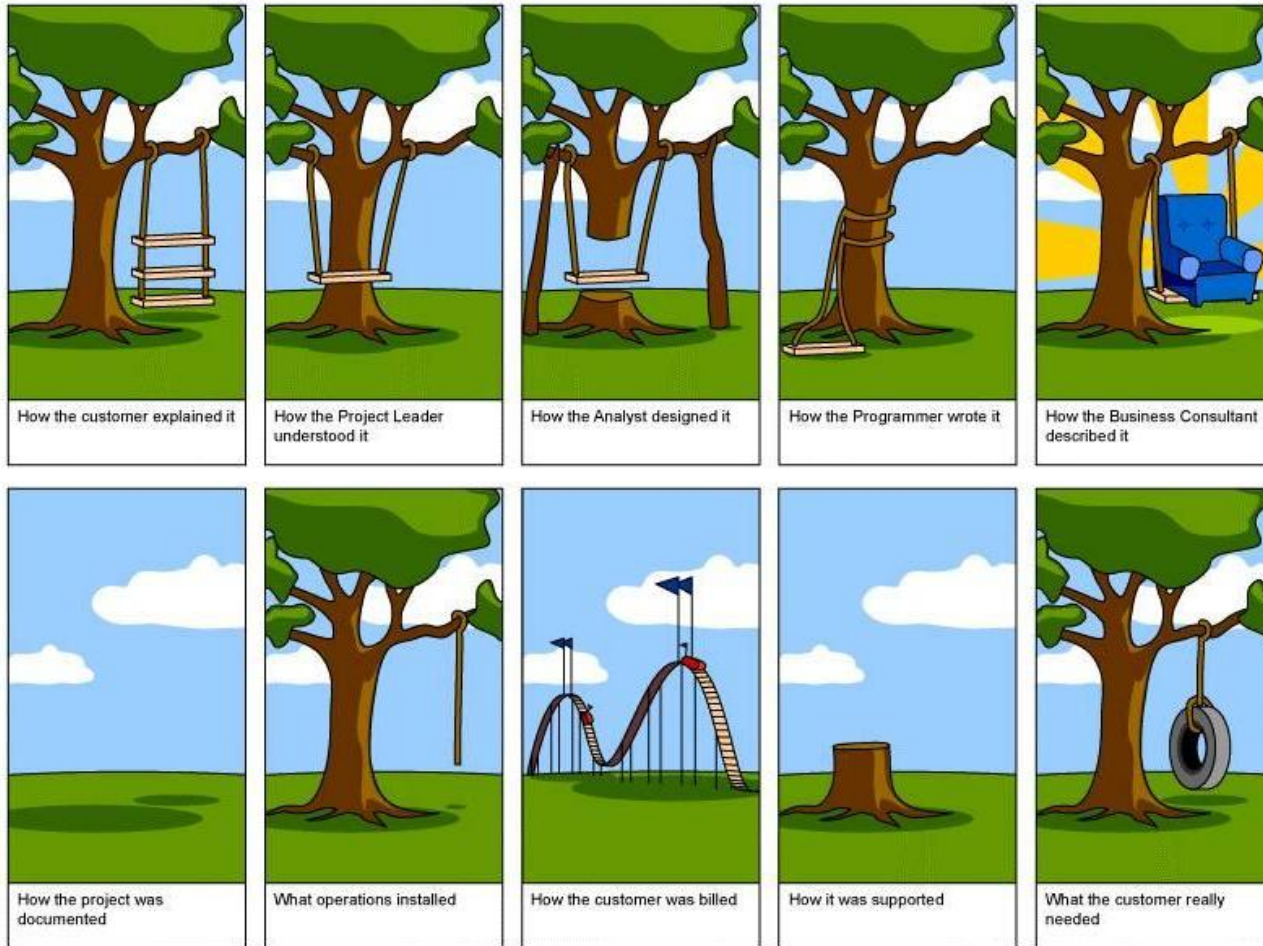




# Software Engineering + Programming tips

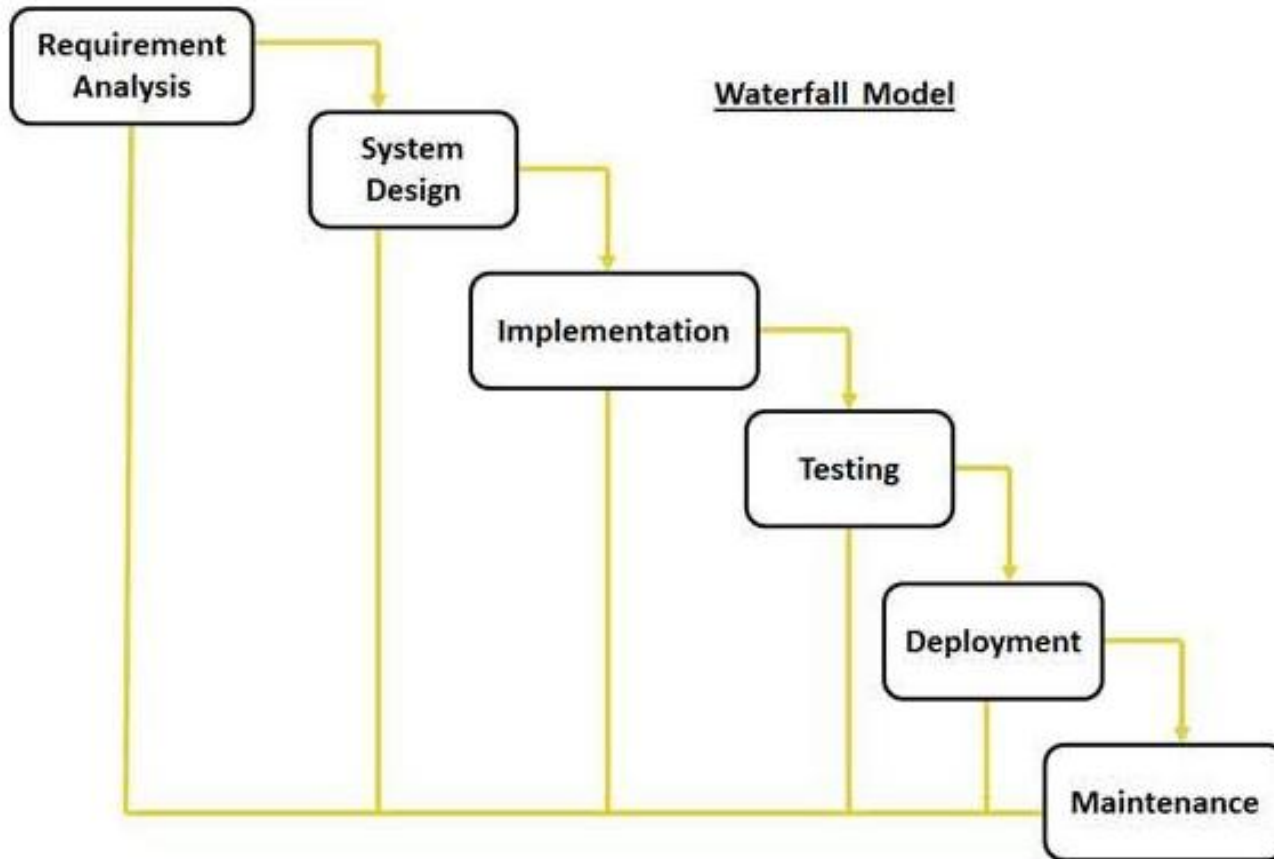
Lena Dankin

# למה צריך "הנדסת תוכנה"?



# מודלים בהנדסת תוכנה - waterfall

---



# מודלים בהנדסת תוכנה - waterfall

---

## ▶ יתרונות:

- ▶ משימות מוגדרות היטב בכל שלב.
- ▶ תיעוד מלא של כל שלבי הפרוייקט.
- ▶ קל לניהול, בפרט לחלוקת משימות.

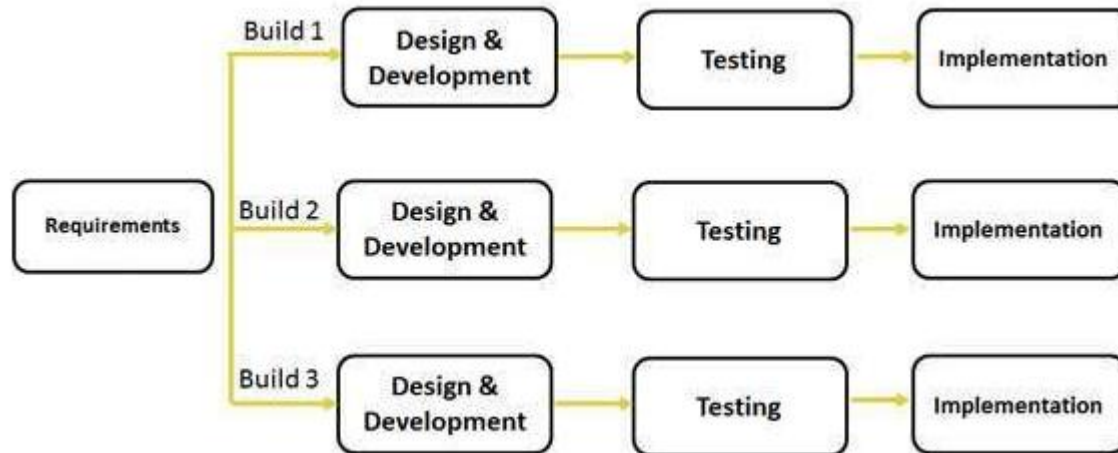
## ▶ חסרונות:

- ▶ לא פרקטי בפרוייקטים גדולים:
- ▶ החלק התכנותי מגיע בשלב מאוחר יחסית בחיי הפרוייקט.
- ▶ חוסר גמישות לשינויים בדרישות/עיצוב הטכני.
- ▶ סיכון גבוה כאשר הדרישות אינן מוגדרות היטב.

▶ מודל זה מיושם לעיתים נדירות בלבד

# מודלים בהנדסת תוכנה - iterative

---



# מודלים בהנדסת תוכנה - iterative

---

## ▶ יתרונות:

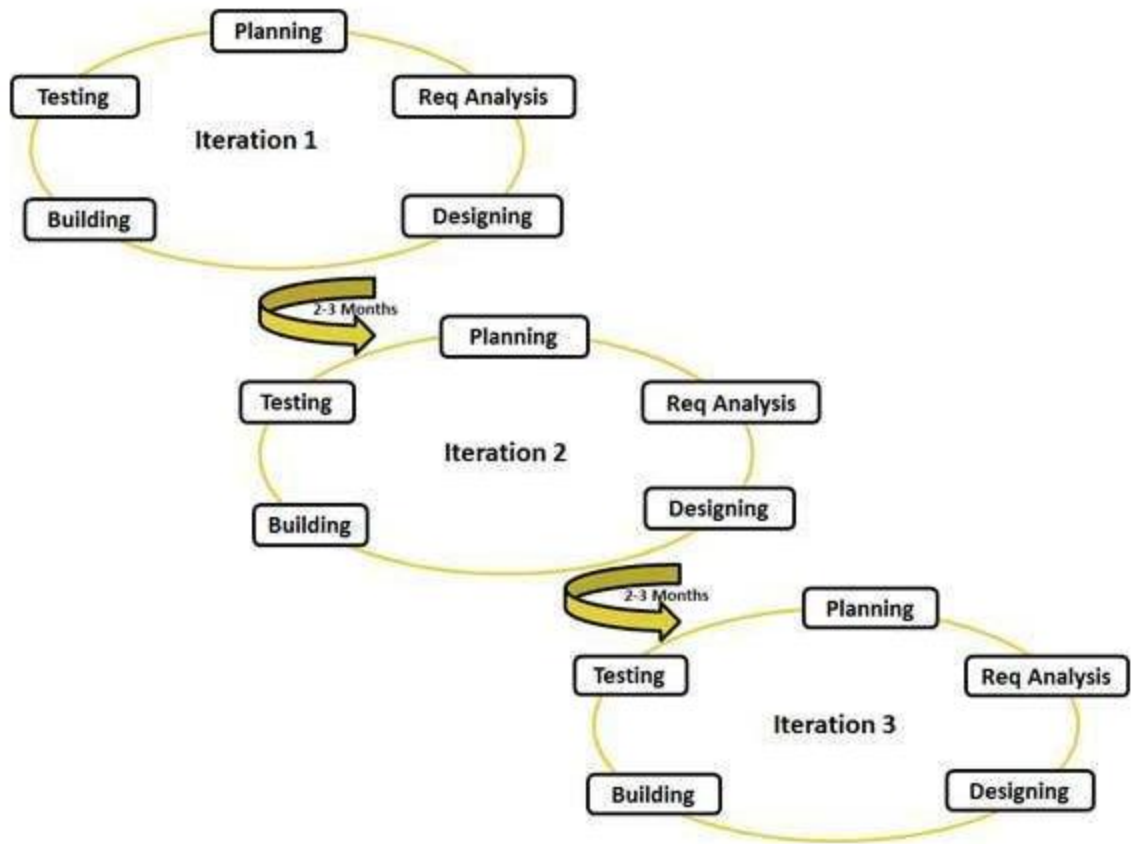
- ▶ ניתן לפתח חלק מהפונקציונליות בשלב מוקדם של הפרוייקט.
- ▶ קל לנהל סיכונים – חלקים בדרגת סיכון גבוה יפותחו ראשונים.
- ▶ המודל תומך בשינויים בדרישות במהלך הפיתוח.

## ▶ חסרונות:

- ▶ מכיוון שלא כל הדרישות נאספות ומנותחות, בשלב מאוחר של הפרוייקט יכולות להתעורר בעיות עיצוביות/טכניות שלא נלקחו בחשבון.
- ▶ מצריך תשומת לב ניהולית רבה.
- ▶ ללא ניתוח מלא של דרישות המשתמש קשה לחלק את הפרוייקט לשלבים ומסירות.

# מודלים בהנדסת תוכנה - agile

► מודל איטרטיבי אינקרמנטלי.



# מודלים בהנדסת תוכנה - agile

---

## יתרונות:

- ▶ מצריך יחסית מעט תיעוד.

- ▶ מעודד עבודת צוות ושיתוף ידע.

- ▶ מסירות בסבבים קצרים.

## חסרונות:

- ▶ נשען חזק על הקשר עם המשתמש לאורך כל שלבי הפרוייקט.

- ▶ תלות חזקה במפתחים, בעיקר בגלל העדר תיעוד.

- ▶ מורכב יותר לתחזוקה והרחבות עתידיות.

- ▶ שיטה פופולרית מאוד בשוק ההייטק, scrum כמימוש פרקטי של השיטה.



# מודלים בהנדסת תוכנה - agile

---

## **Manifesto for Agile Software Development**

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.



# גישה איטרטיבית מול אינקרמנטלית

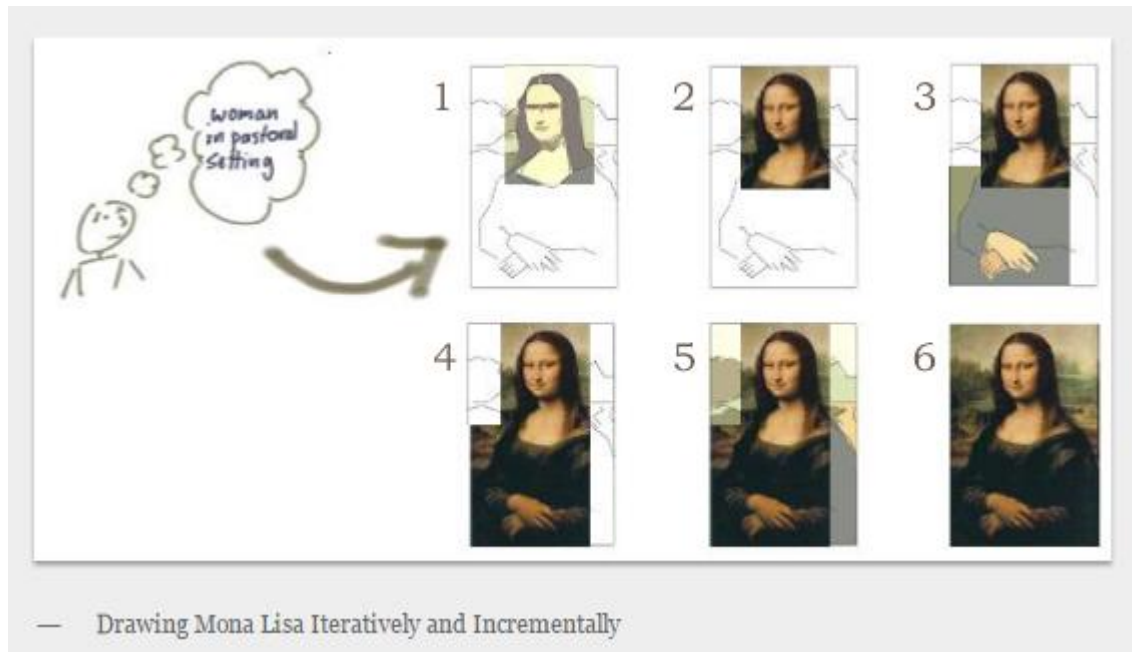


— Iterating Mona Lisa (Source: Jeff Patton)



— Incrementing Mona Lisa (Source: Jeff Patton)

# השילוב של שתי הגישות



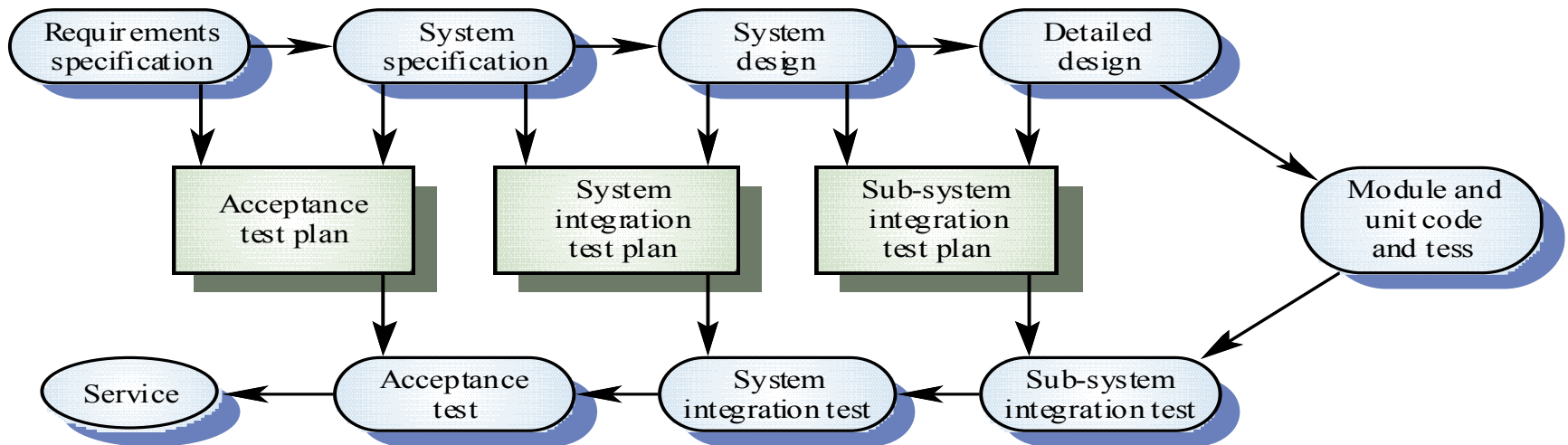
from <http://itsadeliverything.com/revisiting-the-iterative-incremental-mona-lisa>

‣ אילו סוגי בדיקות תוכנה קיימים?



# בדיקות תוכנה

מתי זה קורה? ▶



# בדיקות יחידה (Unit tests)

---

- ▶ נכתבים לכל מרכיב תוכנה בנפרד.
  - ▶ למשל, לכל מחלקה ב Java
  - ▶ לכל מודול ב C.
  - ▶ מתי מריצים?
  - ▶ בכל פעם שמשנים את הקוד.



## בדיקות יחידה

---

▶ דוגמא: מימוש פונקציה אשר מחזירה את תת המחרוזת המשותפת הארוכה ביותר בין שתי מחרוזות.

Sequence 1: **HUMAN**

Sequence 2: **CHIMPANZEE**

▶ מה נבדוק?

▶ אחת המחרוזות ריקה.

▶ שתי מחרוזות זהות.

▶ מחרוזות באורכים שונים.

▶ אין אף תת מחרוזת משותפת.

▶ יש יותר מתת מחרוזת משותפת בעלת אורך מקסימלי.

▶ מחרוזות ארוכות מאוד.

▶ מספר מקרים שבהם ניתן למצוא תתי מחרוזות משותפות, ברמות מורכבות שונות.

▶ ועוד...

---



## בדיקות אינטגרציה

---

- ▶ שני מודולים שיכולים להיות מפותחים ע"י מפתחים שונים משתמשים האחד בקוד של השני.
- ▶ כל מודול נבדק בנפרד אבל הם לא נבדקו ביחד.
- ▶ כיצד ניתן לפתח שני מודלים כאלה במקביל?





## בדיקות של כיסוי איפיון/עיצוב/דרישות

---

▶ מעבר על מסמכי איפיון/עיצוב/דרישות ווידוא שכל הסעיפים מומשו.

▶ אם כותבים את תסריטי הבדיקות לפני תחילת המימוש, סיכוי נמוך שנפספס משהו.

▶ ועדיין, צריך לבדוק כיסוי מלא, בעיקר במערכות עתירות פונקציונליות.



## בדיקות נוספות

---

- ▶ בדיקות מסירה.
- ▶ בדיקות קבלה ע"י המשתמשים.
- ▶ בדיקות אבטחת תוכנה.
- ▶ בדיקות יציבות (מה קורה כאשר אין תקשורת אינטרנט/נותק הקשר משרת ה DB וכו')
- ▶ בדיקות עומסים.
- ▶ צריכת זכרון.
- ▶ דליפות זכרון.



# ניהול זמנים

---

- ▶ **תכנון זמן לכל שלבי הפרוייקט:**
    - ▶ שריון זמן לבדיקות.
    - ▶ זמן לתיקונים של אחרי הבדיקות.
    - ▶ ניהול זמן של השרשרת הקריטית בפרוייקט.
  - ▶ **שימוש ב Buffer-ים בעת תכנון הלו"ז לפרוייקט.**
    - ▶ התמודדות עם בלת"מים.
    - ▶ תלות במשאבים טכנולוגיים.
  - ▶ **תכנון זמנים כך שניתן יהיה לטפל בבעיות תכנוניות שמתעוררות במהלך הפיתוח (למשל, גילינו שהעיצוב לא מספיק טוב).**
  - ▶ **כיצד נדע שתכנון הזמנים שלנו נכון?**
    - ▶ נבדוק את עצמנו אחרי השלמת כל אבן דרך בפרוייקט.
    - ▶ נתקן את תכנון הזמנים להמשך הפרוייקט במידת הצורך.
    - ▶ בהתאם להערכות הזמנים החדשות, ניתן לשנות את סדר מימוש התכולות, איפיון טכני וכו'.
- 



# ניהול גירסאות תוכנה

---

## ▶ למה צריך את זה?

- ▶ נשרף לי ה hard disk וכל הפרויקט נמחק.
- ▶ השותף שלי שינה משהו יומיים לפני ההגשה ועכשיו שום דבר לא עובד.
- ▶ תיקנו את הקוד בהרבה מקומות ובטעות מחקנו משהו, ועכשיו אנחנו לא מצליחים לשחזר איפה התבצעה המחיקה המוטעית.
- ▶ אני והשותף שלי רוצים לערוך את אותו הקובץ, אז אני מחכה שהוא ישלח לי את הגירסא המעודכנת שלו כדי שאעבוד עליה.
- ▶ תסריט אלטרנטיבי – אני והשותף שלי שינינו את אותו הקובץ, והוא דרס את השינויים שלי בטעות.



איך אפשר לנהל גירסאות?

---

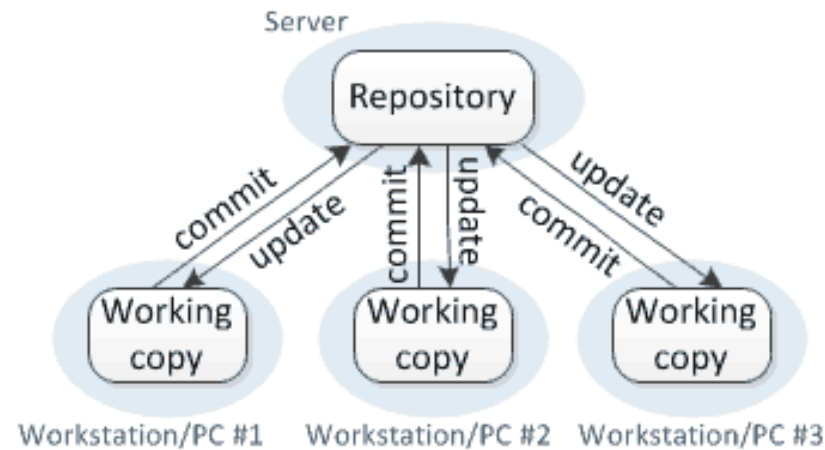
- ▶ שמירת עותקים על הדיסק הקשיח.
- ▶ שליחה במייל האחד לשני.
- ▶ ?Google drive

כלי ניהול גירסאות  
אוטומטיים !



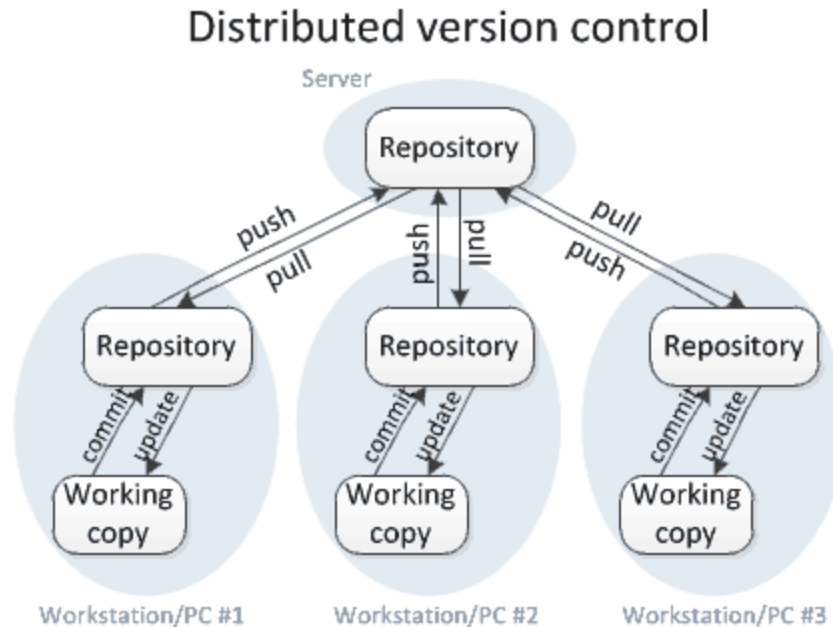
איך זה עובד?

## Centralized version control



לדוגמא: SVN ▶

איך זה עובד?



לדוגמא: GIT ▶

from <https://homes.cs.washington.edu/~mernst/advice/version-control.html>

# SVN plug-in for eclipse

The screenshot displays the Eclipse IDE with the SVN plug-in. The main editor shows a comparison between a local file (revision 3857) and a remote file (BASE) for `SVNClientManager.java`. The History view at the bottom, highlighted with a red box, shows the following table:

Revision	Date	Author	Comment
4298	2/18/09 4:03 PM	markhip	Merge all changes from tree-conflicts branch
4287	2/16/09 10:34 AM	selsemore	Return different paths for remote resources and works
4152	1/7/09 9:39 AM	selsemore	Log errors rather than printing stack trace. Issue #: 825
4150	1/6/09 10:27 AM	selsemore	Fix potential NPE if null progress monitor is passed to

Below the table, the file list shows:

- /trunk/subclipse/org.tigris.subversion.subclipse.core in <http://subclipse.tigris.org/svn/subclipse>
- RevertResourcesCommand.java

The comment for revision 4152 is expanded to show: "Log errors rather than printing stack trace. Issue #: 825".



# SVN plug-in for eclipse

Java Structure Compare

- Compilation Unit
  - HelperClass
    - say(String)
  - MyClass
    - hello()


Java Source Compare

Local File 15	Remote File 16 [tettamanti]
<pre>package polico;  public class MyClass {     public static void hello() {         HelperClass.say("Hello world");     } }</pre>	<pre>package polico;  public class MyClass {     public static void hello() {         HelperClass.say("hello!");     } }</pre>
<pre>class HelperClass {     public static void say(String msg) {         System.out.println(msg);     } }</pre>	<pre>class HelperClass {     public static void say(String msg) {         System.out.println("I say: " + msg);     } }</pre>

Copy Current Change from Right to Left

# איך אפשר להתחבר לשרת SVN

---

← → ↻  <https://www.cs.tau.ac.il/system/servers#servers12>

## SVN - version control system

**svn** עוזר כאשר מספר אנשים עובדים על פרוייקט משותף, כאשר כל מפתח עובד על הספרייה האישית שלו. **svn** משמש למיזוג בין הקבצים השונים. עליכם לעקוב אחר ההוראות הבאות:

בכדי לפתוח חשבון SVN יש לשלוח מייל ל [system@cs.tau.ac.il](mailto:system@cs.tau.ac.il) ובו לציין את שם repository שרוצים שיהיה לפרוייקט ואת שמות המשתמשים שתהיה להם גישה ל repository.



# חשבון ב GitHub לסטודנטים

GitHub Education

Stories Events Student pack Classroom Contact us

TEACH AND LEARN  
BETTER, TOGETHER

Request a discount

STUDENT DEVELOPER PACK

Get the Student Developer Pack

Dozens of free resources from great companies to help students learn.

Get the pack

<https://education.github.com/>

מתי מומלץ לעבוד עם כלים לניהול גירסאות?

---

► כשיש לנו פרוייקט שעובד עליו לפחות בן אדם אחד.



<http://www.dreamstime.com/royalty-free-stock-photo-computer-operator-illustration-woman-who-became-company-image33328965>

## סקרי קוד

---

### יתרונות:

- ▶ גם מתכנתים ותיקים עושים טעויות.
- ▶ עוד אנשים מכירים את הקוד.
- ▶ גם אם אין באגים, אפשר לשפר עיצוב, ליעל את הקוד אם מתסכלים עליו עם עוד מתכנת.
- ▶ כמו הרבה דברים אחרים – כשמסבירים את הקוד שלנו לאדם אחר, זה גורם לנו להתעמק בו ולאתר בעיות.



### חסרונות

- ▶ גוזל זמן.
  - ▶ מעצבן.
- 



## הדפסות בקוד

---

▶ משתמשים בהם הרבה במהלך פיתוח התוכנה

▶ סוגי הדפסות:

▶ אינפורמציה על ריצת התוכנית:

▶ הפונקציה רצה 50 שניות.

▶ הפונקציה טענה X קבצים.

▶ ביצענו קריאה לפונקציה מסויימת.

▶ דברים לשים אליהם לב:

▶ הפונקציה קיבלה מחרוזת ריקה, ולכן היא לא מבצעת עליה שום פעולה.

▶ מידע שימושי לניפוי שגיאות:

▶ הדפסת פרמטרים לפונקציות.

▶ הדפסת ערכי החזרה.

▶ הדפסת מספר הפעמים שלולאה מסויימת רצה עד שהסתיימה.

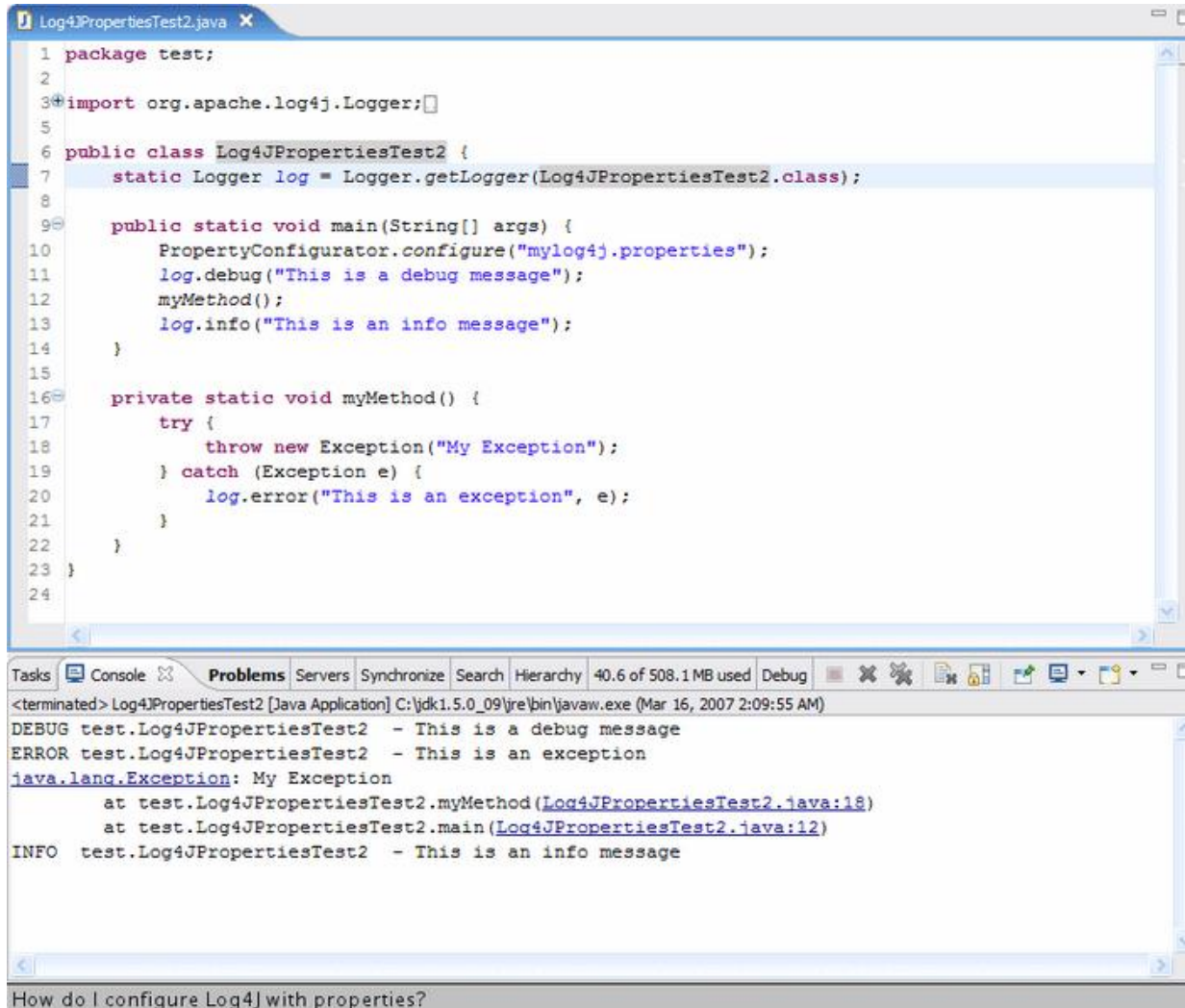
▶ שגיאות:

▶ פרמטרים לא נכונים לתוכנית.

---



# הדפסות בקוד – שימוש ב logger-ים



```
1 package test;
2
3 import org.apache.log4j.Logger;
4
5
6 public class Log4JPropertiesTest2 {
7     static Logger log = Logger.getLogger(Log4JPropertiesTest2.class);
8
9
10    public static void main(String[] args) {
11        PropertyConfigurator.configure("mylog4j.properties");
12        log.debug("This is a debug message");
13        myMethod();
14        log.info("This is an info message");
15    }
16
17    private static void myMethod() {
18        try {
19            throw new Exception("My Exception");
20        } catch (Exception e) {
21            log.error("This is an exception", e);
22        }
23    }
24 }
```

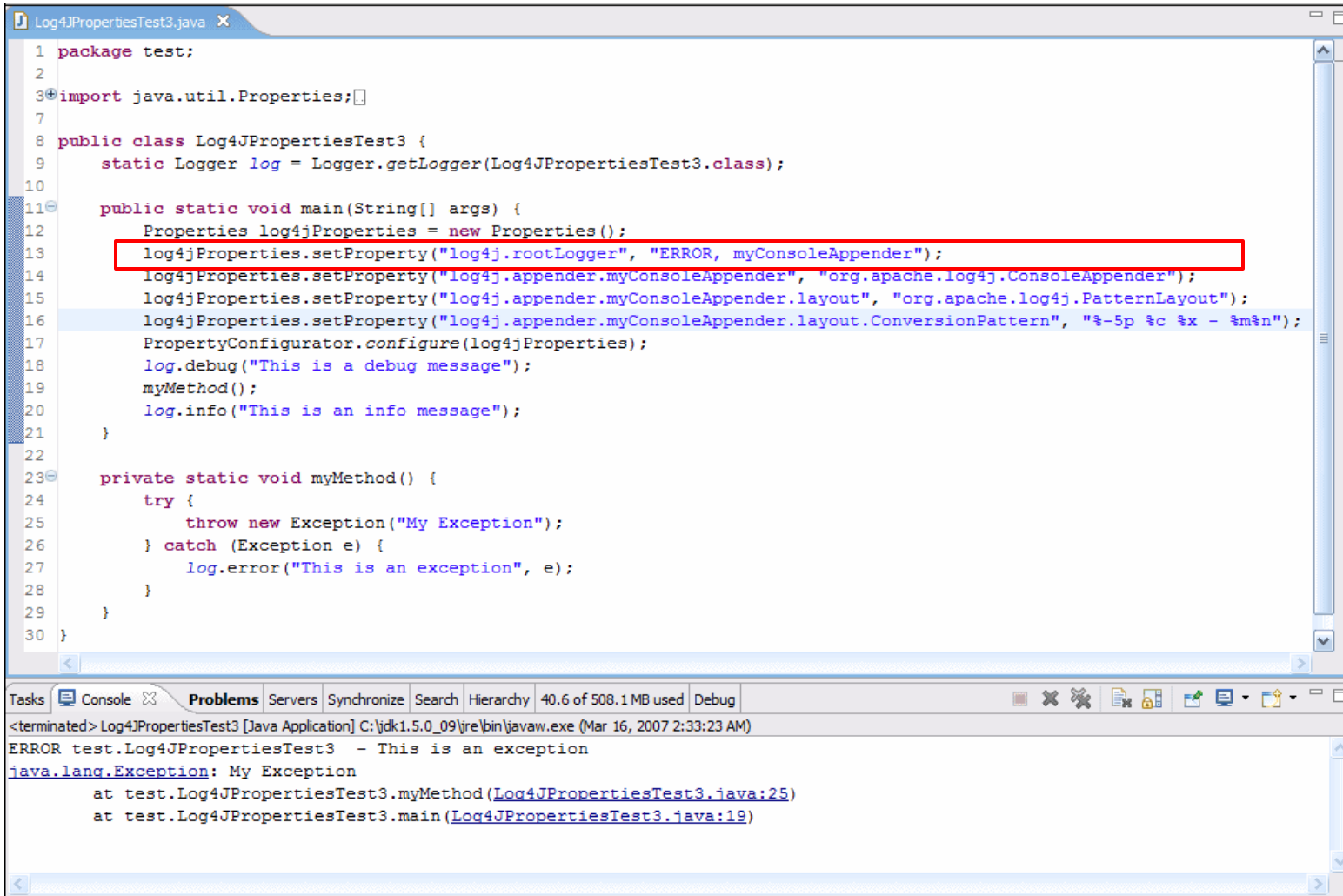
Tasks Console Problems Servers Synchronize Search Hierarchy 40.6 of 508.1MB used Debug

```
<terminated> Log4JPropertiesTest2 [Java Application] C:\jdk1.5.0_09\re\bin\javaw.exe (Mar 16, 2007 2:09:55 AM)
DEBUG test.Log4JPropertiesTest2 - This is a debug message
ERROR test.Log4JPropertiesTest2 - This is an exception
java.lang.Exception: My Exception
    at test.Log4JPropertiesTest2.myMethod(Log4JPropertiesTest2.java:18)
    at test.Log4JPropertiesTest2.main(Log4JPropertiesTest2.java:12)
INFO test.Log4JPropertiesTest2 - This is an info message
```

How do I configure Log4J with properties?

<http://www.avajava.com/tutorials/lessons/how-do-i-configure-log4j-with-properties.html?page=3>

# הדפסות בקוד – שימוש ב logger-ים



```
1 package test;
2
3 import java.util.Properties;
4
5
6
7
8 public class Log4JPropertiesTest3 {
9     static Logger log = Logger.getLogger(Log4JPropertiesTest3.class);
10
11     public static void main(String[] args) {
12         Properties log4jProperties = new Properties();
13         log4jProperties.setProperty("log4j.rootLogger", "ERROR, myConsoleAppender");
14         log4jProperties.setProperty("log4j.appender.myConsoleAppender", "org.apache.log4j.ConsoleAppender");
15         log4jProperties.setProperty("log4j.appender.myConsoleAppender.layout", "org.apache.log4j.PatternLayout");
16         log4jProperties.setProperty("log4j.appender.myConsoleAppender.layout.ConversionPattern", "%-5p %c %x - %m%n");
17         PropertyConfigurator.configure(log4jProperties);
18         log.debug("This is a debug message");
19         myMethod();
20         log.info("This is an info message");
21     }
22
23     private static void myMethod() {
24         try {
25             throw new Exception("My Exception");
26         } catch (Exception e) {
27             log.error("This is an exception", e);
28         }
29     }
30 }
```

Tasks Console Problems Servers Synchronize Search Hierarchy 40.6 of 508.1 MB used Debug

<terminated> Log4JPropertiesTest3 [Java Application] C:\jdk1.5.0\_09\jre\bin\javaw.exe (Mar 16, 2007 2:33:23 AM)

ERROR test.Log4JPropertiesTest3 - This is an exception

[java.lang.Exception](#): My Exception

- at test.Log4JPropertiesTest3.myMethod([Log4JPropertiesTest3.java:25](#))
- at test.Log4JPropertiesTest3.main([Log4JPropertiesTest3.java:19](#))

<http://www.avajava.com/tutorials/lessons/how-do-i-configure-log4j-with-properties.html?page=3>



# הדפסות debug כשאין שימוש ב logger

---

▶ להדפיס את כל המידע

▶ בחלק מהמקרים, בגלל בעיות של null-ים וכו' הדפסת ה debug עצמה יכולה להיכשל, ולכן צריך לטפל בכל המקרים הבעייתיים גם בהדפסה.

▶ כדאי להדפיס את המיקום שבו מתבצעת ההדפסה.

▶ להשתמש בכמה רמות הדפסה – יש הדפסות debug שנרצה לכבות בשלב מסויים, בעוד שהדפסות אינפורמטיביות נרצה להשאיר למשך כל שלב הפיתוח.



# מימוש logger בשפת C

```
#ifndef __dbg_h__
#define __dbg_h__

#include <stdio.h>
#include <errno.h>
#include <string.h>

#ifdef NDEBUG ←
#define debug(M, ...)
#else
#define debug(M, ...) fprintf(stderr, "DEBUG %s:%d: " M "\n", __FILE__, __LINE__, ##__VA_ARGS__)
#endif

#define clean_errno() (errno == 0 ? "None" : strerror(errno))

#define log_err(M, ...) fprintf(stderr, "[ERROR] (%s:%d: errno: %s) " M "\n", __FILE__, __LINE__, clean_errno(), ##__VA_ARGS__)

#define log_warn(M, ...) fprintf(stderr, "[WARN] (%s:%d: errno: %s) " M "\n", __FILE__, __LINE__, clean_errno(), ##__VA_ARGS__)

#define log_info(M, ...) fprintf(stderr, "[INFO] (%s:%d) " M "\n", __FILE__, __LINE__, ##__VA_ARGS__)

#define check(A, M, ...) if(!(A)) { log_err(M, ##__VA_ARGS__); errno=0; goto error; }

#define sentinel(M, ...) { log_err(M, ##__VA_ARGS__); errno=0; goto error; }

#define check_mem(A) check((A), "Out of memory.")

#define check_debug(A, M, ...) if(!(A)) { debug(M, ##__VA_ARGS__); errno=0; goto error; }

#endif
```

ניתן לאתחל בזמן קומפילציה עם -D

<http://c.learncodethehardway.org/book/ex20.html>

# הדפסות debug

---

- ▶ כדאי לשים קטעי קוד קריטיים שנרצה "לעקוב" אחריהם כדאי לעטוף בפונקציה, אחרת יהיו המון הדפסות debug בתוך הקוד.

```
void* mymalloc (size_t size, char *funcName, char* varName)
{
    printf("In function %s: allocating %d for variable %s", funcName, size, varName);
    return malloc(size);
}
```



## שימוש נקודת עצירה מותנית (באנגלית זה נשמע טוב יותר)

- ▶ השתמשו ב conditional break points.
- ▶ ומה קורה כשמנפה השגיאות (debugger) לא מאפשר?
- ▶ הכנסת תנאים לתוך הקוד:

```
100 void foo(int a, int b, int c) {
101     ... blah ...

120     if (((a % 3) == 0) ||
121         ((b < (c * 2)) && ((c + a) < 0))) {
122         printf("BREAK!!!");
123     }

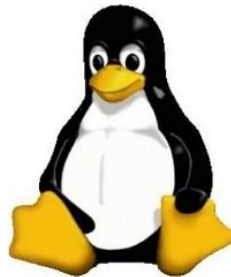
150     ... bleh ...
151 }
```



# מדוע הפסקתי לפחד והתחלתי לאהוב סקריפטים?

---

- ▶ יש לא מעט תוכניות שעדיף לכתוב כסקריפט:
- ▶ לספור כמה מילים באורך 3 תוים לפחות יש בקובץ גדול.
- ▶ להריץ תוכנית כלשהי על כמה קלטים ולהשוות את הפלט לפלט המצופה (תרגילי בית ברוב קורסי התכנות נבדקים ע"י סקריפטים).
- ▶ לנסות הרבה קונפיגורציות עבור אלגוריתם ולבחור את הקונפיגורציה האופטימלית.



- ▶ בלינוקס זה לא היה קורה...



# באג בדיזיין = באסה בדיבאג

---

- ▶ לפני שאנחנו ניגשים לכתוב קוד:
    - ▶ האם ברור לנו מה הוא צריך לבצע?
      - ▶ מקרי קצה
      - ▶ התמודדות עם שגיאות
      - ▶ קלט פלט.
    - ▶ האם ברור לנו איך יראה העיצוב?
      - ▶ היררכיית מחלקות/חלוקה לקבצים.
      - ▶ מבני הנתונים בהם נשתמש.
      - ▶ אלגוריתמים שצריך לממש.
      - ▶ חלוקה למודולים.
    - ▶ האם ברור לנו איך נבדוק את הקוד?
      - ▶ תסריטי בדיקות.
      - ▶ קלטים שקיבלנו מהמשתמש.
      - ▶ בדיקות זכרון/ביצועים.
      - ▶ אינטגרציה עם מערכות אחרות.
- 



---

# שאלות?

