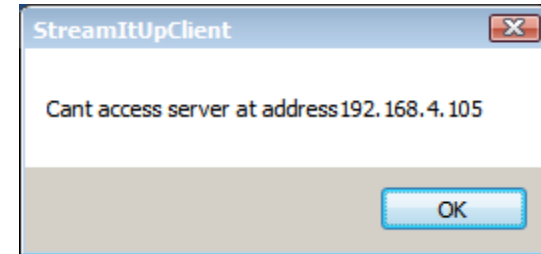


הלל: מה ששנא עליך על
תעשה לחברך



הנדסת תוכנה

13. חווית משתמש – UX

שחרור ותחזוקה + כלים ||

"I have always wished for my computer to be as easy to use as my telephone; my wish has come true because I can no longer figure out how to use my telephone"

[Bjarne Stroustrup](#) (C++ Creator)



http://groups.google.com/group/microsoft.public.powerpoint/browse_thread/thread/27c8f0b03f69fd4d?hl=en&ie=UTF-8&q=powerpoint+undo+deleted+slides#6608dc06dad9ca8a



microsoft.public.powerpoint

can i get a deleted slide back?

★ 3 messages - [Collapse all](#)

Crowe [View profile](#)

I **deleted** a slide from my **powerpoint** slide and hit "save". I need that slide back! Any way to get it???

sigh

Thanks!

[Reply](#) [Reply to author](#) [Forward](#)

Rate this post: ★★★★★

Sandy Johnson [View profile](#)

Dear Crowe,

Sorry can't get it back -- unless you saved it earlier under a different name or perhaps emailed the file to someone previous to saving it...

היום?



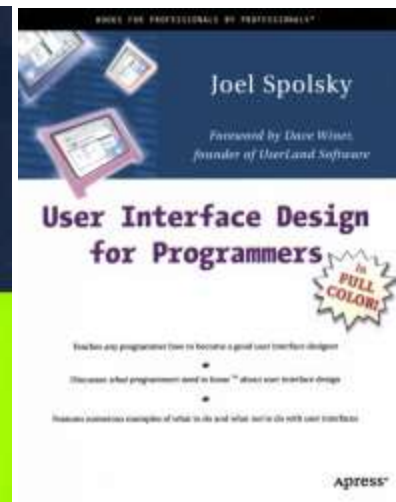
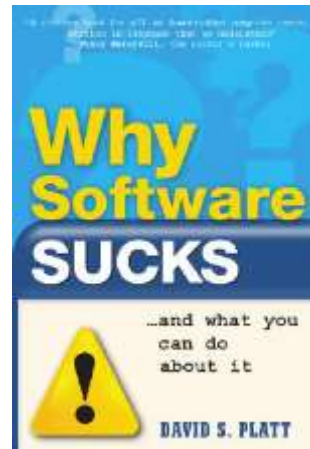
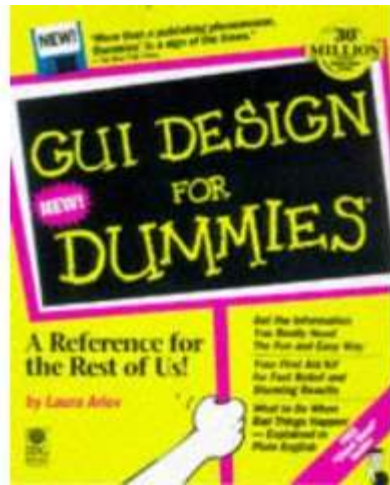
- חווית משתמש
 - שמישות \ עיצוב ממשק משתמש
 - GUI מסורתי, web והתקנים ניידים
- שחרור ותחזוקת תוכנה
- כלים מתקדמים
- שעה 3 \ תרגיל – משימת סבב אחרון
 - Code Review with pull requests
 - סביבת CI / CD (בונוס)
- הכנה להרצאה האחרונה (סקר)

Brooks (MMM): Question:
How does a large software
project get to be one year
late?
se14b-yagel

Answer: One day at a time!

מקורות

- Pressman Chap. 12
- ברק דנין, מדריכי שמישות (ול-סמארטפון)
- Amber, Agile Usability
- Spolsky, User Interface Design For Programmers
<http://www.joelonsoftware.com/primerFriendly/uibook/fog0000000249.html>
- Usability In Practice, MSDN series



Usability in the Front

- [Windows 8 UX Manager](#)
- [The Science and Art of User Experience at Google](#) (gtalk example)
- [Google Hangout Button](#) (btw [readability](#)...)



מאפייני מערכת לדוגמא

- פונקציונליות
- גודל
- ביצועים
- מחיר
- אמינות
- בטיחות
- סטנדרטים
- חווית\ממשק משתמש\שמישות
- בתיכון מערכת צריך לשקלל מאפיינים שונים
- היום דגש מסוים

סוגי ממשקים

- ממשק בין רכיבים
- ממשק למערכות חיצוניות
- ממשק משתמש

שלשת חוקי הזהב [Mandel 97]

- Place the user in control
- Reduce the user's memory load
- Make the interface consistent

Place the User in Control

- Define **interaction** modes in a way that does not force a user into unnecessary or undesired actions.
- Provide for flexible interaction.
- Allow user interaction to be interruptible and undoable.
- Streamline interaction as skill levels advance and allow the interaction to be customized.
- Hide technical internals from the casual user.
- Design for direct interaction with objects that appear on the screen.

Reduce the User's Memory Load

- Reduce demand on short-term memory.
- Establish meaningful defaults.
- Define shortcuts that are intuitive.
- The visual layout of the interface should be based on a real world metaphor.
- Disclose information in a progressive fashion.

Make the Interface Consistent

- Allow the user to put the current task into a meaningful context.
- Maintain consistency across a family of applications.
- If past interactive models have created user expectations, do not make changes unless there is a compelling reason to do so.

ממשק משתמש – לא רק ציור

- עיצוב אינטראקטיביות משתמש
- ארגון המידע
- מחקר משתמשים
- עיצוב ויזואלי

חווית משתמש (נילסן\נורמן)

"חווית משתמש כוללת את כל האספקטים של האינטראקציה בין משתמש הקצה לבין החברה, השירותים שלה והמוצרים שלה. הדרישה הבסיסית לחווית משתמש מעולה היא לענות על הצרכים המדויקים של המשתמש, ללא סיבוכים מיותרים. על המוצר או השירות להיות פשוט ואלגנטי, **כך שיהיה כיף להחזיק בו, וכיף להשתמש בו.** חווית משתמש אמיתית נותנת ללקוחות הרבה יותר מאשר בדיוק את מה שהם אומרים שהם רוצים, או ממלאת אחרי רשימת יכולות ודרישות. כדי להגיע לחווית משתמש באיכות גבוהה על-פני כל החברה, חייב להיות **רצף אחד על-פני כל השירותים** שהיא מספקת בתחומים שונים, כולל הנדסה, שיווק, עיצוב גראפי ותעשייתי, ועיצוב ממשק".

Usability - שמישות

- מידת האפקטיביות שבה משתמש יכול להשיג את מטרותיו עם התוכנה
- חלק מ- Human Computer Interaction, חיתוך עם מדעים קוגניטיביים \ ארגונומיה-פיזיולוגיה \ הנדסת אנוש
- המשתמש במרכז!
- ISO Definition
- Morning: designed for use (video)

DILBERT by Scott Adams





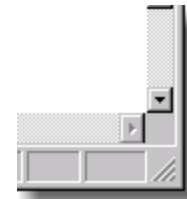
מאפייני שמישות (Nielsen)

- **לימודיות** - המידה שבה הממשק ניתן ללמידה על ידי המשתמש
- **יעילות** - המידה שבה השימוש בממשק הוא יעיל
- **זכירות** - מידת הקלות שבה המשתמש זוכר את פעולות הממשק.
- **שגיאות** - כמות הטעויות הפוטנציאלית הקיימת בממשק
- **שביעות רצון** - מידת שביעות הרצון של המשתמש מהממשק

Affordance - מושג קרוב: נגישות |

- מצב שבו המאפיינים החושיים של אובייקט מרמזים על אופן השימוש בו

– כפתור בולט, מציע שאמורים בד"כ ללחוץ עליו
– פינה עבה של חלון מרמזת על אפשרות להגדלה



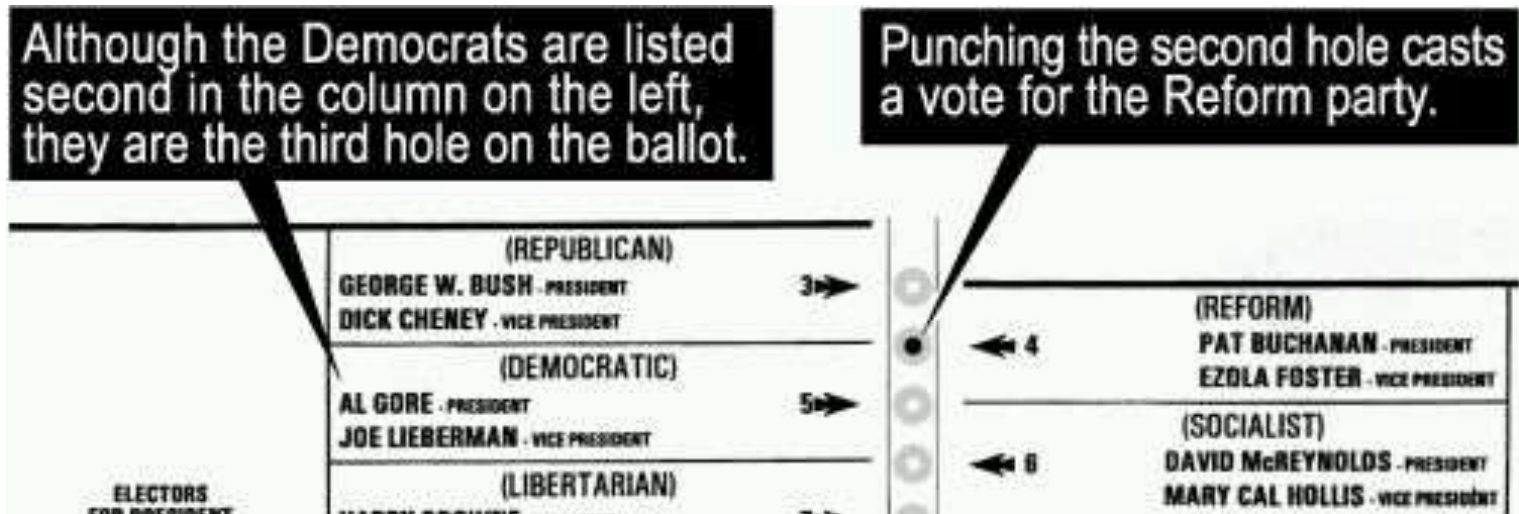
- תכונה רצויה לממשק משתמש, עוד בהמשך...
- [Affordance Definition](#), [Udacity Course](#)

נגישות II - Accessibility

- נגישות למרות מוגבלויות שונות
 - ראייה
 - מוטוריקה
- נגישות למניעת מחלות וכדו'
- תקנים:
- Web Content Accessibility Guidelines (WCAG)
 - <http://www.w3.org/TR/WCAG20/>
- <http://www.access-board.gov/508.htm>
- Tools: Herra, Booby

שמישות וממשק משתמש

- קשר הדוק בין השניים
- לממשק גרוע יכולה להיות השפעה רבה



Apollo Usability

- We have a problem
- UI issue?
- Russ Olsen - To the Moon!
@ a Clojure conf. 2013

כיצד משיגים שמישות?

- מבדקי שמישות (Usability Testing)
 - Eye Tracking
 - ראיונות\סקרים עם מומחים\ קבוצות מיקוד
 - אב טיפוס
 - נייר
 - קוד (VB, Rails)
 - פיתוח איטרטיבי
 - במיוחד ברשת
 - תהליכי עיצוב



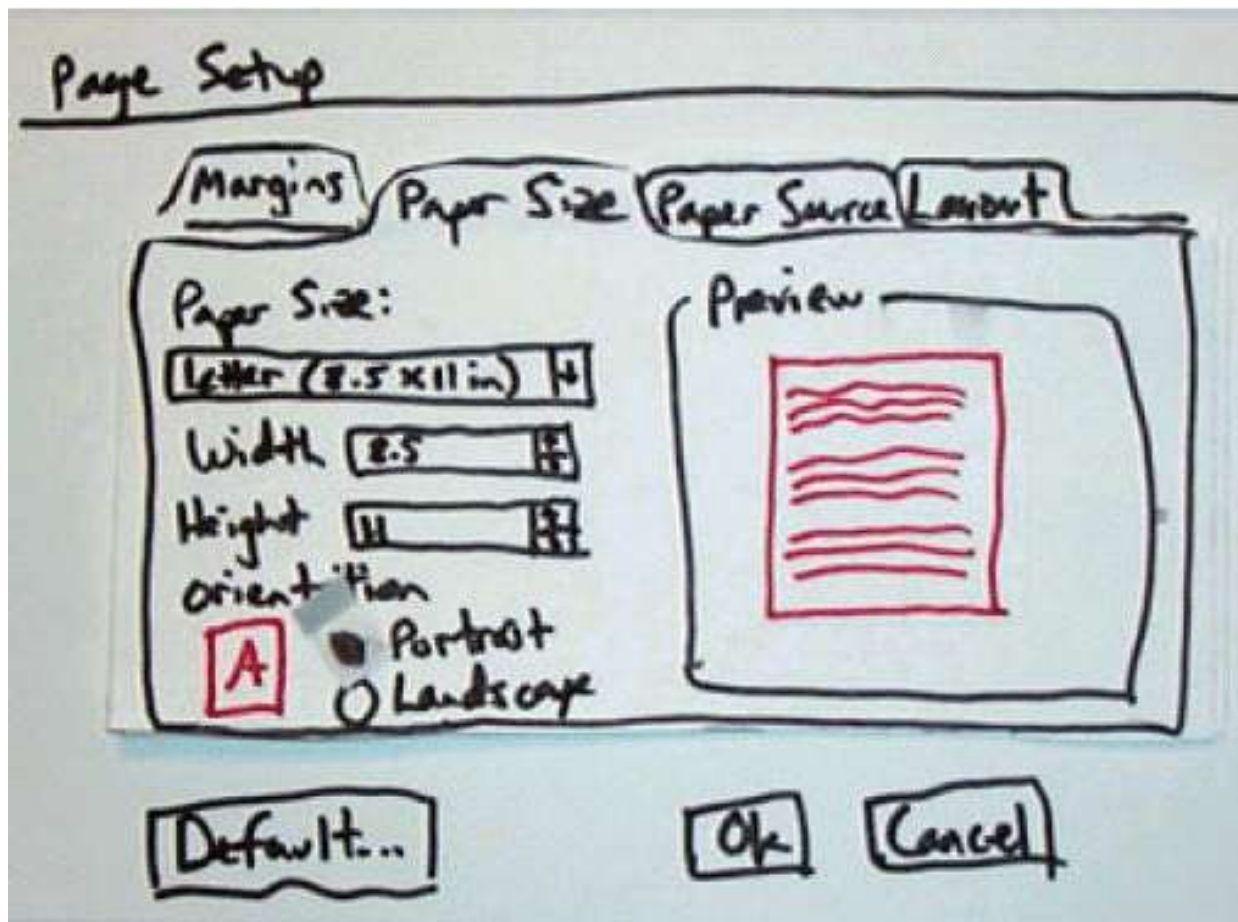
DILBERT reprinted by permission of United Feature Syndicate.

אב טיפוס מנייר

- משחק תפקידים, ללא מחשב



אב טיפוס מנייר



יתרונות לנייר

- בדיקת הרעיונות לפני המימוש
- ביצוע שינויים מהיר
- אפשרות לגלות מה באמת צריך
- נטרול משתנים טכנולוגיים ומימושיים
 - עוד צעד לקראת הלקוח
 - שינויים נראים קלים יותר
 - התמקדות בדברים העקרוניים

בתוכנה



- סביבות פיתוח מהירות, למשל:

- [Ruby on Rails](#)
- Nancy, [ASP.NET MVC](#)

- סביבות אבות טיפוס:

- לאתרים: <https://gomockingbird.com/>
- [SketchFlow](#), MS Expression Blend, [Video](#)

- [BalsamiQ Mockups](#), [וידאו](#)

- <http://www.pretotyping.org/androgen>

- איך עושים זאת? למשל: <http://astrails.com/>

- VS2012 Story Boarding

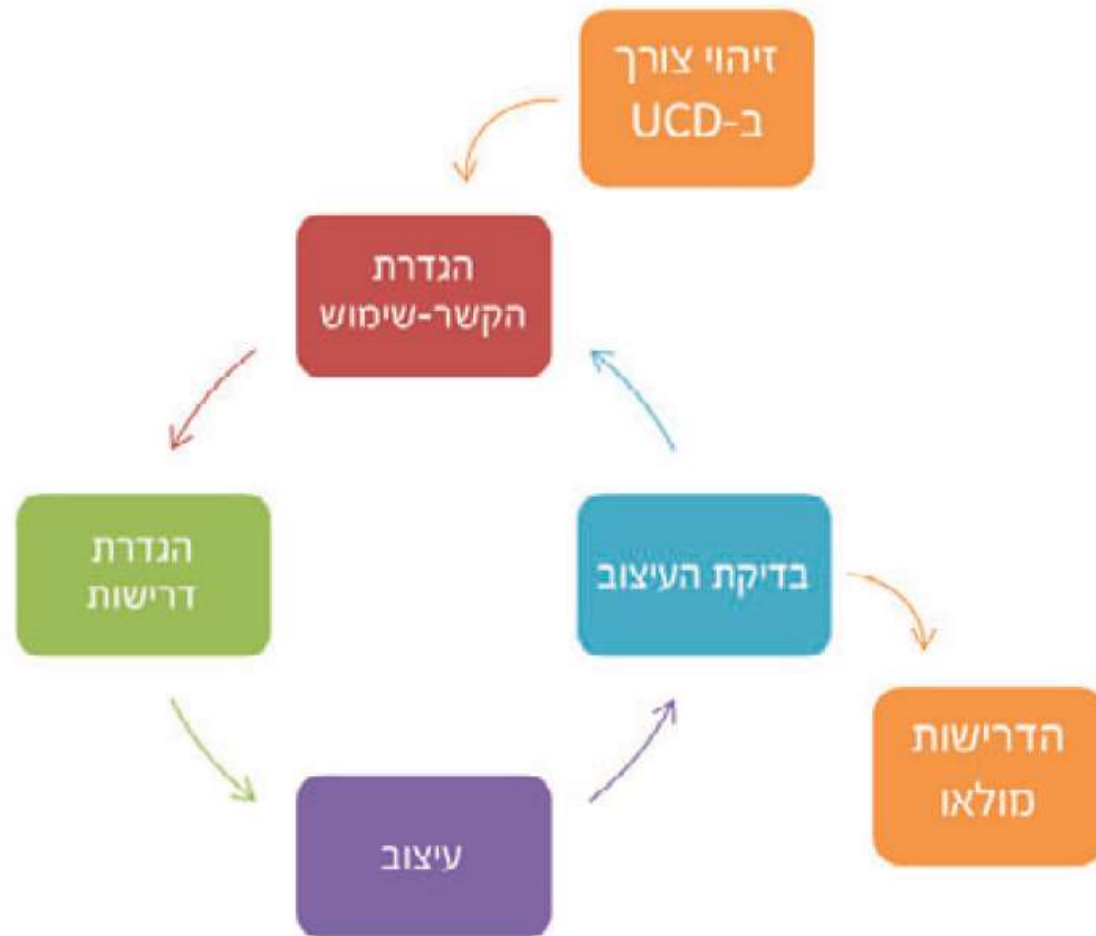
- בדיקות אוטומטיות, למשל [Selenium](#)

- Julian Harty, [Finding Usability Bugs with Automated Tests](#)

איזה סוג בדיקה כללית שלמדנו הכי משמעותי לחוויית משתמש

1. בדיקות קבלה (Acceptance)
2. בדיקות שמישות (מעבדת שמישות)
3. סקר דרישות
4. בדיקות יחידה

תהליך עיצוב מוכוון משתמש (UCD)



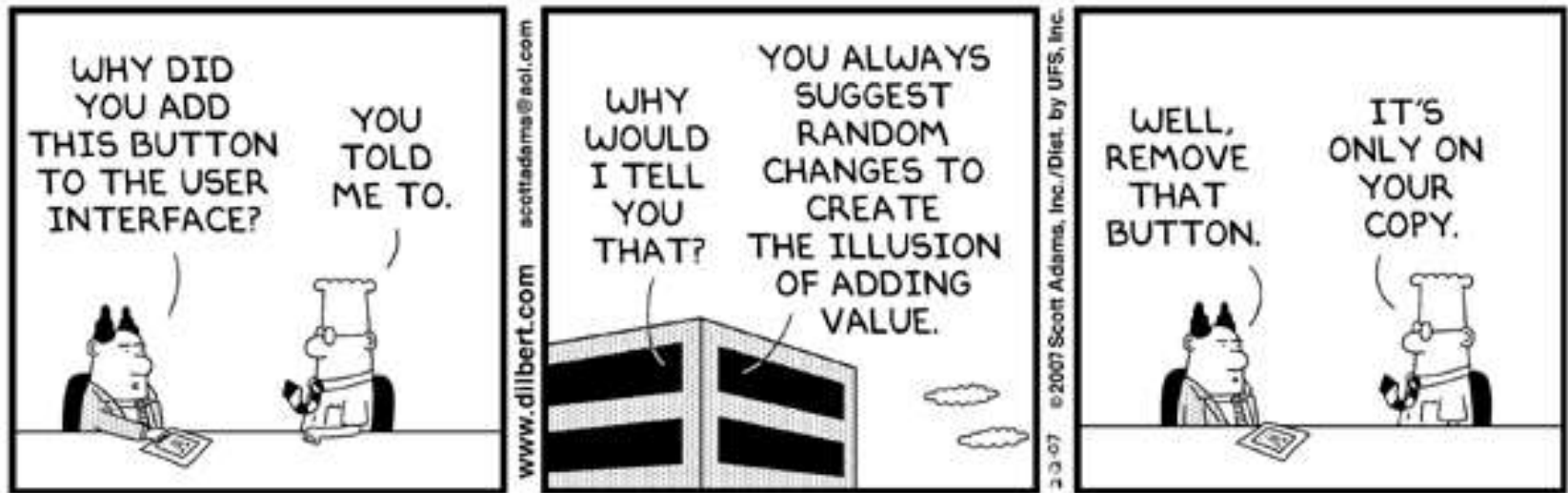
מתי בודקים?

- “The rule of thumb...is that the cost-benefit ratio for **usability** is **\$1:\$10-\$100**. Once a system is in **development**, correcting a problem costs 10 times as much as fixing the same problem in **design**. If the system has been **released**, it costs 100 times as much relative to fixing in design.” (Gilb, 1988, [here](#))

User Centered Design •

עוד לא בדקתם? •

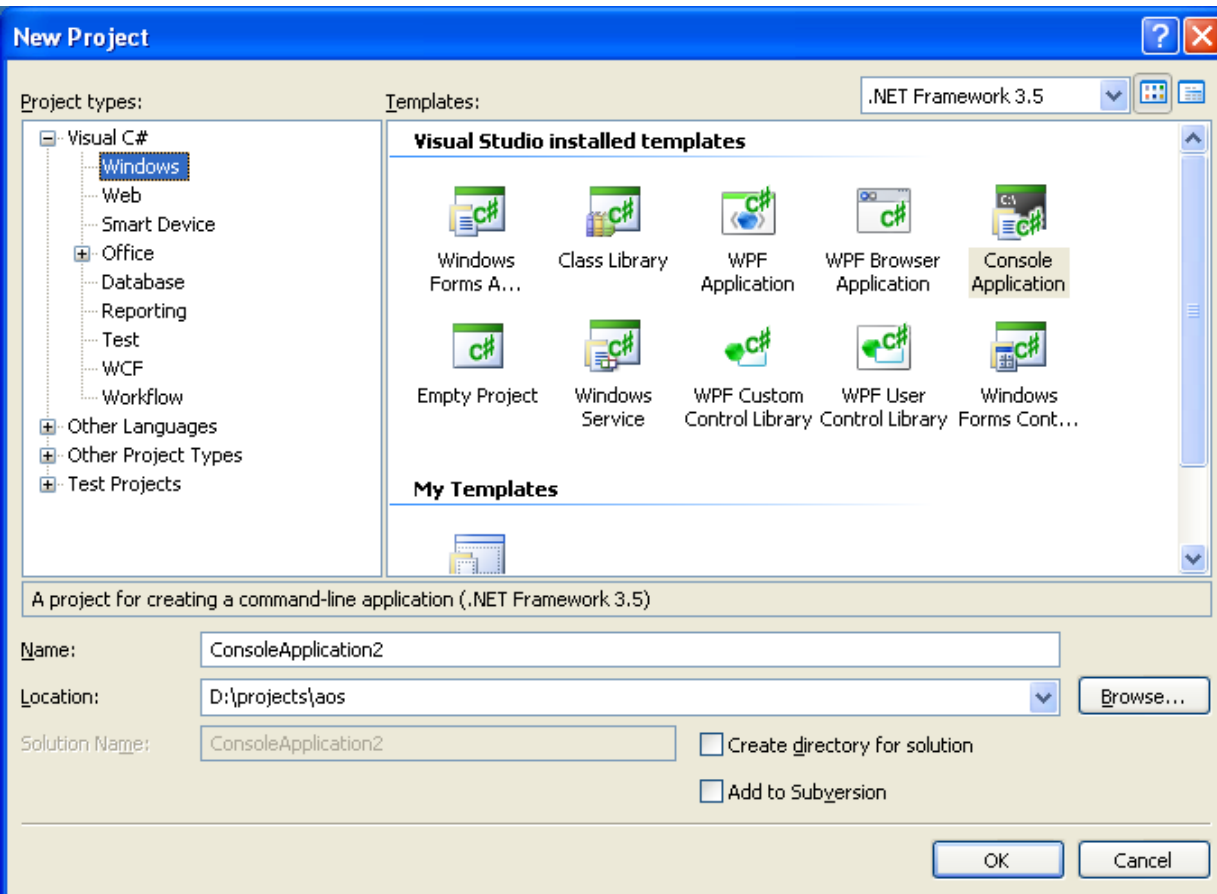
כמה עקרונות בסיסיים לתוכניות חלונאיות



© Scott Adams, Inc./Dist. by UFS, Inc.

מתי כדאי להשתמש ב:

- כפתור?
- תיבת סימון?
- כפתור רדיו?
- שדה טקסט?
- רשימה/נגללת?
- עץ?
- תפריט?
- תיבת שיחה?
- אחרים....?



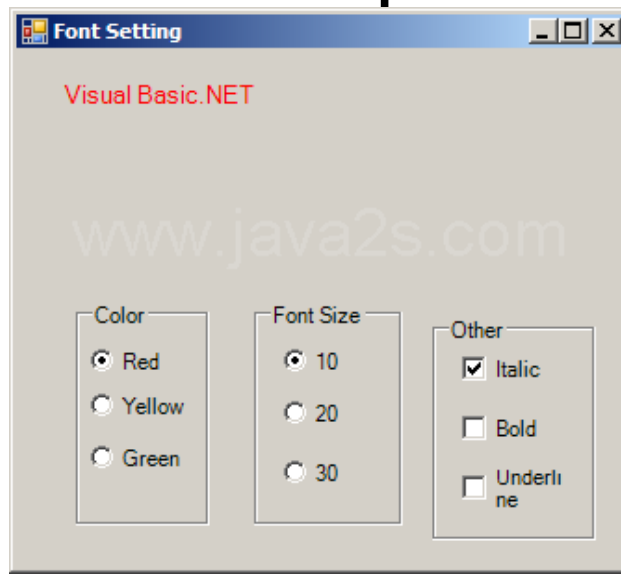
כפתורים, תפריטים

- כפתור – עבור פעולות עצמאיות, רלוונטיות למסך הנוכחי
 - מתי שמים ... (שלש נקודות)?
- סרגל כלים – פעולות נפוצות
- תפריט – פעולות פחות נפוצות שיכולות להיות ישימות למספר מסכים (או כולם)



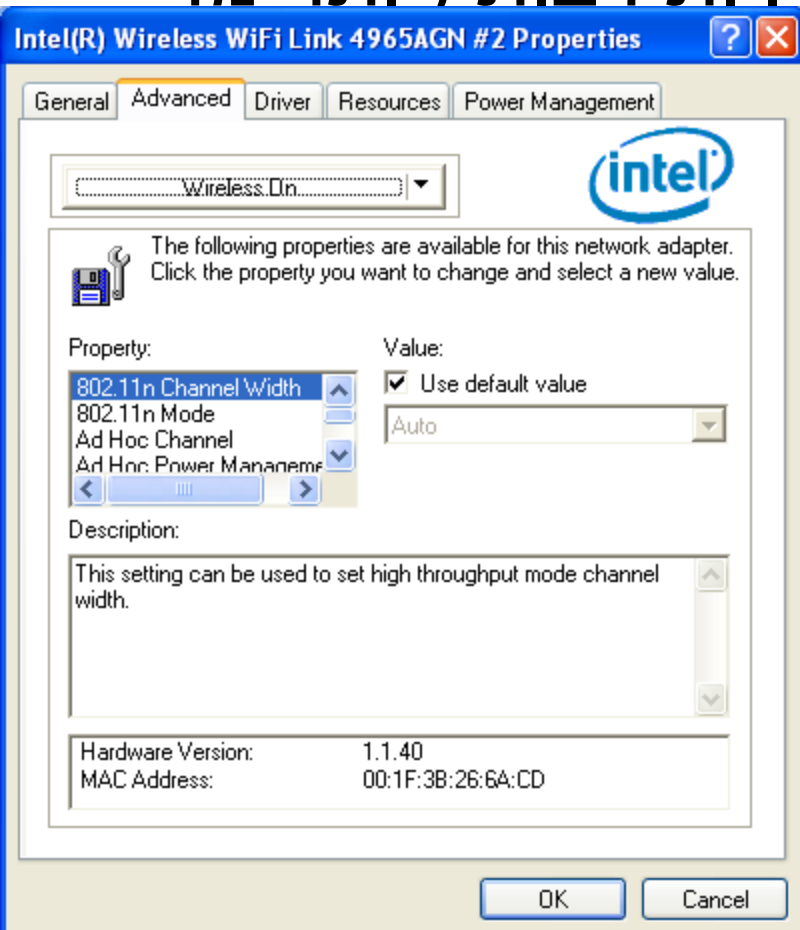
תיבות סימון

- Check-box – מפסקי הפעלה וכיבוי, כל אחד בלתי תלוי באחרים (לרוב תואם לערכים בוליאניים)
- Radio button – אפשרויות קשורות שרק אחת צריכה להיבחר (לרוב תואם לקבועים\enum)



רשימות

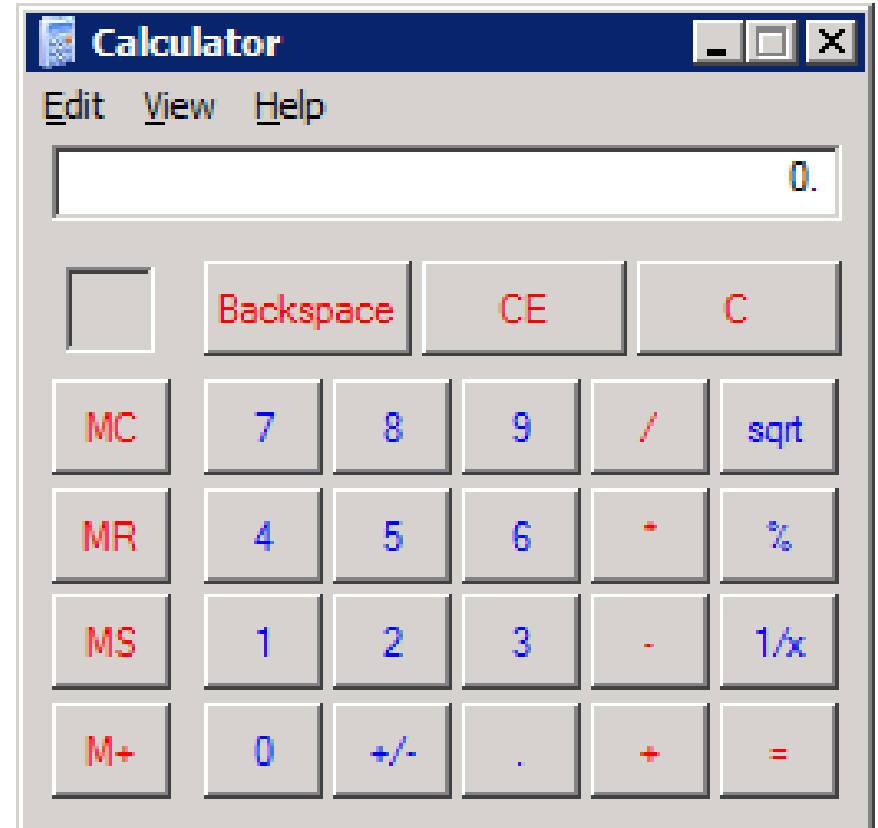
- טקסט (בד"כ בצרוף תגית) – קלט חופשי
- רשימה – בחירה מתוך אפשרויות רבות (יותר מדי בשביל רדיו) ורוצים שיראו את כולן
- Combo-box – כנ"ל, אך רוצים לחסוך במקום על המסך



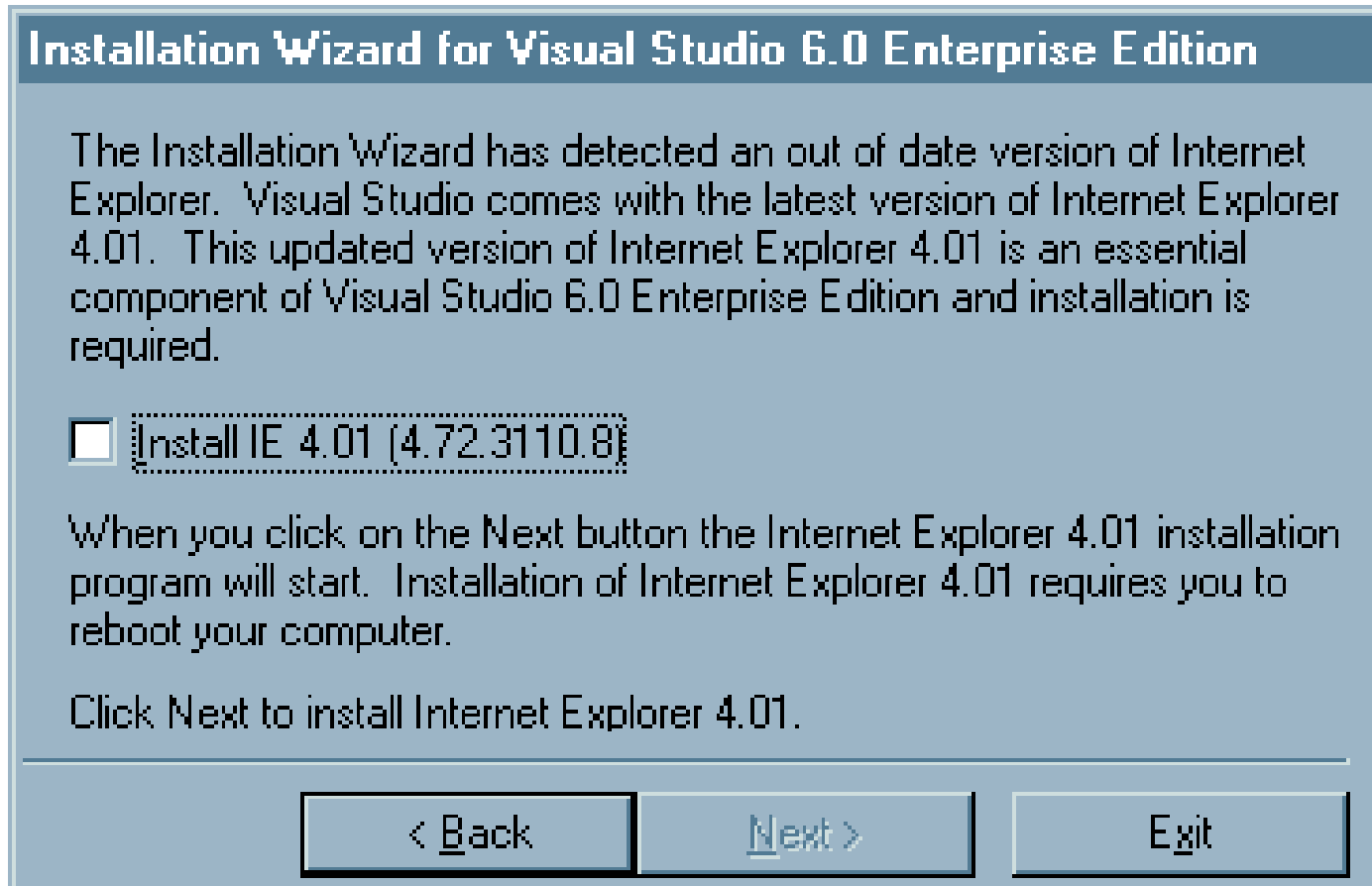
מסכים מרובים

- Tabbed Pane – המשתמש יכול לעבור בכל רגע בין המסכים השונים
- אשף (wizard) – הנחיית המשתמש לאורך תהליך
- תיבת שיחה – הצגת מסכים זמניים או תיבת מאפיינים

על ספסל הנאשמים



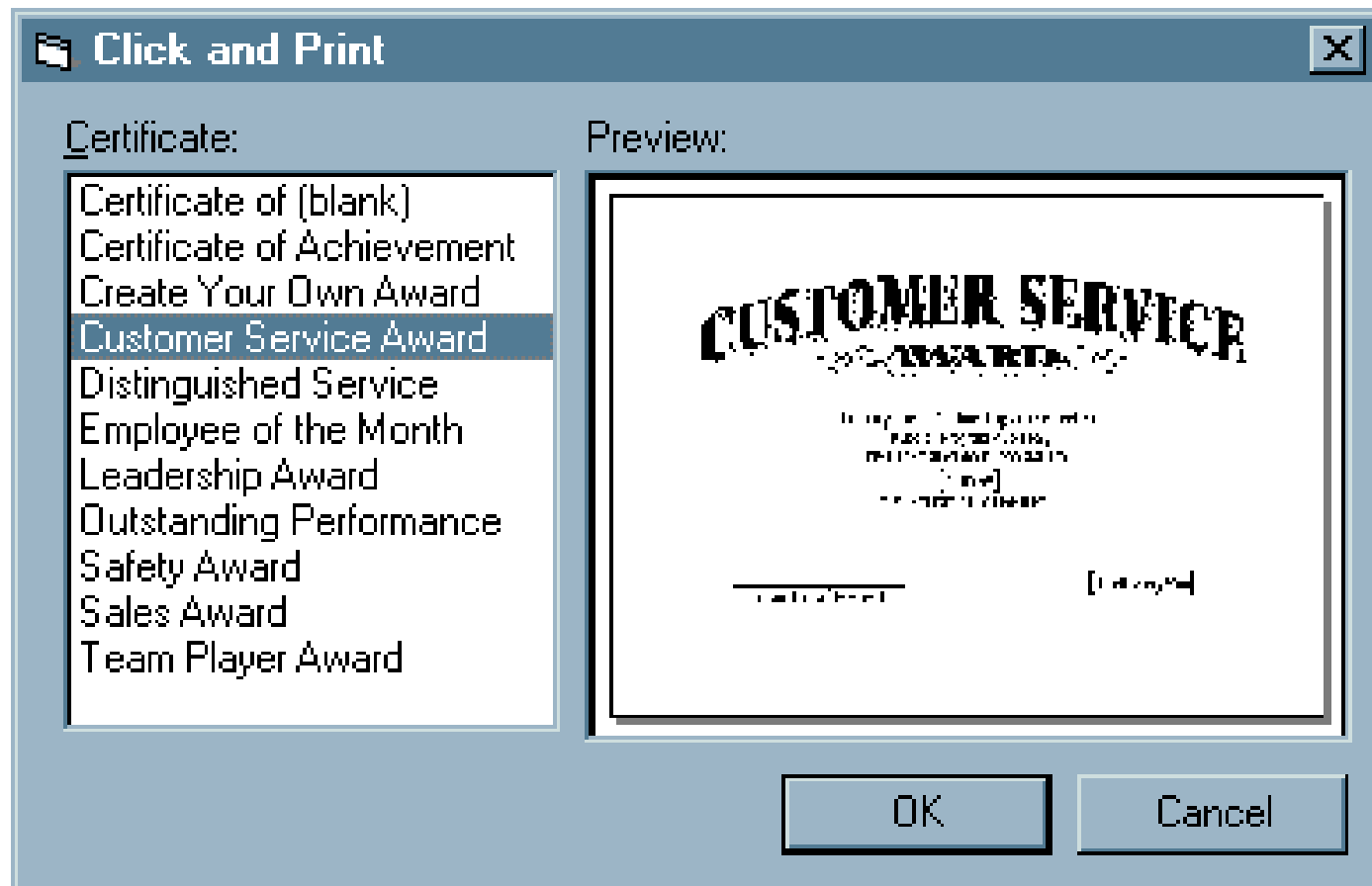
Interface Hall of Shame - דוגמאות



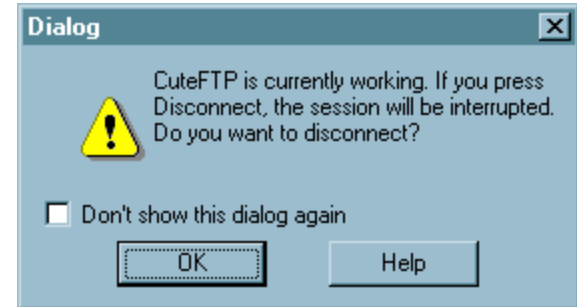
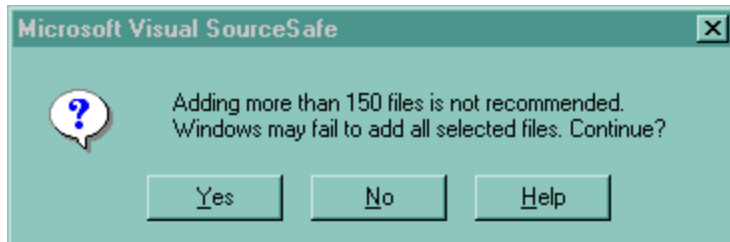
Interface Hall of Shame - דיון



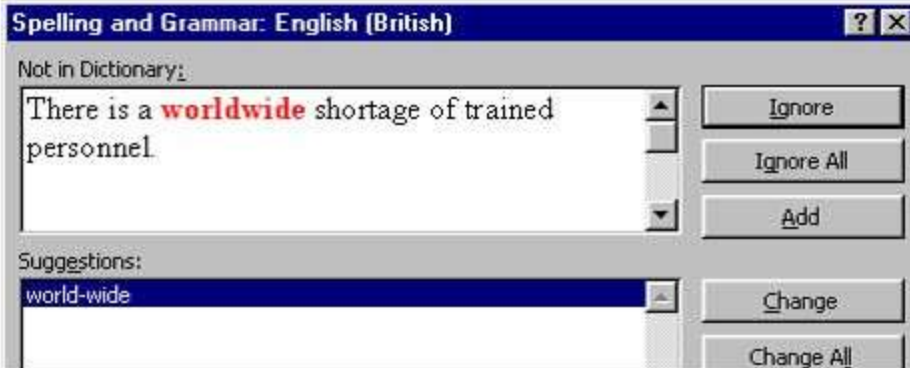
קצת יותר טוב



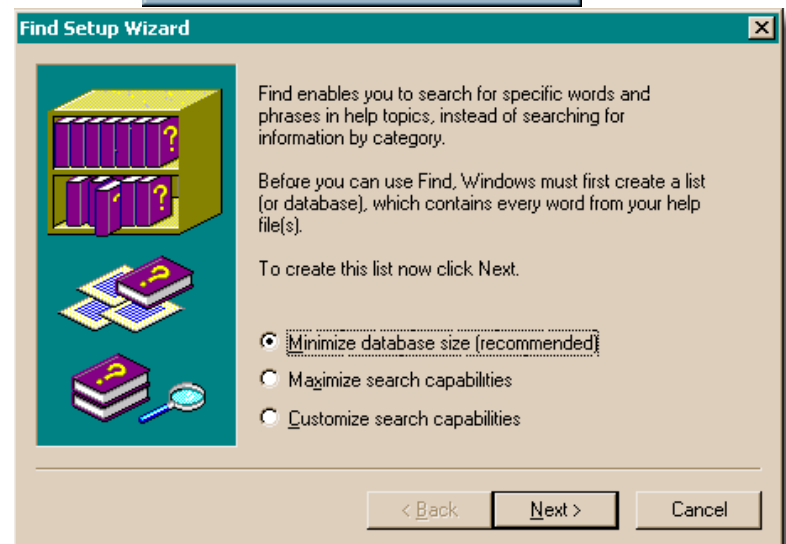
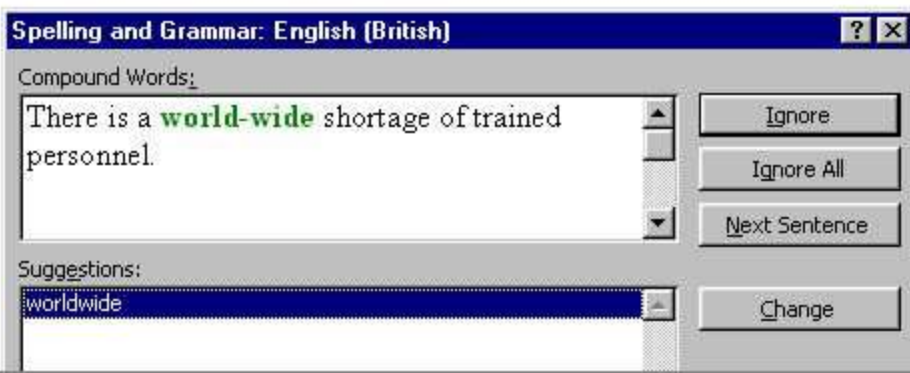
עוד דוגמאות והודעות שגיאה



There is a **worldwide** shortage of trained personnel.



There is a world-wide shortage of trained personnel.



D. Jackson , MIT

- TLV Lecture
<http://people.csail.mit.edu/dnj/talks/tel-aviv14/tel-aviv-sackler-anim-14.pdf>
- P. 35 OSx Alarm

שמישות ברשת (+300M\$)

Login

You must be a registered user to proceed. Please login or register below.

Already registered? Login

Username:

Password:

Login

☐ Remember my username on this machine.

[Forgot your login information? Click here.](#)

New to the Site?

Sign up in a few easy steps!

Register Now

Login

You must be a registered user to proceed. Please login or register below.

Already registered? Login

Username:

Password:

Login

☐ Remember my username on this machine.

[Forgot your login information? Click here.](#)

New to the Site?

Continue

You do not need to create an account to make purchases on our site. Simply click **Continue** to proceed to checkout.

To make your future purchases even faster, you can create an account during checkout.

se14b-yag

42

טעויות נפוצות בעיצוב רשת



- עמודים עמוסים
- התאמה לרזולוציות שונות
- התאמה לתקנים/דפדפנים שונים
- התאמה לאיזורים שונים
- "קירות" של טקסט
- עיצוב סטנדרטי

Jakob's Law of the Web User Experience states that "users spend most of their time on *other* websites."

דברים שלא כל המשתמשים יודעים

- הלוגו מקשר לעמוד הבית



- אבטחה מול פרטיות



NetBank® Banking how you live.
Member FDIC

Apply Now

Get Free Bill Pay & Pay No Fees
NetValue Interest Checking
Only \$50 deposit to open, free unlimited transactions. [Apply Now](#)

Navigation Menu:

- Banking
- Investments
- Loans
- About NetBank
- Demos
- Contact Us
- Home
- Security & Privacy
- Site Map

Loans Sub-menu:

- Overview
- Home Equity
- Mortgages
- Business Leasing
- Loan Glossary
- Loan Disclosures

Services	Our Rates
Money Market Account	4.75% APY* open
1 Year CDs (other terms available)	5.26% APY* open
Home Equity Loans (intro rate fixed for 3 months)	6.99% APR open

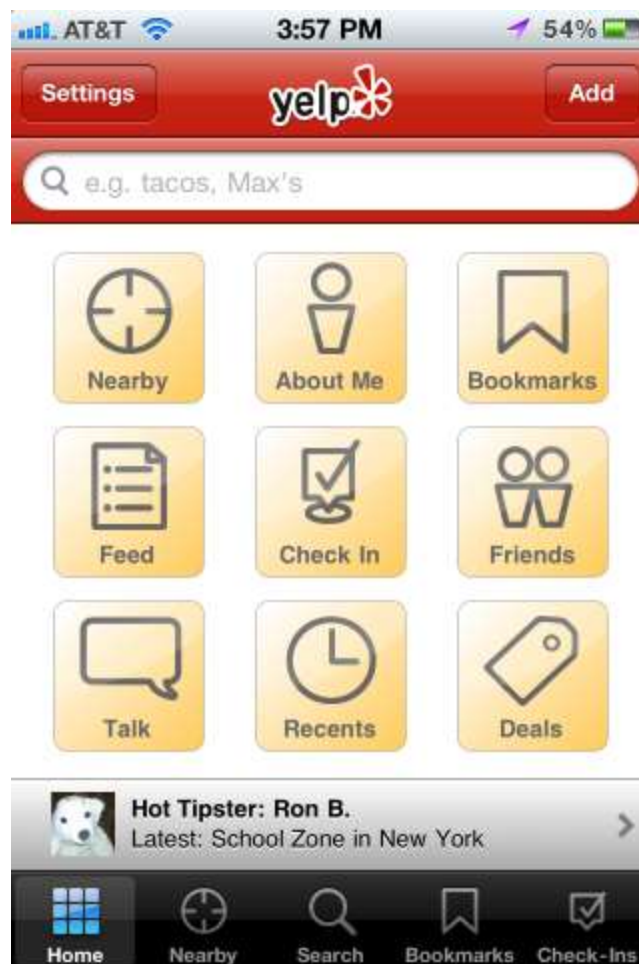
- שימוש בתפריטים נגללים
- כפתורי חיצים לרשימות
- ניווט באמצעות URL-ים
- הכרות עם כפתורי הדפדפן
- פתיחת חלון שני
- Web 2.0

דוגמת שמישות ברשת

- <http://graphs.gapminder.org/world/>
- => Hans Rosling's 200 Countries, 200 Years, 4 Minutes - The Joy of Stats

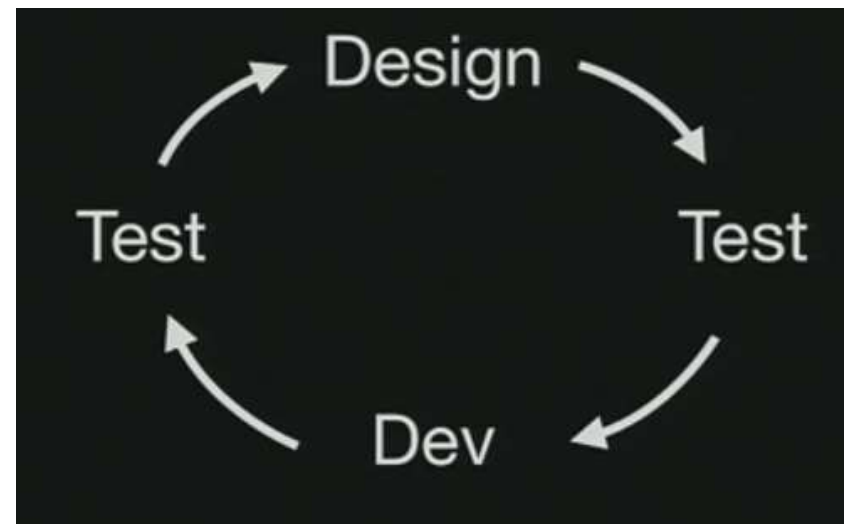


שמישות במחשוב נייד



עיצוב חווית משתמש

- Aral Balkan, “[A happy grain of sand](#)”, NDC 2012
 - Flush
 - Elevator
 - Washing Machine
 - DVD Burner @ Apple
 - Ticket Machine [44:10](#)
 - Phone network
- Bach, Exploratory testing of “[easy button](#)”



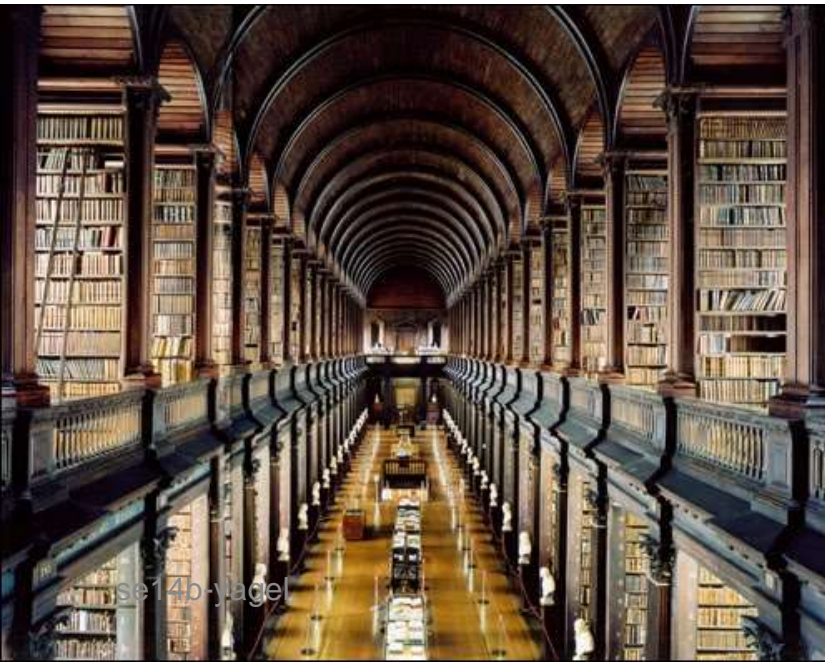
תרגיל עיצוב

- שרטטו ממשק משתמש לחיפוש בספרייה, הכולל את המאפיינים הבאים:

– אפשרות לחיפוש מחרוזת כמחבר, כותר או נושא

– אפשרות לחפש ספר או כתב-עת

– אפשרות לצפות במספר
תוצאות מסודרות ע"פ זמינות
או תאריך הוצאה
(אך לא שתי האפשרויות)



חווית משתמש ואנחנו

- ~~סקר שמישות עם נציג הלקוח במהלך הסבב~~
- ~~שיפורים בסבב הבא (אפשרות להתייחס בסקר עם יש קשר ישיר לקטע קוד)~~
- משוב (מקוון) מהמשתמשים, למשל:
 - A/B Testing
 - [UserVoice](#)
 - Google moderator, idea informer, ideaTorrent, etc...
 - במוצר שלכם?



סיכום - UX

- חווית משתמש
- מי מאפיין? מתי? כיצד בודקים?
- דרישות משתמשים ([The User is Drunk](#))
- עיצוב חווית משתמש כמקצוע
 - ארגונומיה, עיצוב גראפי, קוגניציה, ניסוח טקסט, [Gamification](#)
 - מתי באחריות מהנדס התוכנה?
 - לקחים ל: API Usability, או
- [Cambridge: Usability of Programming Languages](#)
- [MIT: User Interface Design & Implementation](#) קורס: בחירה,



תהליכים וכלים ||

• ראינו: תהליכים, שיטות וכלים (למשל: בדיקות יחידה)

• תהליכים

– שחרור

– תחזוקה

• כלים משלימים:

– כלי פיתוח משלימים

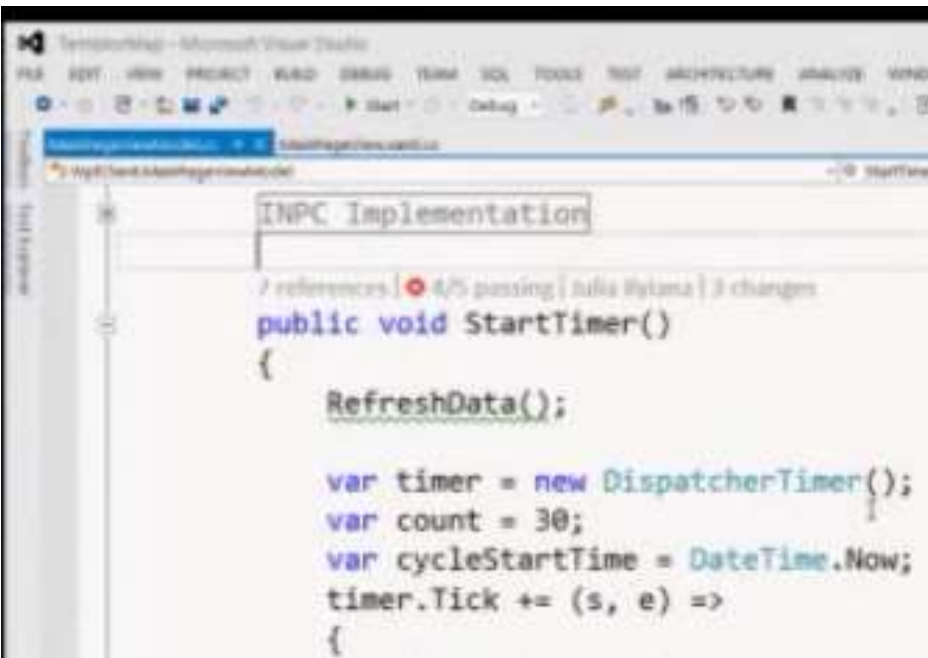
– סקרי קוד

– תיעוד

– בדיקות קבלה

– בדיקות כיסוי

– שילוב מתמשך והפצה מתמשכת



Links - Tools

- Beck&Feitelson, Development and Deployment at Facebook
<https://www.facebook.com/publications/514128035341603>
- G. Weinberger, Egoless Programming
 - אלעד סופר: [כלים אג'ילים – יש דבר כזה?](#)
 - משה קפלן: [סקרי קוד](#), [כלי הפיתוח שאתם פשוט חייבים](#)
- [Continuous Delivery: Anatomy of the Deployment Pipeline](#), Book chapter, 2010
- CI webcast
- Roy Osherov: Code Review [detailed post](#)

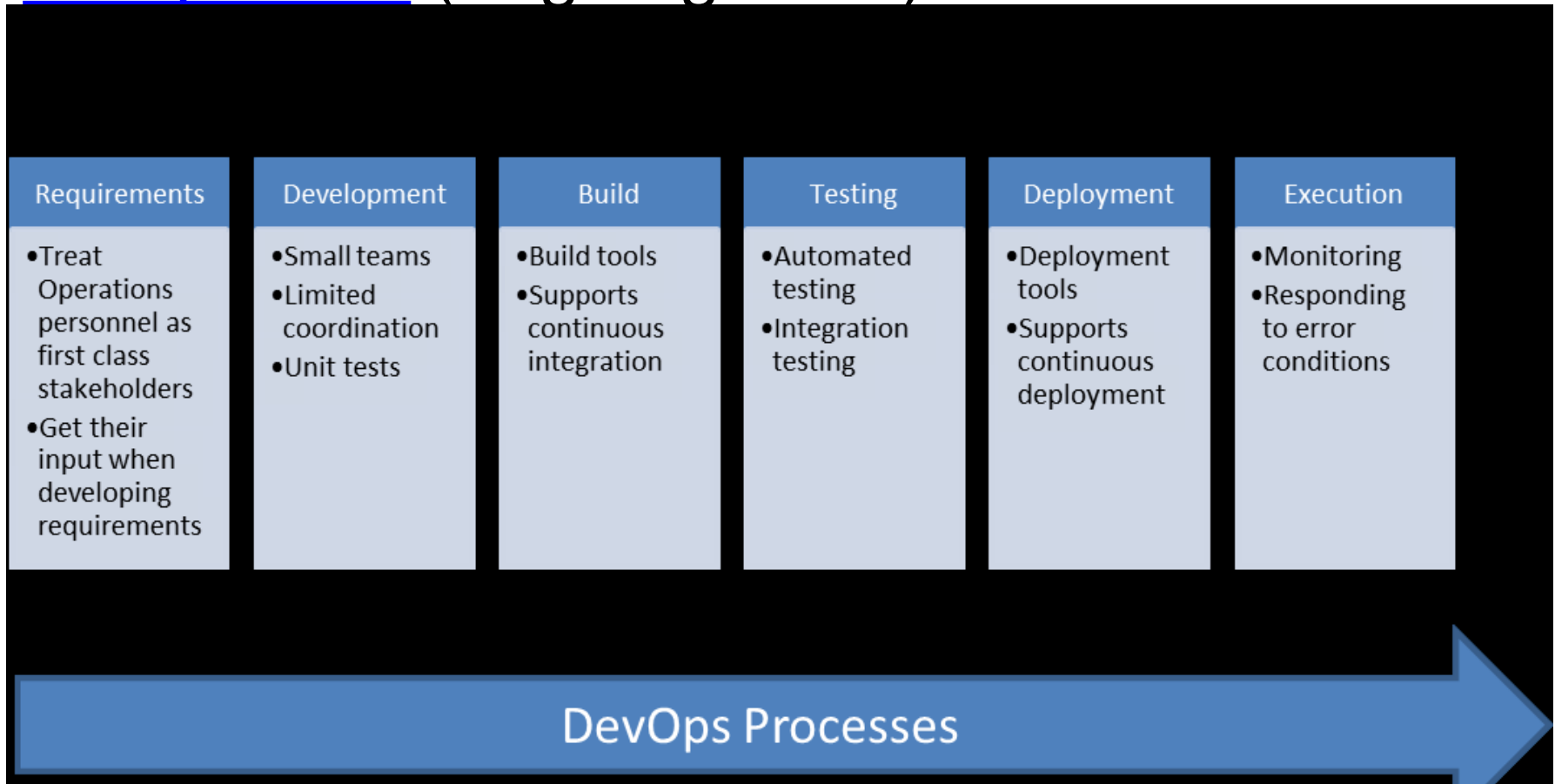
שחרור Release

- E.g., http://en.wikipedia.org/wiki/Deployment_Plan
 - Define and agree release and deployment plans with customers/stakeholders.
 - Ensure that each release package consists of a set of related assets and service components that are compatible with each other.
 - Ensure that integrity of a release package and its constituent components is maintained throughout the transition activities and recorded accurately in the configuration management system.
 - Ensure that all release and deployment packages can be tracked, installed, tested, verified, and/or uninstalled or backed out, if appropriate.
 - Ensure that change is managed during the release and deployment activities.
 - Record and manage deviations, risks, issues related to the new or changed service, and take necessary corrective action.
 - Ensure that there is knowledge transfer to enable the customers and users to optimise their use of the service to support their business activities.
 - Ensure that skills and knowledge are transferred to operations and support staff to enable them to effectively and efficiently deliver, support and maintain the service, according to required warranties and service levels

Agile: DevOps

- [Wiki](#): ..is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) operations professional
- “..is a new term describing what has also been called ‘**agile** system administration’ or ‘agile operations’ joined together with the values of agile collaboration between development and operations staff. Effectively, you can define DevOps as system administrators participating in an agile development process alongside developers and using a many of the same agile techniques for their systems work.”
<http://theagileadmin.com/what-is-devops/>
- => NoOps

Len Bass, [DevOps: A Software Architect's Perspective](#) (ongoing book)



0. כלי פיתוח נוספים

- Typing! (Yegge: non-touch-typists have to make sacrifices in order to sustain their productivity)

1. כלי פיתוח נוספים

- IDE/Compiler
 - Build tools, e.g. make, ant, maven
- Shared Development Env.
 - E.g. vagrant
 - Source control it!
- Static Analysis
 - Compiler warnings
 - E.g., [FindBugz](#) for Java

2. מהו סקר קוד (Code Review)?



- בחינה עקבית של הקוד

- במטרה למצוא באגים ולתקנם $2 < 1 + 1$

- שיפור הקוד ומיומנות המפתח

- האם אתם עושים זאת?

- האם אפשר לא לעשות

זאת?

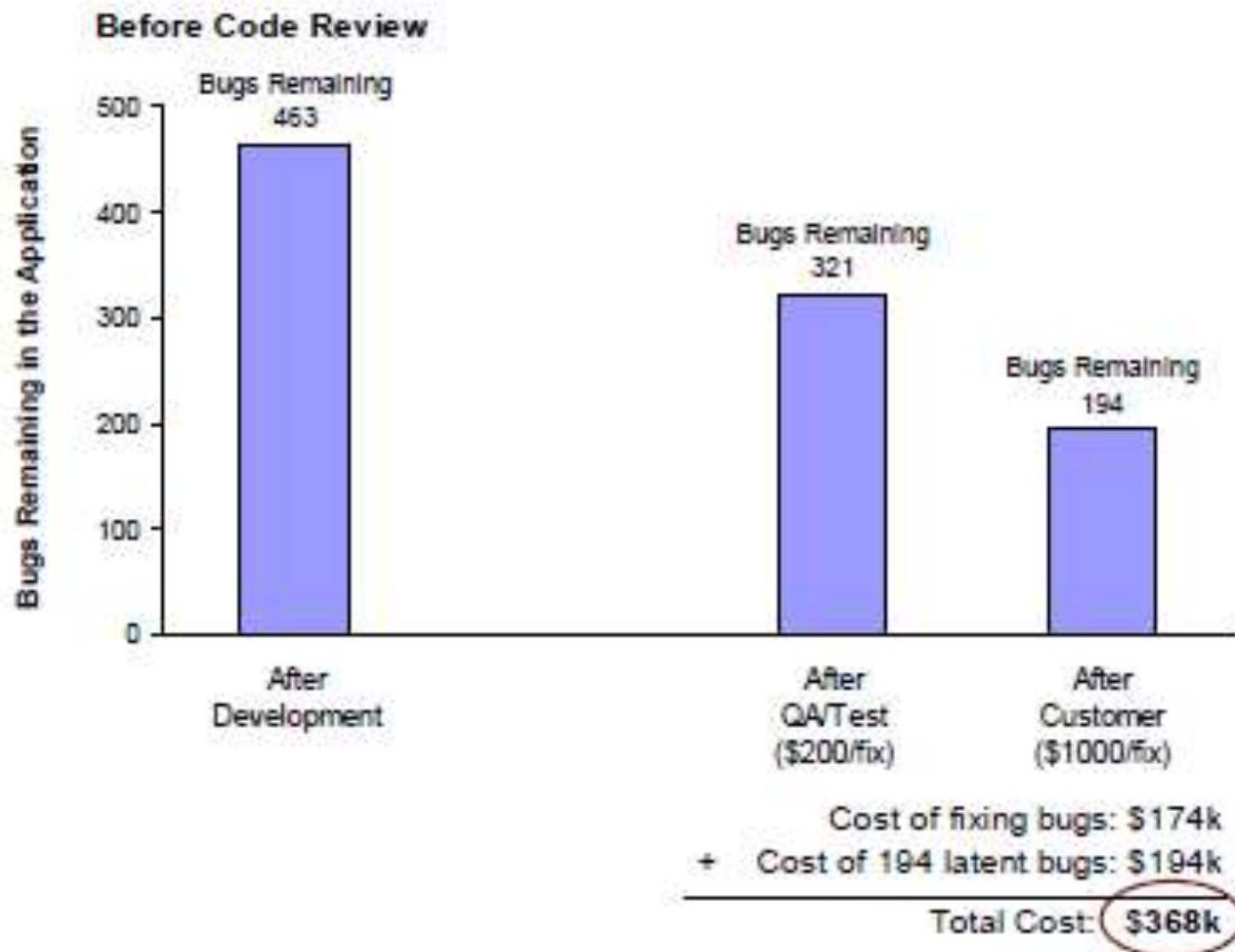
- “Developers at Khan Academy are expected to have all code reviewed”

- Atwood: “peer code reviews are the single biggest thing you can do to improve your code”

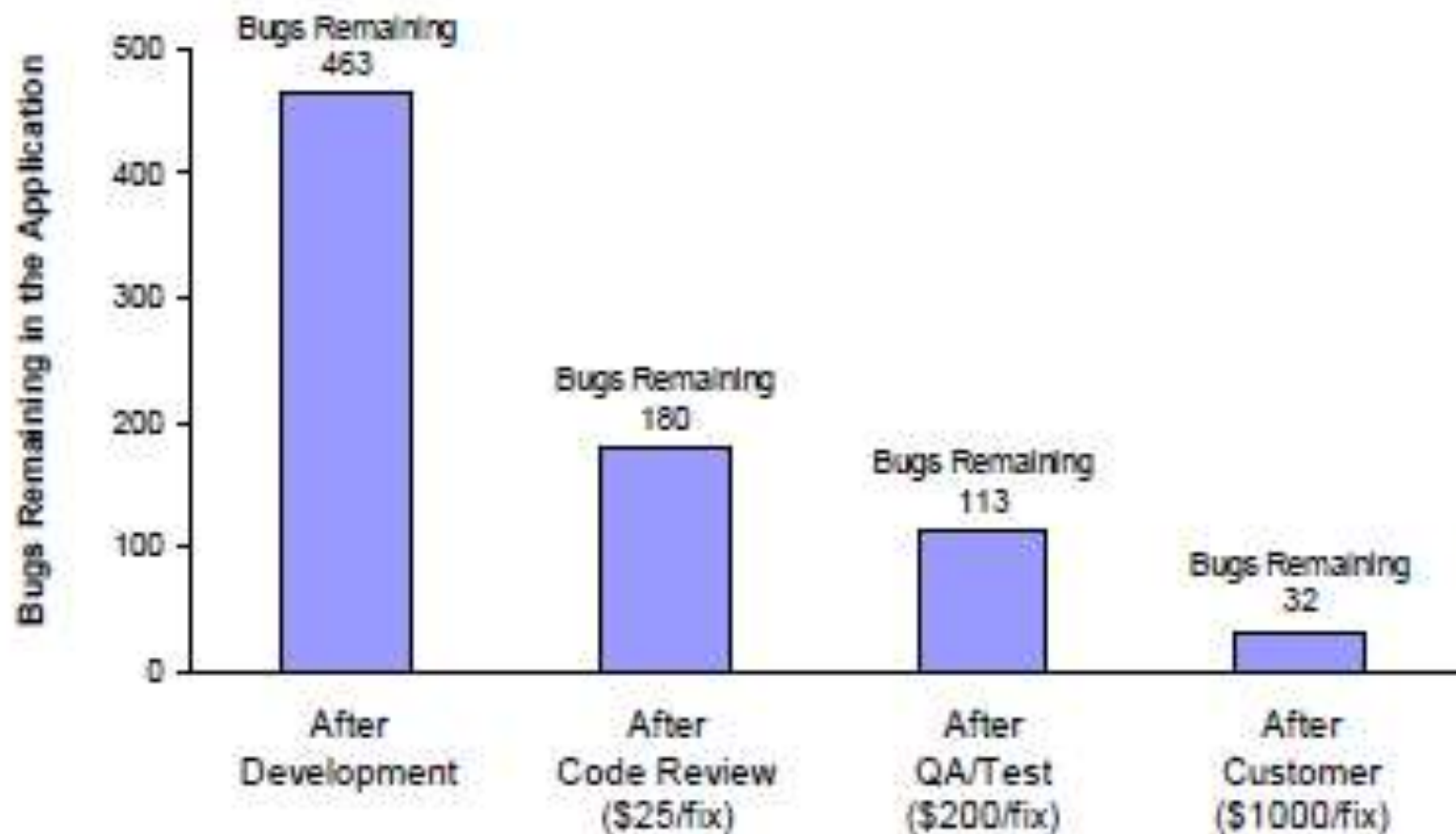
האם כדאי?

- Rigorous inspections can remove 60-90% of errors before the first test is run. (Fagan 1975)
- The first review and hour matter most. (Cohen 2006)

Saving \$150k: A real-world case study



After Code Review



Cost of fixing bugs: \$120k
+ Cost of 32 latent bugs: \$ 32k

Total Cost: **\$152k**

חסרונות?

- זמן מפתחים

– (אבל מצד שני יותר יקר לגלות בהמשך)

- קוד פתוח

– Linus's law: "given enough eyeballs,
all bugs are shallow"

– מי עוד רואה את הקוד?

שיטות וכלים

- מאחורי הכתף
- מייל, IM
- Pair-programming
- בשילוב עם TDD ו-Refactoring
- כלים ייעודיים
 - [Kiln](#)
 - [Rietveld](#) ([דוגמא](#))
 - [Mondrian: Code Review on the Web](#)
 - <http://www.review-board.org/>
 - <https://code.google.com/p/gerrit/>
 - JCE Course: <http://pair2program.appspot.com/>

GitHub Code Review

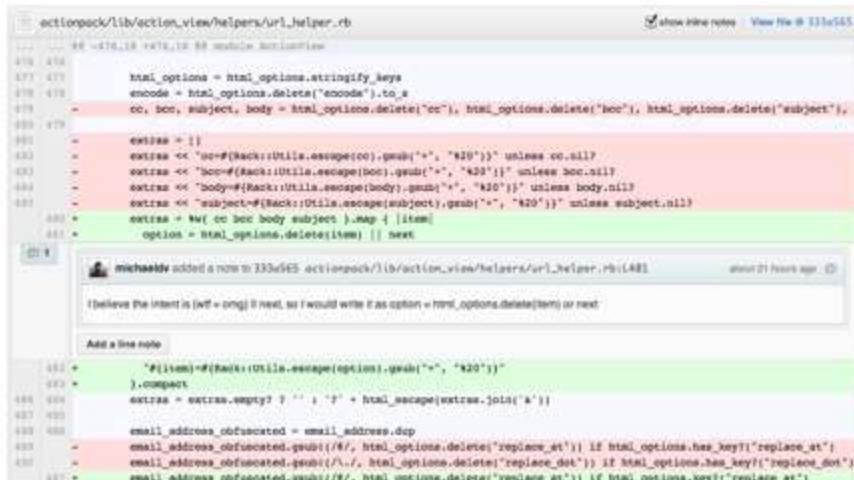
GitHub Pull Request = Code + Issue + •
Code Comments

Pull Requests •

Comments •

Process! •

Open Source Example •



```
476 476
477 477     html_options = html_options.stringify_keys
478 478     encode = html_options.delete("encode").to_s
479 479     cc, bcc, subject, body = html_options.delete("cc"), html_options.delete("bcc"), html_options.delete("subject"),
480 480
481 481     extras = []
482 482     extras << "cc=#{Rack::Utils.escape(cc).gsub(" ", "%20")}" unless cc.nil?
483 483     extras << "bcc=#{Rack::Utils.escape(bcc).gsub(" ", "%20")}" unless bcc.nil?
484 484     extras << "body=#{Rack::Utils.escape(body).gsub(" ", "%20")}" unless body.nil?
485 485     extras << "subject=#{Rack::Utils.escape(subject).gsub(" ", "%20")}" unless subject.nil?
486 486
487 487     extras = %w(cc bcc body subject).map { |item|
488 488         option = html_options.delete(item) || next
489 489
490 490     }
491 491
492 492     #item=#{Rack::Utils.escape(option).gsub(" ", "%20")}"
493 493     }.compact
494 494     extras = extras.empty? ? "" : " " + html_escape(extras.join("&"))
495 495
496 496     email_address_or_fascinated = email_address.dup
497 497     email_address_or_fascinated.gsub!(/%, html_options.delete("replace_at") if html_options.has_key?("replace_at")
498 498     email_address_or_fascinated.gsub!(/./, html_options.delete("replace_dot") if html_options.has_key?("replace_dot")
499 499     email_address_or_fascinated.gsub!(/%, html_options.delete("replace_at") if html_options.has_key?("replace_at")
```

הנשק הסודי?

- של Extreme Programming – [Pairing vs. Code Review](#) דיון
- של חברות שמצליחות להחזיק מוצרים לאורך זמן
- בפרויקט שלכם: תיעוד שליחת diff והערות שהתקבלו



Richie Rump
@Joriss



RT @ashalynd If the programmers like each other, they play a game called "pair programming". And if not, then the game is called "peer review"

9:13 PM - 4 May 2013

3. תיעוד ומסמכים

```
int five = 7; //11
```

- למה\האם לכתוב תיעוד? מי? מתי? סוגים?
- הערות: שנוי במחלוקת, למשל
<http://apdevblog.com/comments-in-code/>
Stackoverflow [thread](#), [Do not comment code](#) 😊
- כלים ליצירת תיעוד מהערות
[Javadoc](#)/Ndoc, Doxygen, Sandcastle, [docco](#), [ועוד](#)
– למשל [String in Java](#)
- מתי כלים אלו נצרכים?
- איזה סוג תיעוד נותנות בדיקות יחידה? (CSVReader [דוגמא](#))
- דיון: [Agile Documentation](#), Amber,

4. כלים ותהליכים תומכי בדיקות

- Unit Testing Frameworks

 - ✓ xUnit

 - Parameterized tests, White-box, fuzzing (security, [Heartbleed bug](#)), GUI

- Acceptance Tests Frameworks

- Code Coverage

- Continuous Integration

- A/B Testing ([paper](#))

```
בכמה מקרים תתרחש שגיאה?:  
int foo(int x) { // x is an input  
    int y = x + 3;  
    if (y == 13) abort(); // error  
    return 0;  
}
```

בדיקות קבלה (הזכרנו)

- שמות שונים: בדיקות משתמש, בדיקות קצה, פונקציונליות, מפרטים מורצים, BDD, **A**TDD
- סביבות מתפתחות לאחרונה, למשל
Fit, [Cucumber](#), SpecFlow, [BDDify](#), [TextTest](#) –
Web Automation: Selinium, Watir –
– <http://swarm.jquery.org/> סטטוס מעודכן לדוג'

כיסוי קוד ע"י בדיקות

- כמה בדיקות צריך לכתוב?
- ב-TDD לא כותבים אף שורה בלי בדיקה
 ⇐ 100% כיסוי
 ⇐ אם יש 90% כיסוי, היכן מסתבר שרוב הבאגים?
- כלים לדוגמא:
 - .Net : Ncover (משולב ב- TestDriven.Net)
 - Eclipse : [EclEmma](#)
 - ~~נבדוק עוד דוגמא [CsvReader](#)~~



5. שילוב מתמשך – CI/CD

- מתי לבנות הכל ולהריץ את הבדיקות?
- באופן אידיאלי: לפני ואחרי כל שינוי
- לפחות כל פעם שמכנסים משהו חדש לבקרת הגרסאות?
 - למה זה חשוב
- מי מריץ?
- שרתים: למשל Jenkins, Cruise Control, TeamCity ([on ec2](#))
- ענן: Heroku, Cloudbee, MS Azure, jelastic
- שרות: travis-ci, [drone.io](#), [codebetter](#), [codeship.io](#)
- כלי בנייה \ אוטומציה כגון: Make, MsBuild, Ant, Kayak
- <http://martinfowler.com/articles/continuousIntegration.html>
- <http://martinfowler.com/bliki/ContinuousDelivery.html>

Continuous integration - Recommended practices

http://en.wikipedia.org/wiki/Continuous_integration#Recommended_practices

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits every day
- Every commit (to mainline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

יתרונות וחסרונות

- איתור באגים קל ומהיר
- לא נופלים על בעיות שילובים בסוף
- עותק זמין ועובד של המוצר כל הזמן
- משוב תמידי על איכות הקוד
- מעודד כתיבת קוד מודולרי ופחות תלותי
- מצריך הקמת תשתיות בהתחלה
- כדי להיות אפקטיבי מצריך פיתוח סט בדיקות

בפרויקט - בונים

- דרישות נינימליות:

- בקרת גרסאות

- הידור

- בדיקות

- הודעות

- (Tray Application)

- שרת, למשל [Jenkins](#)

- שרות: Travis-ci, CircleCI

- github / heroku hooks

- הצגה בשבוע הבא



הפצה מתמשכת - CD

- פרויקט תוכנה כפס ייצור
- לאילו פרויקטים מתאים? האם עדיין יש צורך באיטרציות? (דוגמא [github](https://github.com))
- תשתית CI כשלב מקדים
- תמיכה בשלה התחזוקה
 - מדרגיות (CloudFoundry)
 - ניטור (newrelic)

לסיכום – תהליכים וכלים



- השלבים הנשכחים אך...
- צידה לדרך
 - סקרי קוד, תיעוד
 - בדיקות מעבר ליחידה, כיסוי
 - שילוב מתמשך
- כלים רבים נוספים, למשל
 - אכיפת סגנון FxCop
- מהו המינון הנכון?...



בפעם הבאה

- סיכום הקורס

- [סקב](#) קורס, פרויקט, שיטות, תהליכים וכלים

- מבנה הבחינה

- מצגות סיום

- הצגה צוותית: מוצר + רטרוספקטיבה

- כל אחד מספר על חלקו (משימה אישית 5) ויכול להישאל על כך

- בונוס CI