אוניברסיטת
אריאל
בשומרון

# הנדסת תוכנה

## איפיון, עיצוב ופיתוח פתרונות תוכנה

מירב שקרון, meravgu@gmail.com
יוסי זגורי, yossiza@ariel.ac.il , 052-4668866

# מבנה וחובות הקורס

- מבנה הקורס –
  - שיעורים (3 ש"ש) ותירגולים (1 ש"ש)
  - מרצים- מירב שקרון, ד"ר יוסי זגורי, ספיר אסרף
  - מתרגלים- ספיר אסרף, איתן בר כוכבא
- חובות הקורס-
  - מבחן 40% מהציון
  - פרויקט 60% מהציון
- **יש לעבור את המבחן בציון 60 לפחות בשביל לקבל את הציון על הפרויקט**

# פרויקט

- הפרויקט מחולק לכמה חלקים אותם צריך להגיש במהלך הסמסטר.
- את המטלות והפרויקט מגישים בקבוצות עבודה של **3-4** אנשים (הנשארים קבועים לאורך כל הסמסטר).
- בשיעורים האחרונים בקורס תתקיימנה הצגות של כל הפרויקטים לכל שאר חברי הכיתה ולמרצים
  - יש להכין **מצגת** המתארת את כל תהליך העבודה
  - כל חברי הקבוצה חייבים להציג במצגת ולהיות יכולים לענות על שאלות של המרצה/ הסטודנטים האחרים
  - את הקוד של הפרויקט יש להגיש לבדיקה למתרגלים (ב-GitHub)
  - סטודנטים המגישים פרויקט צריכים להיות נוכחים לאורך כל השיעור שבו הם מגישים.
- את המטלות יש להגיש דרך המודל (אין דחיות)
- נושא הפרויקט – אתם בוחרים את הנושא שמעניין אתכם! (**בכפוף לאישור של המתרגל**)
- הפרויקט ידמה את תהליך פיתוח מערכת כפי שמתרחשת בתעשייה

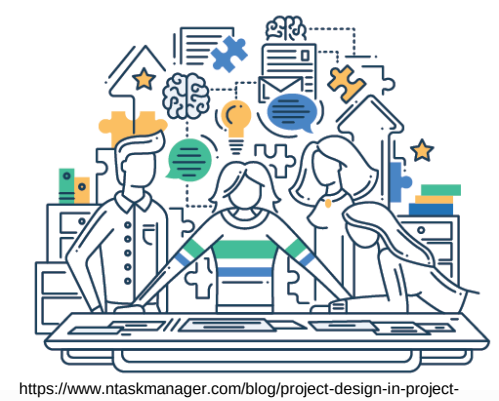
# ועדיין...

הפרויקט לא זהה לפרויקט בתעשייה:

- גודל הפרויקט (תכולה, אורך)
- עבודת צוות
- בעיות ותהליכים מסובכים
- אין לקוח
- דרישות סותרות ומשתנות
- שינויים לאחר ותוך כדי הפיתוח
- עלות תקלות
- ראיה מערכתית

# מטרות הקורס

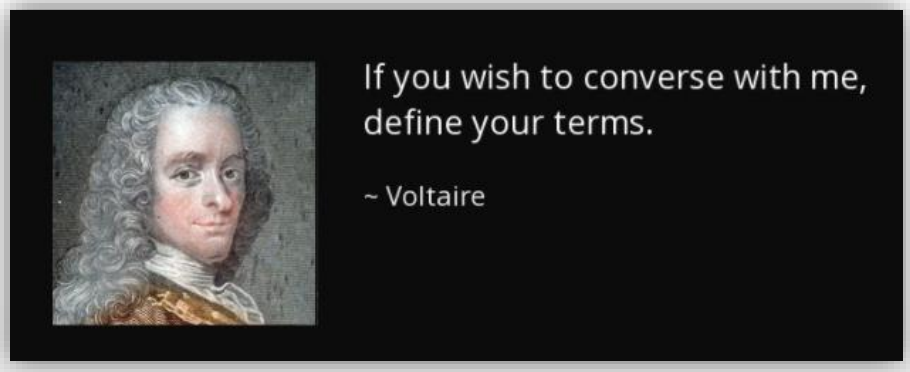
- הכרת מושגים, תהליכים ופעילויות פיתוח תוכנה
  - מתודות וגישות לניהול הפיתוח
  - System Analysis
  - Software Design
  - Programming Patterns & Practices
- הקניית יכולות ישומיות:
  - ייצוג מודלים באמצעות שפת UML
  - שימוש בעקרונות ותבניות תכן לעיצוב וכתיבת קוד "נכון"
  - רתימת כלים לניהול ומימוש שלבי פרויקט
  - פיתוח אפליקצית אנדרואיד



https://www.ntaskmanager.com/blog/project-design-in-project-management/

# מושגי יסוד והגדרות

- מהי מערכת
- מהו מודל ומהו תכנון מודולרי
- מהי תוכנה ומהי הנדסה
- הנדסת תוכנה ומאפייניה
- בעלי עניין
- כשלים בתוכנה- דוגמאות, סיבות ופתרונות
- פעילויות בפיתוח תוכנה
- סוגי תוצרים (ארטיפקטים)

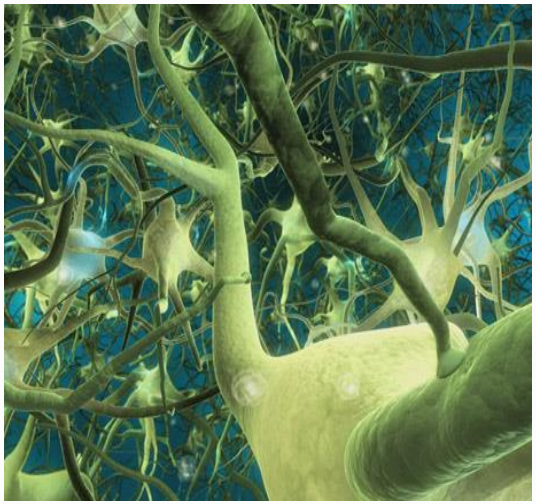


אוניברסיטת
אריאל
בשומרון

# מערכות , מודלים והנדסה

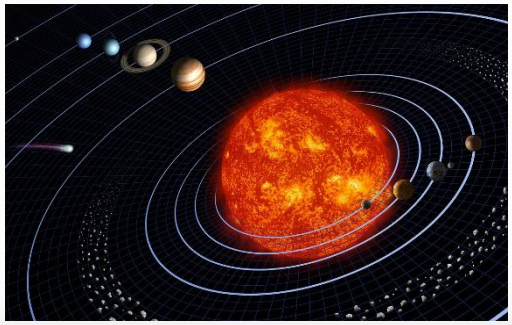
# מערכת

אוסף ישויות ממשיות או מופשטות, המקיימות קשרי גומלין או תלות הדדית מתמשכת ויחדיו יוצרות שלם המשרת מטרה משותפת.



מערכת העצבים

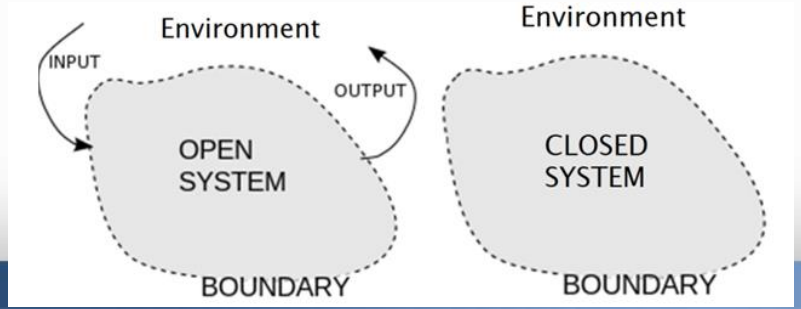
מערכת המשפט



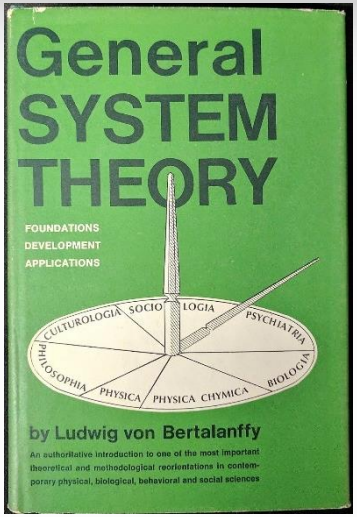
מערכת השמש

# תכונות מערכת

- למערכת יש *מטרה* אחת או יותר.
- למערכת *גבולות* המבחינים אותה *מהסביבה.*
- למערכת *מבנה* המוגדר על ידי מרכיביה והיחסים המבניים בינם לבין עצמם.
- למרכיבי המערכת יחסים פונקציונליים המגדירים *תהליכים*.
- למערכת *התנהגות* אופיינית הכוללת *קלטים* ( אנרגיה/ חומר / מידע ) ו*פלטים* שהם תוצר עיבוד המשנה את הקלט בצורה כלשהיא.



# General Systems Theory

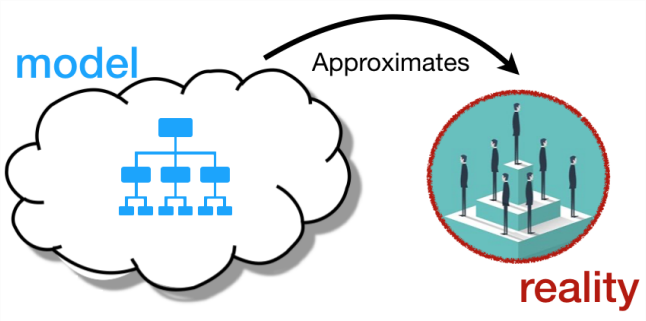



- שדה מחקר מדעי בין תחומי העוסק בהבנת אופיין של מערכות מורכבות באשר הן.
- עקרונות ותפיסות מתחום האונטולוגיה, פיסיקה, ביולוגיה, מדעי המחשב, הנדסה, סוציולוגיה, פסיכולוגיה וכלכלה
  - מערכת היא ישות השומרת על קיומה על ידי יחסי גומלין ופעולות הדדיות בין מרכיביה.
  - Emergence - איפשהו בין האינטראקציות ההדדיות צצות תכונות שלא ניתן למצוא אותן בנפרד : חקר המרכיבים בנפרד אינו מספיק להבנת השלם == השלם גדול מסך חלקיו.
  - אינטראקציות אלו כוללות לולאות ומשובים המייצרים התנהגות לא לינארית ולא דטרמיסינטית המקשה על חיזוי התנהגותה.

Ludwig von Bertalanffy (1968). *General System Theory: Foundations, Development, Applications*

# מודל

- ייצוג מופשט של תופעה, ישות או רעיון.
- הפשטה (אבסטרקציה)
  - השמטת פרטים בכדי ליצור מושגים וכללים עקרוניים.
- סוגי מודלים:
  - פיזי – דגם מטוס
  - קונספטואלי – מודל סטטיסטי
  - ועוד...
- בונים מודלים כדי:
  - לנתח, להבין, לבחון, ללמוד, לבנות...




- מודלים עומדים בבסיס העשייה בתחומים טכנולוגיים והנדסיים (ולא רק בהם).
- למודלים המייצגים אספקטים שונים בתהליך הפיתוח מוגדרים ארטיפקטים סטנדרטיים.

# תוכנה

אוסף שירותים המקובצים על גבי חומרת מחשב ומיועדים למימוש מטרות משתמש

# תצורות מוצר תוכנה

## 1. מוצר עצמאי

– מגיע בדיסק קשיח או בהורדה מהאינטרנט מיועד לשימוש על מחשב המשתמש ומספק מערכת הפעלה ותוכנה תשתיתית

## 2. מוצר משובץ (embedded)

– מגיע ביחד עם החומרה המתאימה לו ספציפית  יש בו מערכת הפעלה ותוכנה תשתית כמוצר שלם

## 3. תוכנה כשירות (SaaS= Software as a Service)

– התוכנה עצמה אינה מגיעה ללקוח ואינה נשארת ברשותו אלא הוא מקבל שרותים הדרושים באמצעותה.

# מה זה תוכנה טובה?

הלקוחות מרוצים

מבצע מה שאמור לבצע

קוד עובד

ממשק נוח

קוד קל לקריאה

עיצוב גרפי מוצלח

עיצוב תוכנה טוב

קוד נקי

המשתמשים מרוצים

# הנדסה

- החלת שיטות ועקרונות מדעיים ליצירת טכנולוגיה ומוצרים.
- דוגמאות-
  - הנדסת מכונות
  - הנדסה אזרחית
  - הנדסת חשמל
- דיסציפלינות בנות אלפי שנים

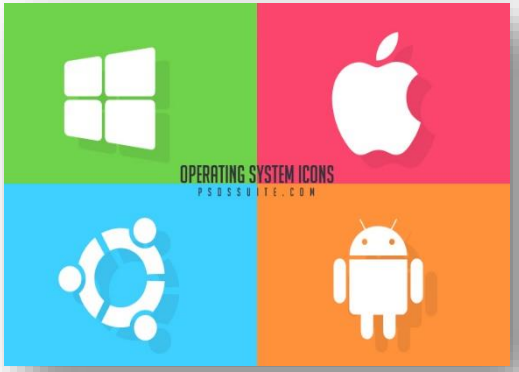
אוניברסיטת
אריאל
בשומרון

# הנדסת תוכנה

# מהי הנדסת תוכנה

אוסף שלבים, פעילויות, תפקידים ותוצרים המשרתים יצירת מוצר המורכב משירותי תוכנה במגוון "אריזות"

מערכות הפעלה

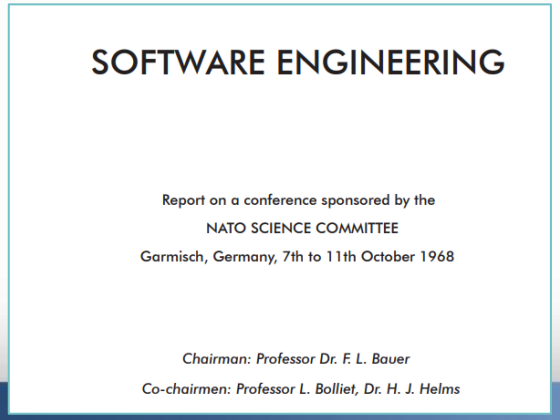



אפליקציות

ספריות ותשתיות פיתוח

# הנדסת תוכנה

- המושג הוטבע לראשונה בכנס נאס"א 1968.
- בנוסף לבסיס מדעי מדובר על החלה שיטתית של שיטות/ טכניקות/ שפות/ כלים ותהליכים בבניית מערכות תוכנה.
- מטרות: יצירת תוכנה איכותית, בלו"ז ובתקציב, העונה לציפיות הלקוחות וצורכי המשתמשים.
- מאפיינים:
  - מורכבות הדורשת עבודת צוות ממספר דיסיפלינות.
  - אינטראקציה ותקשורת בין מגוון בעלי מקצוע
  - דינאמיות וחדשנות בכל אספקט אפשרי


SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

*Chairman: Professor Dr. F. L. Bauer*
*Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms*

# ציטוטים מפורסמים מהכנס

- "*We build systems like the Wright brothers built airplanes — build the whole thing, push it off the cliff, let it crash, and start over again".*

- *"Production of large software has become a scare item for management. By reputation it is often an unprofitable morass, costly and unending. This reputation is perhaps deserved".*

- *"Programming management will continue to deserve its current poor reputation for cost and schedule effectiveness until such time as a more complete understanding of the program design process is achieved".*

# יצירת תוכנה: תהליך הנדסי או אומנותי ?

- בהנדסה המטרה היא להקטין את השונות- בתהליך/ במוצר/ בתקציב/ בידע/ במשאבים
- באומנות השונות רצויה ומבורכת.

- הנדסת תוכנה היא שילוב של שני הדברים-
  - שונות גבוהה בפרויקטים- באופי, בתקציב, בלו"ז...

    יחד עם זאת-
  - שאיפה להקטין את השונות בתהליך עצמו

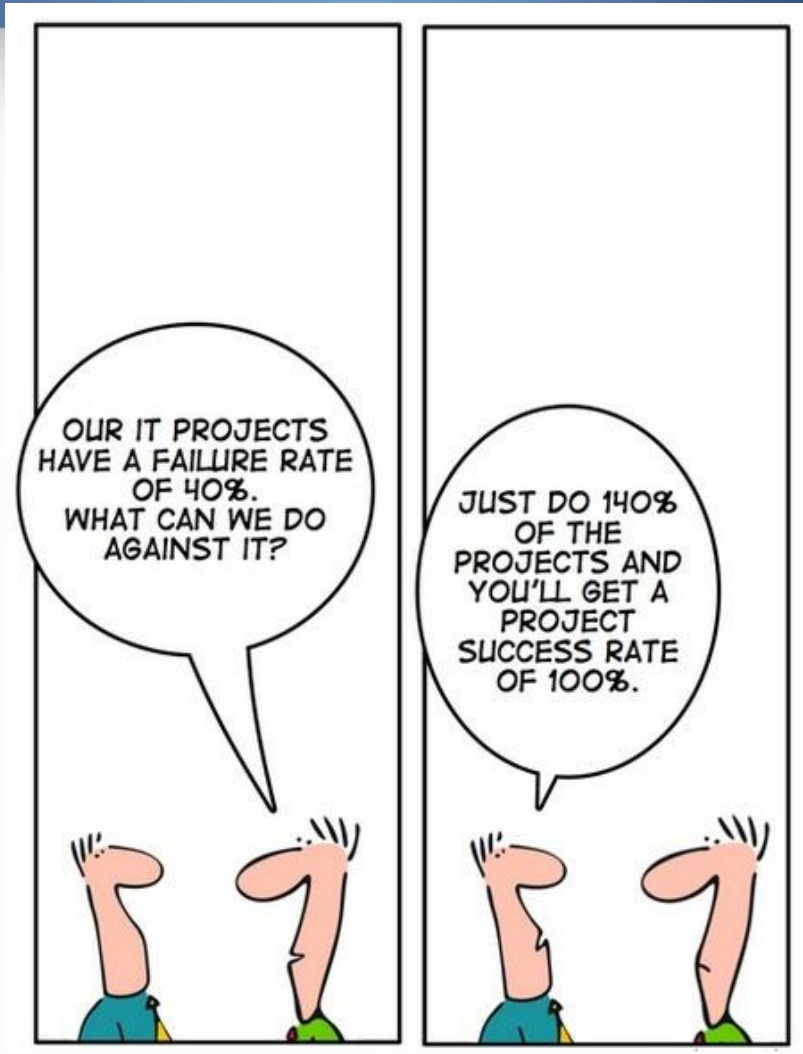
# בעלי עניין בפיתוח התוכנה

- מנתח מערכת
- ארכיטקט
- תוכניתן
- בודק
- מטמיע
- לקוח (מומחה יישום)
- ....

# האם פרויקטי תוכנה מצליחים?

| | 2011 | 2012 | 2013 | 2014 | 2015 |
|---|---|---|---|---|---|
| SUCCESSFUL | 29% | 27% | 31% | 28% | 29% |
| CHALLENGED | 49% | 56% | 50% | 55% | 52% |
| FAILED | 22% | 17% | 19% | 17% | 19% |

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf


OUR IT PROJECTS HAVE A FAILURE RATE OF 40%. WHAT CAN WE DO AGAINST IT?
JUST DO 140% OF THE PROJECTS AND YOU'LL GET A PROJECT SUCCESS RATE OF 100%.

# דוגמאות לכשלים בתוכנה

## שער דולר נקבע על 0, מערכת האשראי קרסה

עשרות לקוחות דיווחו כי אינם מצליחים לבצע עסקאות. רמי לוי: הקונים השאירו עגלות. מקור הבעיה היה במונופול הבנקאי שב"א: בקובץ הנתונים היתה טעות בשער הדולר ונרשם שערכו אפס. השב"כ: מסתמן שלא מדובר במתקפת סייבר. כעבור מספר שעות תוקנה התקלה

http://www.ynet.co.il/articles/0,7340,L-4487921,00.html

# Samsung Galaxy Note 7

*Samsung Galaxy Note 7 after catching fire (Image: Ariel Gonzalez / YouTube)*

# פריימריז במפלגת העבודה

הבוקר נפתחו אתרי ההצבעה כרגיל ,אולם זמן קצר לאחר פתיחת ההצבעה, החלו לזרום למפלגה דיווחים על תקלות במערכת המחשוב ועל תורים ארוכים של מצביעים שביקשו להצביע. גם המתמודדים נתקלו בקשיי הצבעה. כך למשל, המועמדת עינת וילף, שהיתה אמורה להצביע בקלפי בתל אביב, נאלצה לנוכח התקלה במערכת המחשוב לנסוע לעמדת ההצבעה בבת ים, אולם משהתברר כי גם שם לא פועלת המערכת, עברה וילף לראש העין, וגם שם ההצבעה לא התאפשרה. רק כשהגיעה לפתח תקווה הצליחה המתמודדת לממש את זכותה. השר שלום שמחון נאלץ להמתין מ-10:00 בבוקר ועד 13:00 בצהריים בקלפי שביישובו ולא הצליח להצביע. תקלה נוספת שהתגלתה היא כי במערכות מסוימות לא ניתן היה לסמן את שמו של השר בנימין בן אליעזר, ששמו נוסף באיחור לרשימת המתמודדים לאחר שבוטל שריונו.

ההחלטה התקבלה לאחר שמרבית בכירי המפלגה הגישו עתירה לביטול הפריימריז כיוון שלטענתם בוחרים רבים נטשו את הקלפיות. המתמודדים התריעו כי אם לא תבוטל ההצבעה, יגישו ערעורים לוועדת הבחירות של המפלגה באשר לתוצאות.

בקרב המתמודדים והמתפקדים נשמעו היום תלונות כי לא נעשו בדיקות מקדימות למערכת הממוחשבת, שזה השימוש הראשון בה. גם במפלגה זעמו על התקלות והפנו אצבע מאשימה לחברת טלדור, האחראית על המערכת. "זו מיני הונאה - הבטחנו לכולם שהכל יילך מהיר יותר", אמר דובר המפלגה, ליאור רוטברט. טלדור אמרו בתגובה כי לצערם נפלה תקלה במערכת וכי הם יעשו כל מאמץ לסייע בבחירות בשבוע הבא.

לאחר הפארסה היום הוחלט בעבודה לזנוח את המודרניזציה ולחזור לשיטת ההצבעה הידנית.

http://www.globes.co.il/news/article.aspx?did=1000403480

# Software Fails Watch 5th Edition

חברה המוציאה דוח מפורט של באגים ותקלות במערכות תוכנה, עם פילוח לפי שנים, לפי סוגי מערכות ועוד.

החברה בחנה 314 חברות, 1177 סיפורים ו-606 תקלות תוכנה.

מסקנות כלליות:

- כמעט מחצית מאוכלוסיית העולם הושפעה מהתקלות במהלך שנת 2017 (3.7 מתוך 7.4 מיליארד)
- שיאיי התקלות התרחשו בין החודשים אוגוסט לנובמבר
- אומדן הפסדים מתקלות התוכנה: 1,715,430,778,504 דולר!!

https://www.tricentis.com/wp-content/uploads/2019/01/Software-Fails-Watch-5th-edition.pdf

# כמה דוגמאות

המשטרה עצרה בשבוע שעבר פלסטיני בחשד שתכנן לבצע פיגוע דריסה, על בסיס תרגום שגוי של פוסט בערבית שפרסם בפייסבוק. המשטרה הסתמכה על תרגום אוטומטי של ההודעה, שכלל לא נקראה בידי דובר ערבית לפני מעצרו של החשוד.

לפני שבוע העלה הפלסטיני לעמוד הפייסבוק שלו תמונה שבה הוא נראה נשען על דחפור באתר בנייה שבו הוא מועסק, בהתנחלות ביתר עילית. לצד התמונה הופיע כיתוב בערבית, שפירושו "בוקר טוב לכם". ואולם, התרגום האוטומטי שמציע פייסבוק המיר את הברכה במלים "לפגוע בהם" בעברית, ו-"hurt them" באנגלית. בשל כך איתרה משטרת מחוז ש"י את הפוסט ועצרה את כותבו.

## SCARIEST

**Noida fire: Automatic doors trapped six persons, says Fire dept**

Six people tragically died in an office fire after being trapped between sets of automatic doors. The doors were not built with a manual fail-safe in case of emergencies, so when the electricity cut out, the software shut down—locking the doors and the unfortunate victims inside.

# ועוד ועוד...

- https://www5.in.tum.de/~huckle/bugse.html
- http://www.cs.tau.ac.il/~nachumd/verify/horror.html

# הסיבות לכשלים בפרויקטי תוכנה

- הגורם האנושי
  - תקשורת לקויה (לקוח-ספק, הנהלה-צוות פיתוח, בעלי תפקידים בצוות)
  - ניהול דוגמטי חסר מעוף, יצירתיות והבנת התחום
    - תהליכים לא מתאימים לסיטואציה ולמשאבים
  - אינטרסים מנוגדים של בעלי עניין (Stakeholders)
  - אי שימוש בסטנדרטים ושיטות עבודה מוגדרות היטב
  - אי שימוש במדדי תוכנה לקבלת החלטות על בסיס כמותי
  - אי שילוב תהליכי בקרת איכות
  - חוסר מקצועיות צוותים (ארכיטקטים, תוכניתנים, מנהלי מוצר)
  - ...

- טבעו של עולם
  - תהליכים רווי אי ודאות מובנית
  - מורכבות, מורכבות, מורכבות (תהליכים, כלים, פתרונות....)
  - דרישות משתנות ואי יציבות בהגדרת הצרכים
  - לחצי זמנים ותקציב
  - ...

# No silver bullet

Brooks, F., and H. Kugler. *No silver bullet*. April, 1987.

- קשה לפתח תוכנה-
  - אין טכנולוגיה אחת או טכניקת ניהול אחת שאיתה בוודאות ניתן לקבל שיפור בפרודוקטיביות (בניגוד לחומרה שהתפתחה בצורה מאד משמעותית)
- מורכבות מהותית ומקרית-
  - מורכבות מקרית ניתנת לצימצום
  - מורכבות מהותית נובעת מעצם הבעיה

# תכונות מהותיות בתוכנה

- Complexity- סיבוכיות
- Conformity- קוֹנְפוֹרְמִיוּת, התנהגות בהתאם, תאימות
- Changeability- ניתן לשינוי
- Invisibility- בלתי נראה

# סיבוכיות


מעקב אחרי מסלול של אדם על גבי מפה.

נניח שיש 20 תחנות במסלול. אם נרצה לעקוב אחרי כל האפשרויות לעבור בתחנות. מספר האפשרויות הוא 20!

כל הוספת תחנה מסבכת מאוד את המבנה

# תאימות

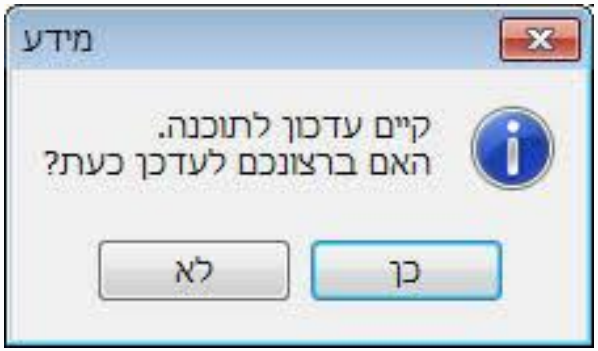
- שינויים והוספת יכולות בחומרה משפיעות על התוכנה.
- תוספת מהירות ודיוק
- התגברות על מגבלות פיזיקליות – למשל: מצלמה
- הוספה או שינוי של יכולות למשל: שליחת סרטים
- תיקון בעיות עבר

# יכולת שינוי

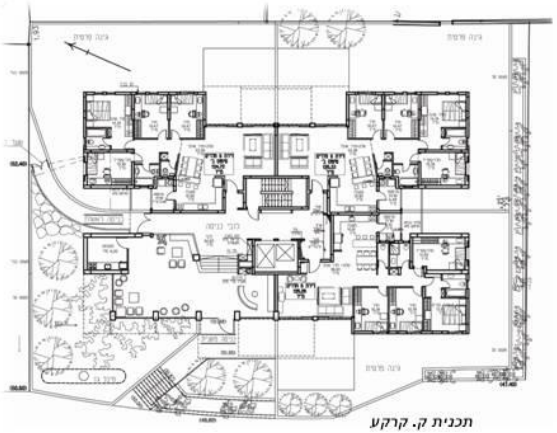
| Code name | Version number | Linux kernel version[2] | Initial release date | API level |
|---|---|---|---|---|
| (No codename)[3] | 1.0 | ? | September 23, 2008 | 1 |
| Petit Four[3] | 1.1 | 2.6 | February 9, 2009 | 2 |
| Cupcake | 1.5 | 2.6.27 | April 27, 2009 | 3 |
| Donut[4] | 1.6 | 2.6.29 | September 15, 2009 | 4 |
| Eclair[5] | 2.0 – 2.1 | 2.6.29 | October 26, 2009 | 5 – 7 |
| Froyo[6] | 2.2 – 2.2.3 | 2.6.32 | May 20, 2010 | 8 |
| Gingerbread[7] | 2.3 – 2.3.7 | 2.6.35 | December 6, 2010 | 9 – 10 |
| Honeycomb[8] | 3.0 – 3.2.6 | 2.6.36 | February 22, 2011 | 11 – 13 |
| Ice Cream Sandwich[9] | 4.0 – 4.0.4 | 3.0.1 | October 18, 2011 | 14 – 15 |
| Jelly Bean[10] | 4.1 – 4.3.1 | 3.0.31 to 3.4.39 | July 9, 2012 | 16 – 18 |
| KitKat[11] | 4.4 – 4.4.4 | 3.10 | October 31, 2013 | 19 – 20 |
| Lollipop[12] | 5.0 – 5.1.1 | 3.16 | November 12, 2014 | 21 – 22[13] |
| Marshmallow[14] | 6.0 – 6.0.1 | 3.18 | October 5, 2015 | 23 |
| Nougat[15] | 7.0 – 7.1.2 | 4.4 | August 22, 2016 | 24 – 25 |
| Oreo[16] | 8.0 – 8.1 | 4.10 | August 21, 2017 | 26 – 27 |
| Pie[17] | **9.0** | 4.4.107, 4.9.84, and 4.14.42 | August 6, 2018 | 28 |

**Legend:** Old version | Older version, still supported | **Latest version**

גרסאות אנדרואיד



# נסתרות

תכנית ק. קרקע

תכנית קומה טיפוסית

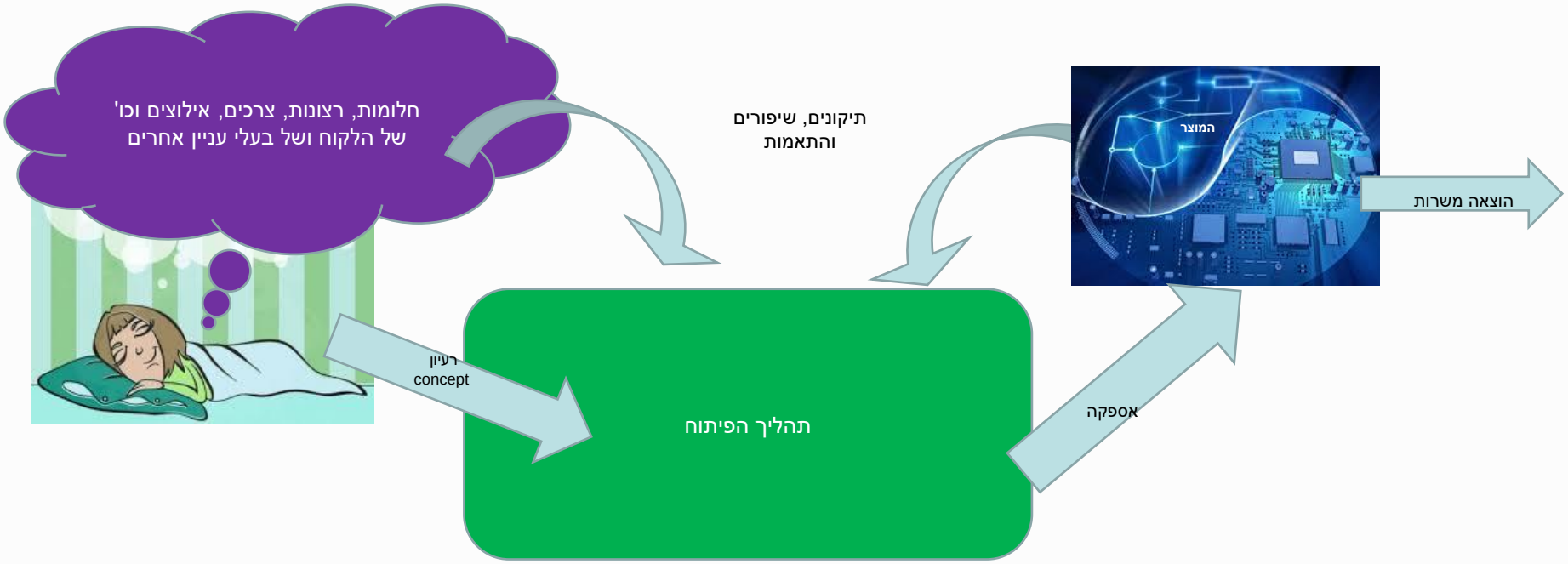
# צמצום המורכבות המקרית

- שימוש ב-High-Level languages – גורם לשיפור של עד פי 5 בפרודוקטיביות
- שימוש במערכות מומחה- יכול להציע שיטות עבודה מומלצות ולעזור למפתחים מתחילים. לדוגמא- יצירת בדיקות אוטומטיות.
- Autonomic Programming. לדוגמא- DeepCoder
  - (Balog, Matej, et al. "Deepcoder: Learning to write programs." *arXiv preprint arXiv:1611.01989* (2016).)
- Program verification- מציאת באגים בתחילת התהליך
- סביבות וכלים. לדוגמא- version control, Visual studio, Eclipse
- שימוש בשיטות עבודה מתקדמות- כדוגמת agile

# צמצום המורכבות המהותית

- קניית תוכנת מדף
- חידוד הדרישות ובניית אב טיפוס
- למצוא מהנדסי תוכנה טובים יותר-
  - לא כל המפתחים שווים ביניהם....
  - יכול להגדיל את הביצועים בעד פי 10
- Write-> Build -> Grow a program- שינוי הגישה של פיתוח מערכת התוכנה, כמו שהמוח גדל ומתווספות יכולות, כך גם בתוכנה- התוכנה גדלה ומתפתחת ויש לה יותר יכולות.
  - התייחסות כזאת משפרת את הביצועים של התוכנה
  - נשתמש במודל עבודה שתומך בשיטה זו

# מחזור החיים של מוצר תוכנה



# פעילויות בפיתוח התוכנה

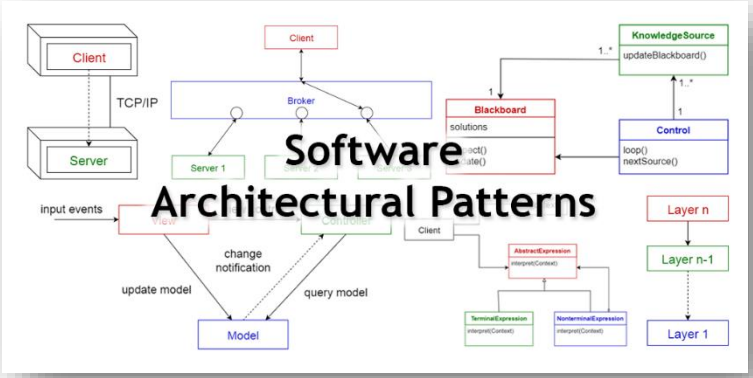
- ייזום - זיהוי בעיות והזדמנויות
- הגדרת דרישות – תיאור מדויק של הנדרש
- ניתוח - כיצד ניתן לפתור את הבעיות
- עיצוב - בחירה ותכנון הפתרון המתאים
- מימוש - תרגום התוכניות למציאות
- בדיקות - בחינת התוצאות מול התכנון
- שילוב – התקנת המערכת והטמעה אצל המשתמשים
- תחזוקה - תהליך מתמשך של ניפוי שגיאות והרחבות

---

דיונים, הערכת סיכונים, תיעוד, תיעדוף, ארכיטקטורה....

# מהו תכן תוכנה (Software Design)

- איפיון והגדרת פתרון על בסיס דרישות ואילוצים.
- מטרה ארוכת טווח : התמודדות מוצלחת עם דרישות משתנות.
- הגדרת מבנה ומסגרת לקידוד.
- מה בין ארכיטקטורה לעיצוב תוכנה (ותבניות מסוגים אלה).
- **עיצוב תוכנה ⇔ מודולציה**




https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013

# מודולים ומערכת

- Separation Of Concerns
  - עקרון תכנוני : "הפרד ומשול".
  - חלוקת התוכנה לחלקים כך שכל חלק מטפל בנושא אחר.
  - החלוקה צריכה להיעשות בצורה "נכונה"
  - חלק מהמערכת השלמה = מודול.
  - תוכנה המחולקת למודולים היא מערכת *מודולרית*.

# מודולים

- הפרדה נכונה:
  - צימוד נמוך
  - לכידות גבוהה
- עקרון SOC


Separation of concerns is a software design principle for separating an application into distinct sections, so each section addresses a separate concern.

At its essence, Separation of concerns is about Order.

The overall goal of SOC is to establish a well-organized system where each part fulfills a meaningful and intuitive role while maximizing its ability to adapt to change (**Maintainability, Extensibility, Reusability** )

# ארטיפיקטים

- תוצר לוואי של פיתוח תוכנה
- ארטיפיקטים שונים –
  - דיאגרמות UML
  - הערכת סיכונים
  - Use cases
  - קוד המערכת
  - ...

# לסיכום

אוניברסיטת אריאל בשומרון

היכרות

פעילויות

**הרבה מושגים חדשים**
**נדבר עליהם בפירוט בשיעורים הבאים**

דסה

הנדסת תוכנה

מודל

בעלי תפקידים

מערכת

ארטיפיקטים

קשיים בפיתוח תוכנה