קול המערכות
The Voice of the Systems

כתב העת של מהנדסי המערכות בישראל

גיליון שני - פברואר 2007

תוכן העניינים

דבר המערכת שגיאה! הסימניה אינה מוגדרת.

ועדת המערכת 2

דבר העורך 3

משולחן הנהלת INCOSE_IL 4

קול קורא לכתיבת מאמרים 7

המאפיינים של מהנדסי מערכות מוצלחים 8

ד"ר מוטי פרנק, HIT - מכון טכנולוגי חולון

Systems engineers are from Mars, software engineers are from Venus; how to help them communicate[1,2] 35

Joseph Kasser and Sharon Shoshany

הכנס הישראלי הראשון בנושא מחקר בהנדסת מערכות- הטכניון 48

האיגוד הישראלי להנדסת מערכות INCOSE_IL - ערכת הצטרפות 51

[1] Acknowledgment is made to John Gray Ph.D. for the use of his metaphor.
[2] This chapter is based on - Kasser J.E., Shoshany S., "Systems Engineers are from Mars, Software Engineers are from Venus", *proceedings of the 13th International Conference on Software & Systems Engineering and their Applications*, Paris, France, 2000, and - Kasser J.E., Shoshany S., "Bridging the Communications Gap between Systems and Software Engineering", *proceedings of the INCOSE-UK Spring Symposium*, 2001.

קול המערכות
The Voice of the Systems

אנו שמחים להגיש לכם את המהדורה השניה של כתב העת למהנדסי המערכת בארץ. אנו תקווה שבעקבותיה יבואו עוד מהדורות נוספות וטובות ממנה, לשם כך אנחנו זקוקים לכם, אנא נצלו כתב עת זה כבמה וכערוץ תקשורת בין מהנדסי המערכות בארץ, אתם מוזמנים לכתוב מאמרים מקוריים משלכם, לתרגם מאמרים טובים שקראתם, להציע להביא מאמרים מאלפים שנתקלתם בהם, לחוות דעה או ביקורת, להשתתף בפעילויות INCOSE_ IL, לכתוב מכתבים למערכת, להגיב על המאמרים שקראתם או סתם ליהנות מקריאתו. כתב העת **'קול המערכות'** מופק על ידי הסניף הישראלי של INCOSE (האיגוד העולמי להנדסת מערכות) במסגרת אילטם - איגוד משתמשי טכנולוגיות מתקדמות בתעשיית האלקטרוניקה, האיגוד הנו עמותה, מלכ"ר בבעלות החברות החברות באיגוד. פעילותו ממומנת ע"י דמי חברות שנתיים המשולמים על ידי החברות ונתמכת ע"י תכנית מגנ"ט של לשכת המדען הראשי, במשרד התמ"ת.

INCOSE_IL הינו הסניף הישראלי של הארגון הבין לאומי INCOSE אשר מטרתו לקדם את הידע, ההבנה, התאום והיישום של הנדסת מערכות בעולם. מטרות INCOSE_IL (כמו בארגון הבינלאומי) הן לקדם את נושאי הנדסת המערכות בישראל: בתעשייה, באקדמיה ובמוסדות ממשלתיים בארץ ע"י:

קידום התאום ושיתוף הפעולה בנושאי הנדסת המערכות בתעשייה בישראל.

טיפוח ההשכלה האקדמית והמחקר בנושאי הנדסת המערכות בארץ.

ביסוס מעמדה של ישראל כאחת המובילות בעולם בתחום זה.

להוות את הגורם המקצועי המתאם בין התעשייה, האקדמיה וגורמי הממשלה בנושאי הנדסת מערכות.

ארגון INCOSE הוא ארגון דינמי הכולל כיום, 17 שנה מיום הקמתו, למעלה מ - 6,500 חברים מכל העולם ו - 53 סניפים מ- 36 מדינות. הסניף הישראלי, INCOSE_IL, הוקם לפני ארבע שנים בשיתוף עם אילטם וכולל כ - 800 חברים פעילים מחברות שונות כגון תע"א, רפא"ל, אל-אופ, קומברס, צה"ל, טכניון, תע"ש, אורבוטק, אי.סי.איי, HP אינדיגו, קודאק, נייס מערכות ועוד. מתוכם חברים כ - 100 מהנדסים באיגוד INCOSE הבינלאומי. הארגון הבינלאומי מקיים שני כנסים בשנה. בכנס השנתי המרכזי משתתפים למעלה מ- 1,000 איש בארבעה ימי הרצאות, סמינרים ותערוכות. הנושאים המטופלים במסגרת הארגון, ע"י קבוצות עבודה הפועלות במהלך השנה, הם בתחומים מגוונים כגון: Process & Methods * Modeling & Tools * Measurement * SE Applications * SE Management * כל אחד מחברי הארגון יכול להשתתף ולתרום בקבוצות עבודה אלו. גם בסניף הישראלי פועלת קבוצת עבודה בנושא ניהול אינטגרציה. מלבד הכנס השנתי הגדול, מקיים הסניף הישראלי: מפגשים מקצועיים בנושאי הנדסת מערכות בהנחיית מרצים מקצועיים מהתעשייה, מפגשים בסגנון קפה-ידע לצורכי החלפת מידע/ניסיון בין החברים ומפגשי חברים מקבוצות עניין.

חברי ארגון **INCOSE** הבינלאומי מקבלים כל שלושה חודשים שתי חוברות מקצועיות בתחום הנדסת מערכות, נחשפים לתוצרים הטכניים של ועדות משנה, ניגשים (דרך אתר הארגון) למידע חשוב וחדשני כגון System Eng. Handbook, משתתפים בכנסים בינלאומיים והחשובה ביותר היא ההפריה ההדדית שבהשתתפות במפגשים עם מומחים ובעלי ניסיון בתחום הנדסת מערכות מהתעשייה, האקדמיה וצה"ל.

אילטם - איגוד המשתמשים שם לו למטרה להגדיל את יכולת התחרותיות של החברים על ידי שיתוף בידע הקיים, האצת היישום של טכנולוגיות מתקדמות, הקמת רשתות מומחים, הורדת עלויות הפיתוח והייצור תוך העלאת איכות המוצרים. **אילטם** פועל למען חבריו **ואינו בעל עניין בשום פעילות מסחרית**. כיום חברות **באילטם כמאה ועשרים חברות** מהתעשייה האזרחית והביטחונית.

אילטם פועל במסגרת תוכנית מגנט של משרד המדען הראשי במשרד התמ"ת.

הרחבת פעולות האיגוד בתחומי הפיתוח והתוכנה הובילו לדרישה של התעשייה לביצוע פעולות בנושא הנדסת מערכות. יחד עם אנשי תעשייה מרפאל, תע"א, אורבוטק והטכניון הוקם התא הישראלי INCOSE_IL של איגוד INCOSE העולמי.

החברות באילטם היא ברמת הארגון / חברה. כל עובדי חברה החברה באילטם זכאים להשתתף בפעילויות האיגוד, לרבות פעילויות INCOSE_IL המיועדת למהנדסי מערכות.

הצטרף ל - INCOSE_IL ותהיה שותף

קול המערכות
The Voice of the Systems

דבר העורך

קורא יקר,

בשעה טובה אתה מחזיק בידך את המהדורה השנייה של כתב העת כתב העת של מהנדסי המערכות בישראל ששמו יקרא מעתה:

קול המערכות -
The Voice of the Systems

זהו השם שנבחר בתחרות השמות שהכרזנו עליה במהדורה הקודמת. הקופירייטר הוא ד"ר אביגדור זוננשיין שיזכה בפרס ובתהילת עולם. מהדורה זו יוצאת לקראת **הכנס הישראלי הרביעי להנדסת מערכות שיתקיים בין ה-20 ל- 22 במרץ 2007** ויהיה מיוחד בתכניתו ובתכניו. הכנס יורכב משלושה ימים בעלי דגשים שונים: **היום הראשון** יתקיים ב- **20 במרץ 2007** במלון דניאל בהרצליה והוא יכלול מושבים מקבילים בהם יציגו מומחים את עבודותיהם ופאנלים בנושאי הנדסת מערכות.

היום השני יתקיים ב- **21 במרץ 2007** במלון דניאל בהרצליה והוא יכלול **Tutorials** שיוצגו על ידי מומחים מהארץ ומהעולם.

היום השלישי יתקיים ב- **22 במרץ** בטכניון בחיפה והוא יכלול את הכנס הבינלאומי בנושא **Systems Modeling**.

הכנס נראה השנה מעניין ומגוון מתמיד.

ואם בכנסים עסקינן, הרי ב 6.12.2006 התקיים בטכניון הכנס הישראלי הראשון מזה 2000 שנה על מחקר בהנדסת מערכות. מטרת הכנס הייתה להפגיש את העוסקים בתחומי המחקר בהנדסת מערכות במוסדות השונים, כדי להכיר את נושאי המחקר המתבצע ולהגביר את שיתוף הפעולה בין החוקרים. הכנס זכה להיענות מפתיעה ולתגובות מצוינות. בעמ' 48 ניתן לקרוא סקירה של פרופ' מנחם וייס על הכנס.

במהדורה הראשונה של כתה העת הכרזנו: "אנו נשאף להפיק כתב עת מקצועי שיכלול מאמרים אקדמיים ומאמרים יישומיים ברמה מקצועית גבוהה שעומדת ברמת העיתונות הבי"ל של הנדסת מערכת. נשאף להביא את כתב העת למוניטין של מוביל ברמה בין- לאומית."

ברוח זו אנו גאים לכלול במהדורה זו שני מאמרים, פרי עטם של חברינו, מהנדסי מערכת ישראלים שזכו לפרסום והערכה בין לאומיים, שניהם עוסקים בדמותו של מהנדס המערכת, כישוריו, תפקידו והתרבות המקצועית בסביבתו.

<u>המאמר הראשון:</u>

" Knowledge, Abilities, Cognitive Characteristics and Behavioral Competences of Engineers with a High Capacity for Engineering Systems Thinking (CEST) "

נכתב על ידי **ד"ר מוטי פרנק מהמכון הטכנולוגי בחולון**. המאמר פורסם בכתב העת Systems Engineering ומציג ממצאים של שלושה מחקרים אשר עסקו בזיהוי המאפיינים של מהנדסי מערכות מוצלחים (מהנדסי מערכות בעלי יכולת ראייה מערכתית גבוהה). המאמר מציג את הרקע, סקר ספרות, המתודולוגיה של המחקרים, הממצאים, דיון ומסקנות. המאמר מטפל בבעיות ובהתלבטויות אופייניות למהנדסי מערכות המוכרות לרבים מאיתנו ומתמודד איתם האופן מאיר עיניים.

<u>המאמר השני:</u>

" Systems engineers are from Mars, software engineers are from Venus; how to help them communicate "

הוא פרי שיתוף פעולה בין **גב' שרון שושני מרפאל ופרופסור ג'ו קסר מהמרכז להנדסת מערכות (SEEC) באוניברסיטה של דרום אוסטרליה**. המאמר משלב שני מאמרים של השניים שפורסמו במועדים שונים. המאמר עוסק בנושא החם ביותר בהנדסת מערכת כיום - תרבויות - או פערי תרבות. הנושא פופולארי הן בזכות פרויקטים חובקי עולם בהם מהנדסי מערכת נדרשים לגשר על פער תרבויות מסוג אחד, והן בזכות המולטידיספלינריות השלטת בפרויקטים גדולים עכשוויים וגוררת התמודדות בפערי התרבות בין הדיסציפלינות המעורבות בצוות הפיתוח. הפער הקלאסי אתו מתמודד המאמר הינו בין מהנדסי התוכנה למהנדסי המערכת - לכאורה הנדסות שרב המשותף בצורת החשיבה הנדרשת בהן, אך בעצם קיים ביניהן פער תקשורת בסיסי (כמו בספר המפורסם - נשים מווונוס וגברים ממאדים). המאמר עוסק לא רק בפער, אלא באיך ניתן להתמודד אתו. מי שכבר התנסה בפער התרבותי הזה ונדרש לגשר עליו, בפרויקט מערכתי מורכב, בוודאי מצא את עצמו בנוגה או במארס...המאמר כתוב גם בצורה מקורית ומהנה לקריאה.

לבסוף, אנו חוזרים וקוראים לכם, מהנדסי המערכות בישראל, לתרום מניסיונכם ומהידע שלכם ולשתף בו את קהיליית הנדסת המערכות שלנו. אנו מזמינים אתכם לכתוב מאמרים, תגובות או להעלות נושאים לדיון שניתן להם במה.

קריאה מהנה ולהתראות בכנס ,
ד"ר אמי הרי, העורך

משולחן הנהלת INCOSE_IL

חברי INCOSE_IL, מהנדס מערכות יקר,
אני שמח לברך את עצמנו על הגיליון השני של ביטאון הנדסת המערכות בישראל "קול המערכות". ביטאון זה משמש קול לכל מהנדסי המערכות שמוכנים לחלוק מניסיונם באמצעות מאמר מקצועי לביטאון.
הביטאון הפעם מוקדש לאנשים שמאחורי הנדסת מערכות – מי הם? כיצד מאתרים אותם? איך מטפחים אותם? כיצד לפתח תקשורת יעילה בין מהנדסי מערכת למהנדסי תכנה?.

קריאה מהנה ומחכימה!

האיגוד הישראלי להנדסת מערכות INCOSE_IL

INCOSE_IL הינו הסניף הישראלי של הארגון הבין לאומי INCOSE. ארגון אשר מטרתו לקדם את הידע, ההבנה, התאום והיישום של הנדסת מערכות בעולם. מטרות INCOSE_IL (כמו בארגון הבינלאומי) הן לקדם את נושאי הנדסת המערכות בישראל, בתעשייה, באקדמיה ובמוסדות ממשלתיים בארץ ע"י:

- קידום הדרכות וחינוך מהנדסי מערכת במפעלים, באקדמיה ובהשתלמויות כלליות.
- קידום שיתופי פעולה ושיתוף במידע וידע בין מהנדסי מערכת באמצעות מפגשים, קבוצות עבודה וכנסים ייעודיים.
- הפצת מידע לידע מקצועי בנושא הנדסת מערכות באמצעות אתר האיגוד.
- עידוד מחקר אקדמי בנושאי הנדסת מערכות.
- ייזום וייצור שיתופי פעולה עם איגודים מקצועיים רלוונטיים כמו PMI, ISQ.

ארגון INCOSE הוא ארגון דינמי הכולל כיום, כ- 17 שנה מיום הקמתו, למעלה מ- 6500 חברים מכל העולם ו- 53 סניפים מ- 36 מדינות. הסניף הישראלי, INCOSE_IL, הוקם לפני ארבע שנים בשיתוף עם אילטם, איגוד משתמשי טכנולוגיות מתקדמות, תיכון וייצור בתעשיית האלקטרוניקה וכולל חברים מחברות שונות אזרחיות וביטחוניות. בפעילויות הסניף הנערכות בארץ משתתפים למעלה מ- 600 מהנדסי מערכות.

החברים בארגון INCOSE העולמי מקבלים את הרבעון Systems Engineering של ידיעון INSIGHT - INCOSE, מקושרים לקהיליית מהנדסי המערכות העולמית דרך אתר INCOSE לרבות גישה לאלפי מסמכים ומאמרים הנמצאים באתר, ויכולים להשתתף בעשרות קבוצות עבודה מקצועיות. לחברי INCOSE הנחה בכנסים הבינלאומיים של INCOSE.

למה INCOSE_IL שווה זהב?

אחת לשנה בוחן INCOSE העולמי את הפעילות המתקיימת באיגודים המקומיים בארה"ב וברחבי העולם.
בשנת 2006, הגשנו את עצמנו לראשונה לבחינה זו לאחר שצברנו נפח ואיכות פעילות מקצועית ראויה לציון.

INCOSE_IL ביצעה ב- 2006 עשרות פעילויות מקצועיות, מפגשים טכניים בנושאים בעלי עניין לקהילת מהנדסי המערכת (כמו: ניהול סיכונים, בחינות ובדיקות), קבוצות עבודה בנושאים חיוניים (כמו: ניהול אינטגרציה), ביקורים מקצועיים בארגונים המקדמים הנדסת מערכות (כמו: ח"א), הפצת מידע טכני ומקצועי לקהילת מהנדסי המערכות ע"י האתר של האיגוד (www.iltam.org).

בנוסף, INCOSE_IL הוערכה על שיתופי הפעולה שייזמה עם איגודים וגופים מקצועיים אחרים, כמו: PMI ישראל, איגוד ישראלי לאיכות, מנה"ר - התכנית למצוינות, הטכניון, ITEA - ישראל.
INCOSE_IL ערכה בשנת 2005 את הכנס הלאומי השלישי להנדסת מערכות בו השתתפו למעלה מ- 300 מהנדסי מערכות מתעשיית הביטחון, התעופה, התקשורת וההי-טק.

INCOSE_IL שמרה על קשר קבוע ויעיל עם קהילת מהנדסי המערכות ע"י המפגשים הטכניים, הכנס ואתר האיגוד.
INCOSE_IL גם הוכיחה התנהלות תקינה מבחינת פגישות צוות ההנהלה, הניהול הכספי ותיעוד הפגישות והפעילויות.
בנוסף, אנשי INCOSE_IL תורמים רבות לפעילות הבינלאומית של INCOSE ע"י פעילויות כמו: הצגת מאמרים ופנלים בכנסים הבינלאומיים, סיוע והשתתפות בארגון הכנסים הבינלאומיים, פרסום מאמרים בירחון המקצועי של INCOSE.

על כל פעילותנו רחבת ההיקף והאיכותית זכינו במדליית זהב של INCOSE.

INCOSE_IL - 5 שנות הצלחה

האיגוד הישראלי להנדסת מערכות INCOSE_IL הוקם בשלהי 1999 במסגרת אילטם.
למרות שמדלית הזהב מתייחסת לפעילות ולהישגים בשנת 2005, הרי הישגים אלו מתבססים על הבסיס שהוקם ב- 5 השנים האחרונות.
נכנסתי לתפקיד נשיא INCOSE_IL בספטמבר 2005, כאשר הפעילות הענפה והמוצלחת בשנות ההקמה הובלה ע"י צוות הנהלת INCOSE_IL בראשות שני הנשיאים הראשונים - מר יעקב קגן, מל"מ תע"א וד"ר מיכאל וינוקור מהנהלת תע"א.
ב- 5 שנים אלו נוצרה תשתית מצוינת לפעילות INCOSE_IL למען קהילת מהנדסי המערכת בישראל. גולת הכותרת בפעילות זו הייתה 3 כנסים לאומיים להנדסת מערכות רבי משתתפים ורבי עניין.

הכנס הלאומי הרביעי להנדסת מערכות יוצא לדרך

הכנס הלאומי הרביעי של INCOSE_IL יוצא לדרך.
פרטים באתר הכנס www.iltam.org/incose_il2007
הכנס הלאומי מתוכנן ל- 20.3.2007-22.3.2007.
בכנס זה יהיה הדגש על **חדשנות וחידושים בהנדסת מערכות**.

הכנס הלאומי הרביעי ישודרג ע"י שיתוף פעולה עם הטכניון, הפקולטה לתעשייה וניהול ומרכז גורדון להנדסת מערכות בטכניון - ונקיים בו ימי עיון בינלאומיים בנושאי מודלים, סימולציות והנדסת מערכות.
רצ"ב כל הפרטים על הכנס הלאומי וימי העיון הבינלאומיים. היכונו להשתתף באירוע מקצועי ברמה גבוהה!

אנחנו נפגשים אחת לחודש!

INCOSE_IL ואילטם מציעים לחברי INCOSE_IL, אילטם וקהילת הנדסת המערכות בישראל פעילות אחת לחודש לפחות.
אני שמח לדווח לכם שהפעילות המקצועית שלנו במסגרת INCOSE_IL תפשה תאוצה ונפח:

- במהלך 2006 בכל חדש הייתה פעילות מקצועית – הרצאה, סמינר, ביקור.
- קיימנו לראשונה יום עיון מרתק בטכניון על ממצאי מחקרים שנערכים בטכניון ובמקומות אחרים בנושאים מערכתיים.
- השקנו את הביטאון "קול המערכות"
- השתתפנו באופן פעיל בכנסים של INCOSE באורלנדו ובאדינבורו.
- התנענו את ההכנות לכנס הלאומי של INCOSE_IL
- קיימנו מושבים מיוחדים בחסות INCOSE בכנס ה- PMI ובכנס הבינלאומי של האיגוד הישראלי לאיכות.

על הפעילות הזאת הגשנו את המועמדות של INCOSE_IL למדלית זהב של INCOSE שתחולק (אם נזכה) בכנס הבינלאומי ביוני 2007 בסאן דיאגו.
להזכירכם, יש לנו כבר מדליית זהב על פעילותנו ב- 2005.

מה צפוי לנו ב- 2007?

נפגש כולנו בכנס הלאומי הרביעי להנדסת מערכות שיתקיים בהרצליה ב- 21, 20 למרץ 2007.
המוטו של הכנס "חדשנות וחידושים". הקפדנו להציג רק חדשות וחידושים.
הכנס מתקיים בשיתוף פעולה של הטכניון, כאשר היום השלישי של הכנס 22/3/07 מתקיים בטכניון / חיפה ומתמקד במודלים וסימולציות. צפוי לנו יום מרתק בטכניון.
נמשיך בפעילותנו המקצועית הענפה במפגשים מקצועיים, סמינרים, ימי עיון, ביקורים בחברות ומפעלים.
בואו והשתתפו בפעילות זו – תלמדו, ותלמדו אחרים מניסיונכם. הפעילות היא אתכם ועבורכם.

בברכה,

ד"ר אפיאדור גולנשיין
יו"ר INCOSE_IL

פעילויות שהיו - רבעון ראשון שנת 2007:

Jon Gross & Tim Kasse	CMMI - Intermediate level	סדנא	14-18.1.2007
Robert Halligan	Systems Engineering Training	סדנא	14-18.1.2007
יצחק לביא, שלום לויפר	הערכת עלויות פרוייקט פיתוח בתעי"א	מפגש	15.2.2007

פעילויות מתוכננות:

מרץ 2007 –

- 12.3.2007 - מפגש חברות באינדיגו HP.
- 19.3.2007 – קבוצת עבודה CMMI , ביקור בחברת אמדוקס.
- 22.3.2007 –20 - הכנס הלאומי הרביעי להנדסת מערכות.

יולי 2007 –

- סמינר בן יומיים בנושא: Tim Kasse – Peer Review.
- יום סמינר בנושא: Bob Klain – CMMI for SME.

מועד עדיין לא ידוע

- מתוכנן מפגש חברות נוסף ביולי 2007
- מתוכנן מפגש טכני נוסף ביוני 2007

קול קורא לכתיבת מאמרים

מוזמנים בזה מאמרים לפרסום בכתב העת

מוזמנים מאמרים אקדמיים ומאמרים יישומיים ברמה מקצועית גבוהה שעומדת ברמת העיתונות הבינ"ל של הנדסת מערכת אנו נשאף שהפרסומים בכתב העת מקצועי של מהנדסי המערכות בישראל יציבו אותו ברמה של כתב עת מוביל ברמה בינלאומית. כתב העת יכלול מאמרים מהסוגים הבאים:

1. מאמרים מקוריים שיכתבו על ידי מהנדסי מערכת וחוקרים מישראל שיבוקרו על ידי רפרנטים מקצועיים.
2. מאמרים בולטים שהתפרסמו בכנסים בארץ ובחו"ל.

המאמרים יכולים להיות בעברית או באנגלית.

נשתדל להפנות מאמרים שימצאו מתאימים לפרסום בכתבי עת בינלאומיים. כותבים בעברית יוכלו לקבל סיוע בתרגום מאמרים לאנגלית לצורך פרסומם בחו"ל.

כתובת למשלוח מאמרים: incose_il@iltam.org

הצטרפו ל- INCOSE_IL!

התועלות המקצועיות והאישיות מהצטרפות ל-INCOSE_IL.

רצ"ב ערכת גיוס והצטרפות (עמוד 52).

מלאו הטפסים והיו חלק פעיל של המקצוענים להנדסת מערכות בישראל

נשמח לראותכם בין חברינו ומשתתפים בפעילויותינו!

המאפיינים של מהנדסי מערכות מוצלחים

ד"ר מוטי פרנק, HIT - מכון טכנולוגי חולון

תקציר

המאמר שלהלן פורסם בכתב העת *Systems Engineering* ומציג ממצאים של שלושה מחקרים אשר עסקו בזיהוי המאפיינים של מהנדסי מערכות מוצלחים (מהנדסי מערכות בעלי יכולת ראייה מערכתית גבוהה). המאמר המלא מציג את הרקע, סקר ספרות, המתודולוגיה של המחקרים, הממצאים המלאים, דיון ומסקנות. בתקציר בעברית יוצגו רק עיקרי הממצאים.

בממד הקוגניטיבי נמצא שמהנדסי מערכות מוצלחים מבינים את המערכת השלמה מעבר למרכיבים הבודדים ו"רואים" איך המרכיב הבודד פועל כחלק מהמערכת או מהמכלול השלם. הם מסוגלים להבין את השלם ואת התמונה הכוללת, פונקציונלית וקונספטואלית, גם מבלי לדעת ולהבין את כל הפרטים על בורים (ומבלי שאי ידיעת/הבנת כל הפרטים מפריעה להם). הם גם מבינים איך תתי המערכות מתחברות למערכת אחת שלמה העונה על המפרטים והדרישות ו"רואים" את קשרי הגומלין וההשפעות ההדדיות בין מרכיבי המערכת בינם לבין עצמם ובינם לבין מערכות אחרות בסביבה. מהנדסי מערכות מוצלחים מבינים את המערכת מכל ההיבטים הרלבנטיים ומנקודות מבט שונות (למשל, operational, technical, system), את ההשלכות הכוללות של שינויים מוצעים - ההנדסיים והלא הנדסיים - הן היזומים ע"י הקבלן והן הנדרשים ע"י הלקוח ואת אופן הפעולה, המטרות, השימושים, היתרונות והמגבלות של מערכת חדשה מיד עם הצגתה. הם מסוגלים למצות את הסינרגטיות מהשילובים בין מרכיבים שונים לפתור בעיות ותקלות מערכתיות. מהנדסי מערכות מוצלחים מסוגלים להפעיל שיקולי אופטימיזציה ולבצע הן אנליזות top-down והן סינתזות ואינטגרציות bottom-up. מהנדסי מערכות מוצלחים מסוגלים להפעיל חשיבה יצירתית-לאטרלית-מסתעפת-היוריסטית (בשלבי העלאת רעיונות) וחשיבה לוגית-ממוקדת-מנתחת-אלגוריתמית (בשלבי היישום). מכיוון שאי אפשר ללמוד ולדעת הכול מהנדסי מערכות מוצלחים מסוגלים להקיש ולהסיק מסקנות מדיסציפלינת ההתמחות על דיסציפלינות אחרות.

בממד היכולות נמצא שבשלב הגדרת הבעיה וייזום הפרויקט, מהנדסי מערכות מוצלחים מבינים ומסוגלים לנתח את צורכי, אילוצי, יכולות, מגבלות ועדיפויות הלקוח ואת ייעוד ומשימות הארגון ו/או את צורכי השוק. הם מבינים ומסוגלים להגדיר את התפיסה התפעולית (התפיסה המבצעית, CONOPS), את מטרות ושימושי המערכת ומהי פונקצית המטרה שלה ולבצע ניתוח דרישות ברמת מערכת, ל"תרגם" את הצורך והתפיסה התפעולית למפרט דרישות (לא טכניות) ו"לתרגם" את מפרט הדרישות למפרט מערכת (הכולל פרמטרים טכניים).

בשלב הגדרת הפתרון ותכן העל, מהנדסי מערכות מוצלחים מבינים ומסוגלים לגבש קונספט לפתרון (מ"קצה לקצה") בגישת Top-Down, לבצע ניתוח פונקציונלי ברמה הלוגית, ל"תרגם" את הדרישות ומפרט המערכת ל"סכימת בלוקים" בה כל בלוק הוא פונקציה מוגדרת, לפרק את סכימת הבלוקים הפונקציונלית מרמת מערכת לאלמנטים, תת מערכות ומכלולים, להגדיר עקרונית ממשקים בין המרכיבים השונים ולבצע תכן ארכיטקטורה ברמה הפיזית – להקצות את הפונקציות לאלמנטים השונים של המערכת (ולהגדיר את הסידור הפיזי שלהם).

לגבי יכולות כלליות נמצא שמהנדסי מערכות מוצלחים מבינים ומסוגלים להפעיל שיקולי תכן מערכתיים (כגון מינימיזציה של צמידות/ממשקים, קונספט ל-BIT, re-use, ביזור/קיבוץ, מערכות פתוחות, סטנדרטים), להפעיל שיקולי אופטימיזציה, פשרות ו-trade-offs במישורים ההנדסי, ההנדסי-כלכלי וההנדסי-מבצעי ולהשתתף או להוביל בדיקות היתכנות, להציג מספר אלטרנטיבות לפתרון בעיות, לנתחן ולבחור בחלופה האופטימלית. הם מבינים ומסוגלים לאפיין, להשתמש (ולבנות) סימולציות מערכתיות (סימולטורים, מודלים, Mock-Up, תוכנות סטנדרטיות/ייעודיות וכד') וכן לאפיין, להגדיר ולהשתמש בכלים ייעודיים להנדסת מערכות. מהנדסי מערכות מוצלחים מסוגלים ל"ראות את העתיד": לטווח הארוך – חזון, התפתחויות טכנולוגיות צפויות, צרכים עתידיים של השוק או הלקוח. לטווח הבינוני – לקחת בחשבון את אורך החיים הצפוי של המערכת ומה יהיה לאחר מכן. לטווח הקצר – לצפות קשיים, בעיות, סיכונים ודרישות לשינויים במהלך הפרויקט.

בממד היכולות האישיות (behavioral competences) נמצא שמהנדסי מערכות מוצלחים הינם בעלי יכולת לפעול ולקבל החלטות גם בתנאי אי וודאות ו/או קיום מידע חלקי ולא חושש להתמודד עם תחומים שאינם מומחים בהם. הם בעלי יחסי אנוש טובים (יכולת לעבוד בצוות; יכולת לעבוד בשיתוף פעולה; יכולת לקשור יחסי אמון עם הלקוח, עמיתים, ספקים, שותפים, קבלני משנה) וכושר לימוד עצמי ויש להם רצון עז לעסוק במערכות. הם מגלים חדשנות, סקרנות, העזה ויזמות ויודעים לשאול את השאלות הנכונות. הם בעלי יכולת ניהולית ומסוגלים להוביל צוותים אינטרדיסציפלינריים, לנהל ישיבות ולשלב בין מהנדסים-מומחים מדיסציפלינות שונות (לרבות בניית צוותים, הגדרת יעדים, קבלת החלטות, פתרון קונפליקטים, הנעת אנשים ויצירת מחויבות). הם מסוגלים להביא את כל תתי הפעילויות לידי סיום מוצלח של הפרויקט ולבצע מעקב ובקרת פרויקט תוך שימוש בכלי בקרה והם יודעים להגדיר גבולות ותחומי אחריות באופן מיטבי (בין

צוותים, מחלקות, שותפים, קבלני משנה, out-sourcing). מהנדסי מערכות מוצלחים מסוגלים להפעיל ראייה עסקית-שיווקית: להתחשב גם בשיקולים כמו שיקולים עסקיים, שיווקיים, כלכליים, ארגוניים, "פוליטיים", אישיים, סביבתיים, הזדמנויות עסקיות, לפתור בעיות הנדסיות, ארגוניות ולוגיסטיות במהלך הפרויקט, להפעיל פיקוח והשפעה על המעורבים בפרויקט (פנימיים או חיצוניים), "להציג נושא בכתב ובעל-פה ולנהל מו"מ טכני וכלכלי בגישת WIN-WIN.

בממד הידע נמצא שמהנדסי מערכות מוצלחים הינם בעלי ידע מעמיק בתחום ראשי אחד לפחות ("עוגן") ובנוסף הם בעלי ידע בתחומים רלבנטיים אחרים. בתחומים ה"אחרים" הם צריכים להיות בעל ידע כללי והבנה ברמת על ולהכיר את ה"ז'רגון" של דיסציפלינות שונות באופן שיאפשר להם לשאול ולתקשר עם עובדים מתחומים שונים.

נראה אפוא שמגוון המאפיינים של מהנדסי מערכות מוצלחים רחב מאוד. קשה למצוא אנשים הניחנים **בכל** המאפיינים שפורטו לעיל. בכל שלב של הפרויקט ובכל רמת וסוג תפקיד נדרש שילוב אחר של מאפיינים. ארגון מוצלח יודע להציב את מהנדסי המערכת הנכונים במקומות הנכונים בהתאם למאפיינים האישיים של כל אחד.

This is a reprint of an article published in *Systems Engineering*, Vol. 9, No. 2, pp. 91-103.
http://www3.interscience.wiley.com/cgi-bin/fulltext/112415412/PDFSTART

Knowledge, Abilities, Cognitive Characteristics and Behavioral Competences of Engineers with a High Capacity for Engineering Systems Thinking (CEST)

Moti Frank *

Department of Technology Management, Holon Institute of Technology, Holon, Israel 58102

ABSTRACT

A major high-order thinking skill that enables engineers to successfully perform systems engineering tasks is Engineering Systems Thinking. To successfully perform their tasks, systems engineers need a systems view or, in other words, a high capacity for engineering systems thinking (CEST). This paper summarizes the findings of three studies aimed at identifying the knowledge, abilities, cognitive characteristics (thinking skills) and personal traits (behavioral competences) of systems engineers with high capacity for engineering systems thinking (or in other words, successful systems engineers). The findings suggest that successful systems engineers possess interdisciplinary knowledge. They are expert in at least one main field but have general knowledge in additional fields and disciplines. They become familiar with the jargon and professional language of the other disciplines and are able to communicate with people or experts from different fields and disciplines. Overall, ten cognitive characteristics, eleven abilities, and ten behavioral competences were identified. In addition, nine additional roles of systems engineering were identified.

* E-mail: frankm@tx.technion.ac.il

Key words: systems engineering; systems thinking; engineering systems thinking; capacity for engineering systems thinking; systems view; systems thinking in engineers; cognitive characteristics of systems engineers; behavioral competences of systems engineers

1. INTRODUCTION

A major high-order thinking skill that enables engineers to successfully perform systems engineering tasks is *Engineering Systems Thinking.* To successfully perform their tasks, systems engineers need a systems view or, in other words, a high *capacity for engineering systems thinking* (*CEST*). The main characteristic of CEST is the ability to see the whole picture and all relevant aspects without getting stuck on details; to be able to identify the system's emergent properties, capabilities, behaviors, and functions without looking inside the system and its parts/components/details.

The question then becomes: Is this ability learned or innate? Can it be developed through experience, job rotation, education and training? Some authors refer to CEST as an innate ability. However, Frank [2002] found that this ability is a measurable and consistent quality of personality, that it can be used to distinguish between individual engineers, and that it is, presumably, a mix of innate talent and acquired experience.

This paper summarizes the findings of three studies aimed at identifying the abilities, cognitive characteristics (thinking skills) and individual traits (behavioural competences) of systems engineers with a high capacity for engineering systems thinking (or in other words, successful systems engineers).

The importance of exploring and identifying these characteristics is that they can become the keystones for developing (1) training classes, university programs and curricula for constructing systems thinking in engineers, (2) a plan for accelerating the development of systems engineers, through job design, job rotations, and development programs, and (3) a test for assessing CEST. This test could then be used for selection, filtering, screening, placement, and classification of candidates for engineering positions that require high CEST.

Remarks:

1. In the context of this paper, the following five phrases are equivalent and will be used interchangeably: (1) capacity for engineering systems thinking, (2) ability to take a systems view, (3) developed ability/capacity of/for systems thinking, (4) systems thinking in engineers, and (5) possessing a high level of engineering systems thinking skills.

2. The paper presents lists of abilities, characteristics and traits. According to the systems thinking approach, one must not see them as 'laundry lists' but rather assume that the items in these lists are interconnected and interrelated. For example, the items "not getting stuck on details" and "ambiguity tolerance" are interrelated. Ambiguity may be created when there are not enough details. Systems thinker, however, does not necessitate knowing the details in order to conceptually analyze the system. Another example is that for convenience "general cognitive characteristics" and "abilities" appear in separate sections yet all items listed under the title "abilities" require cognitive capabilities as well.

3. The paper does not deal with senior managers (CEOs, division managers, etc.). It is obvious that such managers need strategic, holistic, systems and business vision. The paper deals with senior, mid-level, and junior systems engineers. Systems thinking is required in all level of the hierarchy. Similarly, the abilities, competences and traits presented in this paper are hierarchical. This is to say, that the higher in the hierarchy are the areas handled by the systems engineer, the greater is the engineer's need for a broad systems perspective.

4. For convenience, the items in the lists throughout the paper refer to the abilities, characteristics, competences and traits of a single systems engineer. In real life, however, systems engineering work is often carried out by teams. So the task descriptions here refer both to a single systems engineer and to a systems engineer working within a team.

2. METHODOLOGY

As mentioned above, this paper summarizes the findings of three studies dealing with the capacity for engineering systems thinking (CEST). Since this paper focuses on practical issues and not on research methodology, only a brief description of these three studies will be presented. The primary aim of the first study [Frank, 2002] was to identify the characteristics of engineers having high CEST. This study was carried out in three stages. The first stage involved a pilot study in which 11 in-depth, open, non-structured interviews were held with key figures in the Israeli hi-tech industry. In the second stage the researcher spent many hours in the role of 'the-observer-as-participant,' conducting on-site observations at two hi-tech companies. This second stage involved 17 semi-structured interviews and content analysis of 14 lectures that were given in a seminar on systems engineering. The third stage of the study consisted of a survey based on a pilot questionnaire (N=31) and a final questionnaire (N=276). The qualitative-naturalistic inquiry paradigm, in which data collection is based principally on interviews and observations, was found to be suitable for the first and second stages, since the purpose of the research was to examine processes (and not only final outcomes) in the natural environment of the study participants. The number of interviews and observations conducted during the first and second stages was not set in advance. The criteria for concluding the data collection was saturation; that is, when it became clear that adding new subjects did not significantly alter the findings, the interviews were brought to a close. According to Morse [1995], as the study progresses, theoretical insights and linkages between categories increase, making the process exciting as what is going on finally becomes clearer and more obvious. Data collection and sampling are dictated by and become directed entirely toward the emergent model. The researcher seeks indices of saturation, such as repetition in the information obtained and confirmation of previously collected data. The selection of subjects in the first stages of the study was based on two primary assumptions. These were (1) senior systems engineers have high capacity for engineering systems thinking and (2) the appraisal of supervisors is a valid information source.

The second study [Frank & Elata, 2005] tried to understand the process in which undergraduate engineering students' awareness of systems-level considerations is increased due to their participation in the freshmen

project-based-learning course "Creative Introduction to Mechanical Engineering". The course attempts to simulate the process of design and development of new products (naturally, limited in scope) and enable the participating students who execute the assignments to develop CEST. During a 14-week semester, the students are given four design projects, each relating to a different aspect of mechanical engineering, beginning with a challenge presented in class, requiring the construction of a device that should perform a specific task. This challenge includes an explicit procedure for measuring the effectiveness of the device, based on its performance. Working in small teams, the students are given three weeks to complete each design and build a device. In recent years, an increasing number of academic programs for systems engineering have been developed [INCOSE, 2003]. Many of these programs are intended for graduate students or engineering practitioners. The second study was conducted during two semesters in which two classes of about thirty students each, took the course. Many class observations were performed, and specific teams with "strong", "weak" and "neutral" performances were rationally chosen for interviews. Overall, there were eleven team interviews. Additional data was collected in semi-structured interviews with the teacher and the teacher assistant, and by studying the students' submissions.

The third study, which is now drawing to a close, was actually follow-on of the first study. As such, it can be considered as the second phase of the first study. The idea was that the findings of this phase would strengthen (or weaken) the conclusions of the first study. The primary aim was also to identify the characteristics of engineers having high CEST (according to the assessment of peers and supervisors). Forty-six systems engineers, from all levels, participated in this phase. The data collection tool was the same questionnaire used in the first study (with minor updates). The difference between the two was that in the first study the survey was written but in the second phase it was done by personal interviews. All the data collected from the interviews was transcribed and the raw data analyzed by 'content analysis' – "it defines the analysis units and establishes **the categories** (outstanding repeated elements) for analyzing the raw data" [Silverman, 2000]. According to Denzin & Lincoln [1994: 358], "Qualitative researchers study spoken and written records of human experience, including transcribed talk, films, novels, and photographs… Documents of experience can be content analyzed;

that is, themes, issues, and recurring motifs within them can be isolated, counted and interpreted… This reading can be supplemented by the semiotic method, which searches for oppositions, **categories**, and linguistic structures in the text". Strategy of triangulation was employed in the first and third studies, i.e., categories not found in at least three interviews or in three different data collection techniques were omitted.

3. GENERAL COGNITIVE CHARACTERISTICS

Table I presents the general cognitive characteristics of engineers with high capacity for engineering systems thinking, arranged in descending order, ranked according to the weighting of the score they received in the first study's questionnaire and the frequency of their appearance in the appropriate category in the interviews and observations conducted in the first and third studies. An empty cell indicates a characteristic that was not referred to in the questionnaire.

Table I. General Cognitive Characteristics of Engineers with High CEST

Rank	Characteristic	Questionnaire N=276	Interviews	
		Score (1-5 scale)	Frequency	% (of 77)
1	Understanding the whole system and seeing the big picture	4.23	62	81
2	Understanding interconnections; closed-loop thinking	4.22	43	56
3	Understanding systems synergy	4.32	34	44
4	Understanding the system from multiple perspectives	4.26	26	34
5	Thinking creatively	4.24	24	31
6	Understanding systems without getting stuck on details; tolerance for ambiguity and uncertainty	4.25	22	29
7	Understanding the implications of proposed change	3.85	14	18
8	Understanding a new system/concept immediately upon presentation	3.74	12	16
9	Understanding analogies and parallelism between systems		9	12
10	Understanding limits to growth		8	10

3.1. Understanding the Whole System and Seeing the Big Picture

According to the studies' findings, engineers with high capacity for systems thinking understand the entire system and have the ability to see the big picture. They understand the whole system beyond its elements/sub-systems/assemblies/components and recognize how each element/sub-system/assembly/component functions as part of the entire system. They also grasp how sub-systems integrate into a single whole single system aimed at fulfilling predetermined requirements and specifications. They know how to consider the widest aspects of the system and the environment in which it should perform.

Indeed, many authors refer to the necessity of seeing the whole in the process of problem solving. For example, according to Kim [1995], Waring [1996], and O'Connor and McDermott [1997], a problem may not always be solved by breaking it down into elements and finding a separate solution for each of these elements. One cannot predict the properties of a complete system by taking it to pieces and analyzing its parts. According to Senge [1994: 68], "Systems thinking is a discipline dealing with seeing the whole. It is a framework for scrutinizing mutual relations and interconnections more than the chains of linear cause-and-effect, for discerning change patterns and repeating appearances". In a recent study conducted by Davidz [2005], the item "think broadly/see the big picture" is ranked first in the list of "individual characteristics for systems thinking development".

3.2. Understanding Interconnections; Closed-Loop Thinking

CEST involves an understanding of the interconnections and the mutual influences among system elements/sub-systems/assemblies/components/parts. It was found that engineers with high CEST have the ability for a closed-loop (circular) view of causality rather than straight-line (linear) thinking. When system thinkers are asked to analyze the possible reasons that cause a given system failure, they usually provide several possible explanation and not only one. They do not simply submit a list of likely reasons (sometimes dubbed a "laundry list") but make it plain that they understand that the effect usually feeds back to influence one or more of the causes and that the causes themselves affect each other.

3.3. Understanding Systems Synergy

General systems theory holds that all systems are similar in certain ways. According to this view, if synergy exists in all systems, then it certainly exists in man-made technological systems. The systems engineer must, therefore, be capable of deriving the synergy of a system from the very integration of the subsystems under his/her responsibility.

For example, in many models of advanced fighter aircraft, there is a system that allows pilots to see the entire fight arena. The system is based on a communications network that transfers position and radar data among aircraft in the same arena. The integration of the communications system and the radar system can create synergy – capabilities and functions of an electronic warfare system. Having done this, it provides the capacity to continue getting a radar picture even under jamming and allows weapons from one aircraft to be guided by another's sensors and controls. According to our study's findings, engineers with high CEST are able to identify the synergy and emergent properties of combined systems.

Many authors discuss the issue of synergy (sometimes called emergent properties). For example, according to Kahn and Katz [1966], Whitner [1985], Waring [1996], and Hitchins [2003], a system is more than just a collection of parts. The system properties, capabilities, and behaviors certainly emerged from the system parts but they also appear from the interactions among these parts. The properties of the system consist of more than the aggregate of the proportion of the individual parts and interactions. The whole is more than the sum of its parts. Systems have emergent properties that are not found in their parts. This synergistic effect is one of the central and most important attributes of a system, but it is, at times, hard to identify.

3.4. Understanding the System from Multiple Perspectives

The studies' findings showed that a successful systems engineer would avoid adopting a one-dimensional view. For example, one might look at a system once from the perspective of its operating system and then from the hardware point of view; or a system may be viewed once in terms of its functional aspect and once from its architectural aspects. A successful systems engineer would be able to describe a system from all relevant perspectives that go beyond the mere engineering level: the upper layer—technological, economic, managerial,

and social perspectives; the systems layer—electrical, mechanical, and operational perspectives; and the engineering layer—the ability to examine a specific subject or problem from different angles and points of view, to see it from multiple perspectives.

3.5. Thinking Creatively

Although the correlation between systems thinking and creative thinking was not calculated in the framework of the three studies, it was very interesting to find out that engineers with high CEST are capable of creative-lateral-divergent-heuristic thinking in the raising-ideas stages of a project, and logic-convergent-algorithmic-analytical thinking in the implementation stages. As such, successful systems engineers contribute much to brain-storming discussions. They are able to offer workable creative/innovative original solutions, and transform a creative concept into a realizable idea.

3.6. Understanding Systems without Getting Stuck on Details; Forest Thinking; Tolerance for Ambiguity and Uncertainty

It was found in the three studies that engineers with high CEST are able to conceptually and functionally understand the system, even without understanding all its minutiae. They avoid getting snagged by the details. They are able to understand the whole/overall picture and continue to act without understanding fully all of the system's details. Such engineers feel comfortable with ambiguity and working in unclear conditions and in an uncertain environment; not knowing all the details does not disturb them or hinder their efforts to solve a systems problem. This thinking skill enables senior managers to set priorities and allocate resources on the organization level.

In opposition, engineers who have to thoroughly understand all the details involved in a given problem in order to be able to form a decision and come up with a solution, usually find it difficult to develop high CEST and work as senior systems engineers.

According to Richmond [2000], "forest thinking" involves a "view from 10,000 meters rather than focusing on local trees" and "considering how the system influences systems on the other side of the line and how these latter systems influence the former system". In the study conducted by Davidz [2005], the item "think out-of-

box" was ranked seventh in the list "individual characteristics for systems thinking development" and the item "tolerance for uncertainty" ranked fifth.

Forest thinking allows systems engineers to distinguish between the major and minor, relevant and irrelevant dimensions of a given system/issue. Engineers with high CEST are able to simplify – to filter out redundant, unnecessary information and "noise" in order to clear up and understand the picture.

3.7. Understanding and Seeing the Implications of Proposed Changes to the System

It was found that changes and modifications are central to the work of systems engineers. The systems engineer must understand the system as a whole and be capable of anticipating and detailing all implications (including side effects) of changes in the system — engineering and non-engineering alike — both those initiated by the contractor and those required by the customer (sometimes even after freezing the design). The systems engineer has to be able to assess the impact of a proposed change on the Form, Fit and Function of the system under consideration. In many cases, the systems engineer must be able to take care of every step of implementing the change – from identifying the need, through coordinating with all the involved departments and disciplines and preparing its paperwork (Change Request, Specification Change Notice, and Engineering Change Proposal), up to the approval of the customer and the Configuration Control Board.

According to McDermott and O'Connor [1997: 14], a change involving the addition of a component to the system raises the complexity of the system to be handled: "New interconnections among parts of a system add complexity, and adding another piece can create many new connections. When you add one new piece, the number of possible connections does not increase by one. It may increase exponentially…"

3.8. Understanding a New System/Idea/Concept Immediately Upon Presentation

It was found that engineers with high CEST understand and are able to describe the operation, purposes, applications, advantages, and limitations of a new system/sub-system/idea/concept immediately after receiving an initial explanation. As one of the interviewees in the first study put it: "another thing characterizing engineers

with the ability to engage in systems thinking ... whenever you present them with a totally new system they manage almost immediately to evolve a conceptual grasp …"

3.9. Understanding Analogies and Parallelisms between Systems

General systems theory asserts the "similarity of all systems". Our studies found that engineers with high CEST are able to compare and draw parallels between different disciplines. They are able to learn, infer, and draw conclusions from one discipline and apply these conclusions to another. They know how to relate and find resemblances and common ground between different disciplines, and project from their prime specialization to other disciplines related to the project/system under consideration.

3.10. Understanding Limits to Growth

It was found that when engineers with high CEST are asked to indicate possible ways to improve performance they take into account factors and processes limiting and balancing the performance growth. For example, when the systems engineers that participated in the first study were asked to suggest ways for improving the effective output power of a given data transmitter, they offered methods such as increasing the amplification but also related to factors such as the power supply, antenna, cooling system and mutual interferences of other systems in the area.

4. ABILITIES

Table II presents the abilities of engineers with high capacity for engineering systems thinking arranged in the order in which the ability appeared in the system life cycle. Likewise, the table shows the score received in the first study and the frequency of its appearance in the appropriate category in the first and third studies. An empty cell in the score column indicates a characteristic that was not mentioned in the questionnaire,

Table II. Abilities of Engineers with High CEST

Phase	Ability	Questionnaire N=276	Interviews	
		Score (1-5 scale)	Frequency	% (of 77)
Concept Exploration and Program Definition	Analyzing the need	4.56	42	55
	Analyzing/Developing the concept of operations	4.45	33	43
	Requirements analysis	4.41	55	71
Solution Definition and Preliminary Design	Conceptualizing the solution			
	Generating the logical solution – functional analysis	3.65	28	36
	Generating the physical solution – architecture synthesis	3.25	24	31
All Phases	"Seeing" the future	3.85	34	44
	Using simulations and systems engineering tools	3.53	30	39
	Optimizing	3.28	48	62
	Using systems design considerations	3.07	28	36
	Conducting trade studies	3.04	18	23

4.1. Abilities in the Concept Exploration and Problem/Program Definition Phase

4.1.1. Analyzing the Need

It was found that engineers with high CEST understand and are able to analyze customer (or market) needs, constraints, limitations, priorities, and capabilities. They see the big picture and understand the mission and tasks of the customer organization. Sometimes systems engineers are required to understand the needs even better than the customer.

4.1.2. Analyzing and/or Developing the Concept of Operations

Engineers with high CEST understand and are able to analyze and/or develop the operational concept (CONOPS - concept of operations), goal, objectives, and uses of the required system (or the solution it could provide); or in other words, what the system should do and what it will do. Even at the early stages of concept development, caution must be taken to refrain from setting over-extended goals and objectives. It was found that engineers

with high CEST understand the operational environment and how a given system functions alongside its operator/s. They recognize that the human element is part of the system and are careful not to propose impractical designs.

4.1.3. Requirements Analysis

According to the Systems Engineering Handbook [INCOSE, 2004] requirements are the foundation of the project. They form the basis for design, manufacture, test, and operations. It is essential that a complete, but minimum set of requirements be established early. The objective of requirements analysis is to identify and express verifiable requirements that state user needs in appropriate terms to guide system concept development.

In the three studies it was found that engineers with high CEST understand the requirements and are able to perform requirement analysis including capturing source requirements, defining requirements, formulating requirements, ensuring that each requirement does not sub-optimize the system, generating System Requirements Documents (SRD), "translating" the concept of operations and the requirements into technical terms and preparing system specifications, validating the requirements, tracing the requirements, and ensuring that all needs, goals and external interfaces (context diagram) are covered by the requirements. Engineers with high CEST are also able to lead the process of requirements allocation and decomposition to lower levels.

4.2. Abilities in the Solution Definition and Preliminary Design Phase

4.2.1 Conceptualizing the Solution

It was found that engineers with high CEST are able to generate, using a top-down approach, a conceptual end-to-end solution, prior to any detailed design or analysis. Many (junior) engineers find it difficult to adopt a top-down approach because traditional education emphasizes the bottom-up approach. For example, the sequence of the courses in a typical curriculum for communications engineering is: mathematics, science (physics, etc.), electricity basics, components, circuits, modules, basics of transmission and receiving, sub systems, communications systems. There are some B.Sc. programs in which the stage of dealing with a complete system is never reached. However, in our studies it was found that systems thinkers use top-down approaches successfully for conceptualizing solutions. In recent years we have begun seeing a growing number of undergraduate systems engineering programs that also stress systems aspects.

4.2.2. Generating the Logical Solution – Functional Analysis

The first step of the preliminary design is functional analysis. Engineers with high CEST are able to generate the logical solution by conducting functional analysis. They are able to "translate" the requirements and the system specifications into a Functional Flow Block Diagram (FFBD), in which each block represents a determined function. They are also able to allocate/decompose the functions down to the system elements, sub-systems, assemblies and components. The process of generating the FFBD requires them to determine the interfaces and flows among the blocks/functions.

4.2.3. Generating the Physical Solution – Architecture Synthesis

System architecture creation is defined as the selection of the types of system elements, their characteristics, and their arrangement. Engineers with high CEST are able to generate the physical solution and to design the architecture of the system. The process of designing the architecture involves allocating the functions to physical "boxes" – hardware and/or software configuration items (CI), and determining the physical location of each CI. They are also able to allocate/decompose the system elements down to sub-systems, assemblies and components.

The process of designing the architecture is actually a synthesis process. Many (junior) engineers find it difficult to synthesize because traditional education emphasizes analysis processes. According to Bloom's taxonomy, analysis is the disassembly of a system to its components, while retaining the components' interconnections, with the purpose of analyzing its operation. Synthesis, on the other hand, is the combination, connection, arrangement, organization, and assembly of elements, sub-systems and parts into a whole with the purpose of creating a system that did not exist before.

It was found that CEST means the ability to analyze and synthesize. Engineers with high CEST are able to move from the whole to the parts, and analyze the system by breaking it down into its components. They are able to track signals from the input through every sub-system, and interface to the output. In addition to this, however, successful systems engineers are able to synthesize. They are able to assemble or connect sub-systems into a complete system and provide end-to-end solution.

Usually, the preliminary design is subject to approval in the Preliminary Design Review (PDR). It was found that successful systems engineers are able to prepare, actively participate and lead PDR (all or in part).

4.3. Abilities in All Phases of the System Life Cycle

4.3.1. "Seeing" the Future

It was found that engineers with high CEST are able to "forecast, anticipate and foresee the future". For the long-term they demonstrate technological vision and are able to analyze future customer/market needs and technological developments. For the mid-term they are able to analyze the system life cycle from the first idea to the disposal and take into consideration the various possible solutions following the system going out of service. For the short-term they are able to analyze expected and/or unforeseen difficulties, problems, risks and change requests.

4.3.2. Using Simulations and Systems Engineering Tools

Engineers with high CEST are able to use, design and build simulations (including simulators, models, mock-ups, simulation running on standard work station, etc.). They are aware of the advantages and limitations of simulation. They are able to define, use and build systems engineering tools. This is a very broad topic but due to the limited scope of this paper, it will not be elaborated on any further.

4.3.3. Optimizing

Searching for an optimum solution is a key activity in systems engineering. The notion of "optimum" is related to the notion of "trade-off". For example, the process of system architecture synthesis is essentially a *trade-off* leading to the final architecture, selected from among a set of candidates, as a final output, assuming that at least one of the candidates is close to the true *optimum*. Indeed, the definition of optimum includes satisfaction of all requirements and factors such as acceptable design maturity, compatibility with the development schedule, minimum cost, acceptable risk, etc. Similarly, the process of generating a set of element options should include a range of well proven items to allow trade-offs among cost, performance, development, time and risk [INCOSE, 2004: 142].

Engineers with high CEST understand optimization considerations and are able to decide when to trade-off and compromise in three dimensions. The first dimension refers to engineering optimization. Engineers with high CEST are familiar with engineering constraints, are able to mediate conflicts of interest between the different engineering disciplines, and know how to refrain from over-designing. An example found in one of the studies was resolving conflict requirements between buffer size and bandwidth. The second dimension refers to cost and schedule. Engineers with high CEST are able to apply cost-effectiveness, cost-benefit, and design-to-cost considerations. The process of optimization requires trade-off analyses and must take into consideration various conflicting constraints such as performance, cost, time, available resources, engineering constraints and more. The third dimension involves engineering and operational considerations – how to achieve the best possible operational performance with the least possible engineering efforts, and stay within project cost and schedule.

4.3.4. Using Systems Design Considerations

Engineers with high CEST are able to apply systems design considerations. Examples found in the studies are: generating a Built-In-Test (BIT) concept, comparing decentralized and centralized design options, offering client-server architecture, considering interface and dependence minimization, considering re-use of already developed configuration items, and using appropriate standards.

4.3.5. Conducting Trade Studies and Providing Several Alternatives

Trade studies provide an objective foundation for the selection of one of two or more alternative approaches to solution of an engineering problem [INCOSE, 2004]. It was found in the studies that engineers with high CEST are able to successfully take part or lead trade studies and feasibility studies. In the trade study report they usually offer a number of alternative solutions (and not simply the first and/or only one that comes to mind). For each alternative the engineering implications are detailed, along with the estimated schedule and the required resources. They are able to compare the alternatives and, by using various techniques, to point out the optimal solution.

All the systems engineers who participated in the three studies realized that any given engineering problem may have several alternative solutions. However, the more successful systems engineers were able to consider many solutions in parallel as an ensemble of different options, and could easily handle the temporary ambiguity and uncertainty in their design process. In contrast, the less successful systems engineers considered their different alternative solutions sequentially, and when they encountered a difficulty in one option, they prematurely dismissed it and turned to the next-in-line alternative, without further consideration.

5. BEHAVIORAL COMPETENCES AND INDIVIDUAL TRAITS

Table III presents the behavioral competences and individual traits of successful systems engineers, sorted in descending order by the frequency of their appearance in the appropriate category discussed in the interviews and observations carried out in the first and third studies.

Table III. Behavioral Competences of Successful Systems Engineers

Rank	Behavioral Competency/Individual Trait	Frequency	% (of 77)
1	Management Skills	66	86
	Team Leader		
	Building and Controlling the Work Plan		
	Defining Boundaries		
	Taking into Consideration Non-Engineering Factors		
	Additional Management Skills		
2	Good Human Relations; Team Player; Good Communication Skills; Good Interpersonal Skills	40	52
3	Autonomous and Independent Learner; Strong Learning Skills	38	49
4	Willing to Deal with Systems	34	44
5	Curious, Innovator, Initiator, Promoter, Originator	28	36
6	Ask Good Questions	26	34

5.1. Management Skills

It was found that successful systems engineers have the following management skills (all or part):

5.1.1 Team Leader

The studies found that successful senior systems engineers aspire to manage and are able to lead interdisciplinary teams (sometimes called IPDT – Integrated Product Development Teams). This task includes: building and operating the team, leading the team through all development stages, chairing the team meetings, goal setting, decision making, conflicts resolving and motivating the team members. Successful system engineers are able to bring together into cooperative professional activity expert systems engineers from different disciplines. For this purpose they need interdisciplinary knowledge, organizational and management capabilities, and an ability to integrate disciplines.

Teamwork is not a natural process arising from a meeting of a group of people. Rather, it is an initiated process that requires organizational activities and specific procedures over a period of time. The term 'organizational activities' relates to activities such as distributing the team calendar in advance, writing up protocols of meetings, defining the preparations needed before each meeting, characterizing the ways of distributing the reports, nominating a leader, and determining the structure of the meetings (e.g., reviewing the status of previous decisions, presenting a new subject, holding discussions, summing-up, making decisions). "Specific procedures" refer to activities such as how decisions are taken (vote, consensus, by the chairman, etc.), and how to cope with conflicts. Successful systems engineers are able to lead all these activities.

5.1.2 Building and Controlling the Work Plan

It was found that successful systems engineers are able to bring together all activities under their responsibility to successful completion of the mission/project. This includes tracing and controlling the missions/projects under their responsibility by building, controlling and managing a work plan containing a tasks list, schedule, milestones, design reviews, status meetings, activities for identifying bottlenecks, etc.

5.1.3. Defining Boundaries

It was found that successful systems engineers know how to set up proper boundaries and allocate tasks to the various teams, departments, partners, sub-contractors and out-sourcing contractors in a way that each of them finds it possible to cope with its missions and tasks.

5.1.4. Taking into Consideration Non-Engineering Factors

It was found that successful systems engineers are aware that sometimes, when preparing proposals or designing solutions, one must bear in mind non-engineering consideration such as ecological/environmental, marketing, "political", organizational, economic, and personal issues, personality styles, business, re-use, opportunities and different viewpoints considerations.

5.1.5 Additional Management Skills

It was found that successful system engineers are able to resolve technological, organizational, "political", personal and logistics problems, control all personnel involved in the mission/project (internal or external to the organization), make successful presentations and successfully WIN-WIN negotiate.

5.2. Good Human Relations; Team Player; Communication Skills; Interpersonal Skills

Complex systems and issues often require several engineers to cooperate in order to grasp and comprehend all aspects of the system. By its nature, systems thinking calls for the review of multiple perspectives, and so teamwork is often required. In most cases group decisions, expressing the various perspectives of a team's members, are better than individual decisions [Parker, 1990]. Teamwork requires interpersonal skills such as communication skills, negotiation skills, and an ability to cope with conflicts. Employers frequently identify the lack of team skills as a critical lacuna in the preparation of engineering students [Prince, 2004].

In the three studies it was found that successful systems engineers are characterized by good communication and interpersonal skills and are able to collaborate with others, are strong team players, and establish trusting relations with customers, colleagues, partners, vendors, and sub-contractors. In a study conducted by Davidz

[2005], the item "strong communication skills" is ranked fourth in the list "individual characteristics for systems thinking development" and the item "strong interpersonal skills" is ranked eighth in this list.

5.3. Autonomous and independent Learner; Strong Learning Skills

It was found that successful systems engineers are capable of autonomous and independent self-learning. They are willing to take responsibility for acquiring new knowledge. They are capable of learning whatever is required for the job, including know-how from other disciplines; they know how to let others assist them, and usually they participate in the "learning organization" activities. They read a lot, and keep up-to-date in the pertinent fields, ensuring that new, relevant information reaches them.

5.4. Willing to Deal with Systems

Analysis of the first study's findings revealed that not every engineer aspires to be a systems engineer and to deal with systems. Some engineers prefer to be research and development engineers and conduct detailed, in-depth work. The first condition for being successful in a position that requires CEST is to have the desire and interest to work with systems and to "love" working on the systems level and with its respective issues. It was found that senior systems engineers seek to look at the whole picture and not at the details.

5.5. Curious, Innovator, Initiator, Promoter, Originator

It was found in the three studies that engineers with high CEST are curious and open-minded, and usually have broad interests, beyond the limited area of their expertise. They are uninhibited and do not find it difficult to think under pressure. They do not "freeze up", nor are they "afraid" to think and dare. They are willing to cope with (new) areas outside their area of expertise. In the study conducted by Davidz [2005], the item "curiosity" is ranked second in the list of "individual characteristics for systems thinking development" and the item "open-minded" is ranked third in the same list.

It was found that engineers with high CEST do not stop thinking and initiating. They are constantly on the watch for what more can be done and where additional/new opportunities may emerge. They always investigate issues that are related to their current work, and how their work can be improved and expanded. They are willing to

take risks. They see failures, problems, and screw-ups as challenges and opportunities for development, not as "the end of the road."

5.6. Ask Good Questions

It was found that successful systems engineers constantly question information they are given. It has been shown that the ability to ask good questions is a managerial tool that helps oversight and enables the asker to see the whole systems picture. The significance of this issue can be learned from the findings of the pilot questionnaire distributed in the first study. The average grade that the item "to what degree, in your opinion, can CEST be acquired/learnt/developed by learning to ask good questions?" was 4.01, with a mean standard deviation of 0.82 on a scale of 1-5.

Richard Feynman, the 1965 Nobel Prize winner for physics and a member of the Presidential Inquiry into the Challenger Explosion of 1968, also spoke of the asking of questions as an effective tool for getting information:

"This is the only way that I know of getting technical information in a hurry: you don't sit and listen…instead you ask a lot of questions…and learn exactly what to ask…I got an exceptional education on that day [Feynman, 1988: 121].

Likewise, O'Connor and McDermott [1997: 138] recommended developing question asking skills:

"...The other way to get feedback is by questions. Good questions get good feedback. The more precise and focused the question, the more useful the answer. So cultivate the art of asking good questions".

In the study conducted by Davidz [2005], the item "ask questions" is ranked sixth in the list "individual characteristics for systems thinking development".

6. ADDITIONAL ROLES OF (JUNIOR) SYSTEMS ENGINEERS

This section presents additional roles of systems engineers found in the first and third studies. Although some measure of systems thinking is required to perform these tasks, the opinion of most systems engineers participating in the studies was that they can be performed by junior systems engineers. Indeed, it was found in the studies that the roles in this list are related, usually, to junior systems engineers.

The additional roles are:

Testing – to take part in (or to lead) the following stages of the project: integration, verification, validation, evaluation, qualification, to be familiar with standard and specific diagnosis tools and test equipment (including FTE, ATE, etc.); to lead "by similarity" analyses.

Interfaces – to design interfaces; to be proficient in handling the hardware and software input and output of units under their responsibility; to be familiar with various types of interface components and standards.

Wiring – to design wiring and/or to examine wiring designs and drawings; to be proficient in dealing with types of wires, cables, optic fibers, connectors, plugs, bonding, shielding, etc.

*Installation*s – to design installations and/or to examine installation designs and drawings (both mechanical and electrical); to be familiar with installation considerations.

Documentation – to take part in (or to lead) preparing/analyzing engineering documents including Requests For Proposals (RFP), proposals, hardware/software development specifications, Interface Control Drawings (ICD), test specifications and various technical reports.

Supporting and Coordinating – to support the production line; to support the various operational stages and the end users (as part of the maintenance effort); to coordinate development teams' activities during the detailed design phase and other phases.

Resolving System Failures or Problems – to remedy/resolve/analyze system failures or problems; to be familiar with system malfunction/failure analysis tools such as Fault Trees, Failure Modes and Effect Analysis (FMEA), and Structured Analysis and Systems Design (SASD).

Standardization – to design to (and be familiar with) the standards; to carry out a design compatible with adjacent systems; to carry out a modular design by integrating open sub-systems and/or existing products; to be familiar with standard communication protocols; to ensure compatibility with commercial products and standard software tools and platforms.

Quality Assurance and Risk Management – to support quality assurance activities and ISO and CMMI efforts; to take part in (or to lead) the risk management activities.

7. KNOWLEDGE

In the opinion of most systems engineers who participated in the first and third studies, the single most prominent characteristic of engineers with high CEST is wide, diversified, multidisciplinary and interdisciplinary knowledge. Usually, successful systems engineers have a B.Sc. in at least one main area (an "anchor") and become specialists in this field. However, in addition to this, they acquire knowledge in additional areas. The knowledge in these other areas need not be equal to that of a specialist.

In the additional areas successful systems engineers possess general knowledge and understanding. Over the years they become familiar with the jargon and professional language of the other disciplines (relevant to their job and tasks), are able to communicate with people or experts from different fields and disciplines and ask intelligent relevant questions. According to O'Connor and McDermott [1997: 4], "Therefore you can make predictions about the system behavior without knowing the parts in detail.... Instead of seeing separate fields of knowledge, all needing years of study to understand, systems thinking lets you see the connection between different disciplines."

The findings obtained from the respondents in the first study clearly point to the need for interdisciplinary knowledge. On the pilot questionnaire, the mean grade of the item "To what extent is the systems engineer required to have multidisciplinary knowledge" was 4.10, with a standard deviation of 0.79, on a scale of 1–5. In the final questionnaire, the interviewees were asked the following: "In addition to extensive knowledge in the area of your main specialty, to what extent do you need knowledge (perhaps less extensive) in additional

disciplines, at least on the level enabling you to ask intelligent questions, understand (on the top level) what the issue is and be able to communicate with colleagues from different disciplines?" The mean grade to this response was 3.7 on a scale of 1–5.

The majority of the electrical/electronics systems engineers who participated in the first and third studies indicated that they were required to possess general knowledge in the following additional areas of electrical/electronics engineering in addition to their expertise in a sub electrical engineering area: programming, software/hardware integration, computer networks, signal processing, control, microelectronics, and electro-optics. Systems engineers dealing with airborne systems reported that they also need general knowledge in the different areas of avionics. Furthermore, they were required to acquire general knowledge in the following additional disciplines: industrial engineering and management (project management, business administration, economics, pricing, marketing, financing, human engineering, operations research, production engineering, human resource management), mechanical engineering (design, mechanical packaging, thermal design), specialist engineering (quality assurance, reliability, ISO standards, electromagnetic compatibility, etc.), computer science (software engineering, knowledge on a level that would enable them to evaluate the content of a programmer's job – hours of work, costs, and time, familiarity with considerations for selection/planning computer peripheral components or equipment), support engineering and ILS – Integrated Logistics Support (maintenance and support principles, levels of maintenance, service measures, spares, long-lead spares, inventory, inventory turnover, inventory levels, inventory consumption, manuals, technical handbooks, technical training).

8. CONCLUSION

From an analysis of the results, it would appear that engineers with certain personal characteristics can acquire or enhance their CEST in a gradual and long term process, by appropriate training, job rotation, systems work roles, job design and development programs. Executing a wide range of jobs enables the engineer to get acquainted with many systems and technologies, as well as acquire broad experience, learn from others' experiences and work on teams with engineers who have systems engineering thinking and are dealing with areas that require CEST. Nevertheless, work tenure and experience are not enough for an engineer to gain CEST. There are many veteran engineers who have never attained it. Many of those who participated in the studies said that, often, an engineer with potential high CEST will demonstrate it even in the first two or three years of their first position. Moreover, in the second study it was shown that the capacity for engineering systems thinking can be developed

in the early stages of engineering education. These two findings bear witness to the fact that there are additional factors that allow engineers to acquire CEST – innate potential is probably one of them.

During these studies, it became clear that a quantitative tool for assessing CEST is currently not available, and that such a tool is needed. Such a tool may also be used for classification and selection of engineers in industry and government, as well as engineering students. Based on the results presented here, a follow-on study that includes quantitative measurements is being designed. To this end, a test for assessing the capacity for engineering systems thinking is currently being developed.

REFERENCES

H. Davidz, Accelerating the development of senior systems engineers, paper presented at INCOSE 2005, the 15th International Symposium of the International Council on Systems Engineering, Rochester, New York, 2005.

N.K. Denzin, and Y.S. Lincoln, Handbook of Qualitative Reseasrch. Sage Publications, Thousand Oaks CA, 1994.

INCOSE, Systems Engineering handbook, Version 2a, Technical board, INCOSE, 2004.

INCOSE, Academic Forum 2003, INCOSE International Symposium, Washington DC, July, 2003.

R.P. Feynmam, What Do You Care What Other People Think? Bantam Doubleday Dell Pub, New York, 1988.

M. Frank, Characteristics of engineering systems thinking: A 3-d approach for curriculum content, IEEE Transaction on Systems, Man, and Cybernetics, Part C, 32 (3) (2002), 203-214.

M Frank, and D. Elata, Developing the capacity for engineering systems thinking (CEST) of freshman engineering students, Systems Engineering, 8 (2) (2005), 187-195.

D.K. Hitchins, Advanced Systems Thinking, Engineering and Management. Artech House, Boston MA, 2003.

D. Katz, and R. Kahn, "Organization and the system concept," in: Social Psychology of Organizations. pp. 14-29, John Wiley and Sons, New York, 1966.

D. H. Kim, Systems thinking tools, Pegasus, Cambridge, 1995.

J. M. Morse, The significance of Saturation, Qualitative Health Research 5 (1995), 147-149.

J. O'connor, and I. Mcdermott, The art of systems thinking, Thorsons, San Francisco, 1997.

M.G. Parker, Team players and team work, Prentice-Hall, New York, 1990

M. Prince, Does active learning works? A review of the research. Journal of Engineering Education, 93(3), (2004), 223-231.

B. Richmond, The "Thinking" in Systems Thinking, Pegasus, Waltham MA, 2000.

P.M. Senge, The fifth discipline: The art and practice of the learning organization, Doubleday, New York, 1994.

D. Silverman, "Analyzing talk and text," in Handbook of qualitative research, 2nd edition, N.K. Denzin, Y.S. Lincoln (Editors), Sage Publications, Thousand Oaks, CA, 2000, pp. 821-834.

A. Waring, Practical systems thinking, Thomson Business Press, Boston, 1996.

P.A. Whitner, Gestalt Therapy and General System Theory, The University of Toledo, Ohio, 1985.

Systems engineers are from Mars, software engineers are from Venus; how to help them communicate[3,4]

Joseph Kasser and Sharon Shoshany

הקדמה

מאמר זה נכתב במקור באווירת קפיטרית קמפוס אוניברסיטאי נינוחה, באדלייד הנמצאת בדרום אוסטרליה. אני הייתי בשבתון, לאחר סיום תפקיד כראש מחלקת פיתוח תוכנה ולכן מטבע הדברים ייצגתי את אנשי התוכנה. גו' קאסר ייצג את מסורת מהנדסי המערכת (במציאות העכשווית ג'ו מרצה במרכז להנדסת מערכת בדרום אוסטרליה, עם רקע מקורי בהנדסת חשמל ואני ראש מנהלה עם רקע מקורי ב..הנדסת חשמל).

רק לאחרונה גיליתי כי המאמר בעצם עוסק בנושא החם ביותר בהנדסת מערכת כיום - תרבויות - או פערי תרבות. הנושא פופולארי הן בזכות פרויקטים חובקי עולם בהם מהנדסי מערכת נדרשים לגשר על פער תרבויות מסוג אחד, והן בזכות המולטידספלינריות השלטת בפרויקטים גדולים עכשוויים וגוררת התמודדות בפערי התרבות של כל הדיסציפלינות המעורבות.

הפער הקלאסי איתו ניסינו להתמודד היה בין מהנדסי התוכנה למהנדסי המערכת - לכאורה הנדסות שרב המשותף בצורת החשיבה הנדרשת בהן, אך בעצם קיים ביניהן פער תקשורת בסיסי (כמו בספר המפורסם - נשים מוונוס וגברים ממאדים).

מאז עברו מים רבים בירדן ובטורנס הזורם במרכז אדלייד, והמאמר הורחב כך שיעסוק לא רק בפער, אלא באיך ניתן להתמודד איתו.

בברכה
שרון שושני

This article views the organisation from the perspective of its culture – the people who comprise the organisation and how they communicate because communications is a vital ingredient to the success of any enterprise. Once work is split up between different people the need to communicate shows up. One of the many barriers to successful communications is the background of each individual which assigns different meanings to words. Consider the word “secure”. When told to “secure” a building it has been related that in the USA,

- The Navy issues a purchase order for the building.
- The Air Force locks the doors and turns on the alarm systems.
- The Army evacuates the personnel, then locks the doors and turns on the alarm systems.
- The Marines assault the building using ground troops and air support, and then deploy squads in and around the building checking the credentials of all who aspire to enter the building.

This example illustrates a subtle communications problem. When one hears unknown words, such as in a foreign language, the failure to communicate is obvious. However, when one hears words that sound correct in the context, the failure to communicate is not realised and sometimes produces serious consequences. This situation can happen when communications takes place between different organisations, different national cultures and even different engineering specialties as demonstrated in this article. The differences between systems and software engineers has been chosen as a demonstration, because in the last three decades of the 20th Century, a period roughly matching software engineering’s history, the development of complex projects seems to have resulted in massive failures. This article provides some insight that the failure of systems and software engineers to communicate, and more importantly their failure to understand that they are not communicating, may be a hitherto undetected cause of the failures experienced in developing today’s complex projects. The article then explores some of the reasons for the communications failure and recommends an approach for improving the ability of systems and software engineers to communicate.

Five thousand years of experience in hardware engineering have provided its practitioners with methodologies that have, in the main, been successful. It was during that time that projects were engineered. Only in the last three decades of the 20th Century, has the development of complex projects been plagued by massive failures. Yet when seen from a historical perspective the complex projects of the last three decades are no more complex than the large engineering projects of the past, projects like the railroads, canals,

[3] Acknowledgment is made to John Gray Ph.D. for the use of his metaphor.
[4] This chapter is based on - Kasser J.E., Shoshany S., “Systems Engineers are from Mars, Software Engineers are from Venus”, *proceedings of the 13th International Conference on Software & Systems Engineering and their Applications*, Paris, France, 2000, and - Kasser J.E., Shoshany S., “Bridging the Communications Gap between Systems and Software Engineering”, *proceedings of the INCOSE-UK Spring Symposium*, 2001.

pyramids and military sieges, within the constraints of the then-available tools and technology[5]. The growing recognition that the multidisciplinary environment in which today's complex software intensive systems are developed is characterized by poor requirements, poor change management and poorly defined processes resulted in several approaches to improve software development, including:

- The development of Computer Aided Software Engineering (CASE) Tools.
- The focus on process and methodology embodied in the ISO/IEEE standards.
- The United States of America's DOD initiated Software CMM.
- The development of the "domain abstraction concept" in object-oriented Methodologies (OOM).
- The adoption of the UML.
- The development of reusable generic software components.
- The inclusion of software engineers on IPTs.

However the adoption of these approaches will not guarantee that current and future software based complex projects will not fail. This is because the projects that software engineers develop do not exist in a vacuum (Pfleeger, 1998). Software cannot perform its function without an underlying hardware platform. Consequently, software engineers need to communicate with the systems engineers, hardware engineers and the other members of the IPT.

During the course of some research Sharon Shoshany (the software engineer) and I (the systems engineer) realized that a communications gulf existed between systems and software engineers, and tried to identify it. After some discussion[6] we developed a conceptual meta-model of the system development process within the System Life Cycle (SLC). The meta-model, which became our Rosetta stone, summarizes the development process in the following manner: When faced with the problem of meeting the customer's needs:

- Engineers don't always reuse solutions or components that worked in the past; they reinvent them or try to invent new ones.
- Even when engineers try to reuse solutions or components that worked in the past, according to good engineering practice, there are two implementation choices:

1. The problem is similar to other problems that have been solved in the past. Thus this time around, providing a similar solution may solve the problem. The process then becomes one of identifying the applicability of the solutions of the past, to the problem of the present and applying the elements of one or more solutions of the past to solve the problem of the present.
2. The problem is unique so there are no known solutions. The process then becomes one of identifying a solution that makes the maximum use of existing solutions to past problems (components) and the minimum use of components to be developed so as to reduce the risk of non-delivery on time and within budget.

At this time, while we came to the realization that while much of the communication problem was with the semantics; we also identified other underlying barriers including:

- Training and Background Differences
- A lack of respect for the other's profession.
- The use of language
- The role of systems engineer in the SLC.
- Different Concepts.

Before considering each of the barriers to communications it is important to distinguish between a true barrier, and poor application of the methodology since each engineering discipline has both good and bad practitioners. Each barrier in the above list is discussed in this chapter and several examples are cited from the published literature as well as from personal experience.

Training and Background Differences

This section analyses the different training and background of software, systems, and hardware engineers.

Hardware Engineers

The most common background for system engineers is hardware engineering. Moreover, hardware engineering is often used as a reference when describing generic processes. Therefore it is important to understand the background of the hardware engineer. Hardware engineers have been using models and blueprints in the form of schematic diagrams and sketches since the early days of

[5] While we can see the products or outputs of those development processes, little information about cost and schedule overruns or prior failures has survived through the ages.
[6] No blood was shed in the process but we came close.

engineering. They also use a component-based methodology. However, they develop components that are physical in nature, e.g. black boxes, printed circuit boards, integrated circuits and subsystems. When hardware engineers design a digital system, they use integrated circuits. The hardware engineer will pattern match sections of the design to the offerings in the vendor's family of components. They then partition the design to make use of standard components and design a small amount of custom circuitry as required to interface between the standard components. The physical nature of the components themselves enforces the mapping of functionality onto physical components.

Software Engineers

In its early days software development practitioners used two basic approaches. Some developed their own libraries of subroutines or modules for functions, that once debugged, could be used in other programs. The majority however:

- Tried to implement the same functions in various ways in different projects.
- Suffered from the "not invented here" syndrome and would not reuse code from external sources.
- Failed to obtain the benefits of code reuse because they either changed working code or reused the code in a context different from its original one and the difference was the cause of the failure.

However, software engineers have since matured and created several more important methodologies, including

- **Componentware** - Components are large grained entities that are defined by their interfaces (services provided and demanded) and can be independently developed and used (Szyperski, 1997). The component approach promises to turn software development to software assembly in a similar manner to the use of integrated circuits by the hardware engineer.
- **Design Patterns** – a way of expressing general solutions to recurring problems.
- **Abstract modelling** – a systematic way of capturing the system requirements in an abstract, consistent and complete manner.

Software engineers are often in the vanguard when it comes to system changes due to:

- The common misconception that it is easier to make software changes than hardware changes.
- Software is responsible in most cases for the user interfaces – an area constantly requiring changes.

Software engineers are often a product of a three-year undergraduate course of computer and information science. This course seldom teaches math and physics at a level that is considered basic for engineering.

"What do you mean you don't know what a Fourier Transform is" yelled the systems engineer at the software engineer.

With current technology at hand, software graduates rarely come to consider Hardware as constraint or consideration.

"No wonder most IT systems tend to crash at their first field test" grumbled the systems engineer).

In a scan of the indices of books about software engineering and development picked at random from those used for, or considered for use for, teaching software engineering and management at the postgraduate level, the following books identified the term 'systems engineering' in their indices[7]. (Shumate and Keller, 1992; Pressman, 1993; Sommerville, 1998; Donaldson and Siegel, 1997; Brodie, 2001; Pressman, 2005; Endres and Rombach, 2003). The remainder made no mention of the term (Shlaer and Mellor, 1988; Marcotty, 1991; Jacobson, Christerson, Jonsson and _vergarrd, 1993; Arthur, 1993; Yourdon, 1993; Gamma, Helm, Johnson and Vlissides, 1995; Perry, 2000; DeMarco, 1995; Humphrey, 1995; Metzger and Boddie, 1996; Berg, Levia and Rouillard, 1996; Yourdon and Argila, 1993; Budd, 1996; Fowler, 1997; Weinberg, 1998; Bass, Clements and Kazman, 1998; DeMarco and Lister, 1999; Van Vliet, 2000; Britton and Doake, 2003). Even (Peters and Pedrycz, 2000) made no mention of systems engineering in their index. Thus a generation of software engineers seemingly was being taught to engineer software without knowing much about traditional engineering and what function systems engineers perform.

In addition, software engineering which does not directly address systems engineering is also teaching methodologies for eliciting requirements (e.g. Use Cases and business objects). At the same time it is ready to accept the system engineer as a requirement source in the manner of any other stakeholder.

[7] Two of the books are used in the management classes not the engineering classes.

System Engineers

Systems engineers in general have little if any formal training in systems engineering. They graduated to the discipline from another engineering discipline, mostly from hardware; therefore the software engineer claims that they may not have an understanding of the nature of software. In addition, software engineers are assuming the role of the systems engineer in software-intensive systems.

The systems engineering community has recognized the need for formal education and training in systems engineering (Faulconbridge and Ryan, 1999; Kasser, 2000). A scan of the indices of books on systems engineering for the word "software" had mixed results. Software is cited in the index of several books on systems engineering (Rechtin, 1991; Thome, 1993; Eisner, 1997; Kendall and Kendall, 1997; Martin, 1997; Blanchard, 1998; Faulconbridge and Ryan, 2003). And although the requirements for software engineering are discussed in the context of adapting (MIL-STD_2167A, 1998) to systems engineering in (Kasser, 1995), software only shows up in the bibliography section of (Martin, 1997) and fails to show up completely in two books (Buede, 2000; Hitchins, 1993).

"Yes" pointed out the software engineer "while (Buede, 2000) doesn't list software engineering in its index, it borrows a lot of terms and concepts from software engineering".

The lack of mutual respect

IPTs containing engineers from multiple disciplines develop today's complex projects. One of the characteristics of successful teams is respect for each other's specialty knowledge. Yet systems and software engineers, in general tend to have little respect for each other. This section explores the following reasons that system and software engineers fail to respect each other:

- The discipline of engineering
- Poor engineering practice
- The use of language

The discipline of engineering

Many of the software development personnel are not engineers but are programmers (equivalent to technicians). Other software engineers lack a formal background in math and physics which tends to preclude them from fully understanding a system that has more then just software in it. On the other hand, many hardware and systems engineers have some programming experience which makes them believe that they are "software experts". Comments heard in organizations included:

- "I tried using a real time operating system 15 years ago and it didn't work, therefore it shall not be used in my project".
- "This is something that my high school kid can program in a week".

Additionally, systems engineers expect software designs to be yet another direct functional decomposition of the system design, and they get confused when the designs don't look that way.

"Why did they redo my nice object diagram" pondered the confused systems engineer?

Poor engineering practice

Some software engineers feel that systems engineers seldom provide the process-product deliveries expected from them. The software engineer claimed that she knew of several organizations that had to retrain their system engineers when going through the Capability Maturity Model (CMM) process in order to be able to qualify for the Level 2. Other common complaints were:

- "They keep handing us new requirements on the way to the canteen".
- "They fail to uncover all the system behaviour issues in their requirements, concentrating on the static and most apparent ones".

These examples of poor systems engineering lowered the respect for systems engineers in the eyes of the software engineers. On the other hand, from the systems engineer's perspective, Watts Humphrey created a CMM for the individual called the Personal Software Process (PSP) (Humphrey, 1995).

"His process, used by systems and hardware engineers for years, is being treated with wonder by the software engineering community because of the reception that the PSP is receiving" said the system engineer.

"Please don't mention the PSP," she retorted, "The PSP never was an issue in my community and was never actually mentioned or used".

"But listen to this" said the systems engineer quoting *"These techniques (the CMM, PSP, and others like them) apply basic engineering and management principles to software. Over the years, we software practitioners convinced ourselves that software was different. The basics did not apply to us; we needed to break the mould. We were wrong. Software is different in some ways, but it has more in common with other fields than we want to admit. We must use what others have proven works at work*" (Phillips, 1998).

In his eyes this quotation proves what systems engineering has held for a long time, namely, software engineering is one of many engineering disciplines.

The systems engineer then picked up a book on software reusability and read "*In well-established disciplines like civil or electrical engineering, reuse is based on the existence of previously coded knowledge. There are two different levels of reuse to consider: (1) the reuse of ideas or knowledge and (2) the reuse of particular artefacts and components… Electrical engineers, for example, consult component catalogues, check which available part best fits the design constraint, and, in some cases, relax the original design requirements to take advantage of existing components* (Prieto-Díaz, 1987).

He, the systems engineer agrees with the author but thinks that it is poor engineering practice to relax the original design requirements without consulting the customer. He has seen it happen in the world of systems engineering and knows that the correct approach, the relaxation of requirements takes the form of a change request, the after the Configuration Control Board has accepted the request, a sensitivity analysis and an informed decision with the customer. He reads on to discover
"We propose a model for reusability based on these observations and on the assumption that available components usually do not match the requirements perfectly, making adaptation the rule rather than the exception. Our approach is to provide an environment that helps locate components and that estimates the adaptation and conversion effort necessary for their reuse. The reuse process is as follows:
A set of functional specifications is given. The user then searches a library of available components to find the candidates that satisfy the specifications.
If a component that satisfies all the specifications is available, reusing it becomes trivial".
This is good stuff he thinks; software engineering has finally learnt to use components properly. He then reads on and discovers the following text:
"More typically, several candidates exist, each satisfying some specifications. We call them similar components. In this case, the problem becomes one of selecting and ranking the available candidates based on how well they match the requirements and on the effort required to adapt the non-matching specifications. Once an ordered list of similar candidates is available, the re-user selects the easiest to reuse and adapts it".

"Adaptation!" cries the systems engineer, "that's like performing the functional equivalent of drilling into an integrated circuit and attaching a wire internally, instead of designing some external circuitry to interface the wire to the component".

"No" replies the software engineer, "Adaptation does not necessarily mean changing or modifying the component. For example, when you bring a 110 Volt radio from the USA to Australia which uses 220 Volts how do you adapt it?"

"Well you can change the power supply internally, or add a transformer externally" was the reply.

"So the word adapt does not necessarily mean modify!" she said triumphantly.

"Yes" he admitted.

"So why did you immediately associate 'adapt' with 'modify'? She asked.

"Because in my experience, that's what software engineers did, years ago" he replied.

After some discussion they agreed that indeed it was the perception that drove the reality, and even if today's software developers did not modify existing components, the systems engineer's perception, based on his experience, shaped his lack of respect for software engineers.

In addition systems engineers have the perception that software engineering much the same as any other engineering discipline is characterized by poor practice. For example, as the systems engineer said, (Boehm, 2000) states *"the software drives system considerations such as performance and cost. For example, in a recent survey of 16 books on object-oriented design, only six had the word 'performance' in their index, and only two had the word 'cost'"*.

"From the system engineering perspective, requirements drive performance and cost as well as functionality, which in turn drive the software and all other parts of the system. So why should Boehm in the year 2000 have to point out cost and performance to software engineering?" he asked.

She replied "You will not find cost in hardware engineering text books either, simply because the books discuss technology and not management. On the other hand I agree that in some areas of information systems technology performance is not treated with enough respect".

The use of language

Just as a common language opens the door to communication, so too the lack of it erects a barrier not easily overcome (Cox, 1996). However, even with a common language, communications is not guaranteed. While the notion that Great Britain and the United States are separated by a common language (Shaw, 1925) may be known, its ramifications are subtle and it is not an easy concept to understand unless one has been sensitised to it. An example of the situation occurred in the mid 1970's during contract negotiations between the Communications Satellite Corporation (COMSAT) and British Aerospace. The language of the meeting was English. The meeting became stuck on one point. Someone then suggested "tabling the issue". Both sides agreed and the meeting deteriorated. The situation was much improved when the bi-lingual interpreter pointed out that the verb "to table" means in:

- **English** -to place the subject on top of the table for immediate discussion.
- **American** - to place the subject under the table or put it away for later discussion.

Consider the following SLC analogue, concerning the meaning of the word "component".
The systems engineer quoted (Buede, 2000) page 50
"A component of a system is a subset of the physical realization (and the physical architecture) of the system to which a subset of the system's functions have been (will be) allocated".

They agreed that systems engineers live in the physical world and the software engineer also stressed that components are not only physical but are also logical. So they decided to look up the UML 1.3 definition of a component (UML, 1999), and found:
"A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces. A component represents a physical piece of implementation of a system, including software code (source, binary or executable) or equivalents such as scripts or command files".

"So components ARE physical" said the systems engineer, to which she replied "What's physical for us is logical for you".

She explained further "software engineering separates the issue of component (what it does) from both deployment (where it executes) and from its instances (how many times it runs). It creates another layer of abstraction that has more in it than just the physical entity as thought of by systems engineering. "

They then looked up the definition of a component in UML Version 2.0 and found the definition of a component as "*a modular unit with well-defined interfaces that is replaceable within its environment*" (UML, 2005).

"That means it can be physical or logical" she said.

"I think I'm getting a headache" he sighed.

The following day he found a cartoon, shown in Figure 1, which, they both agreed seemed to express the situation and should be consulted before any discussion with a person from another discipline to sensitise the participants to the pitfalls associated with the use of ambiguous wording.

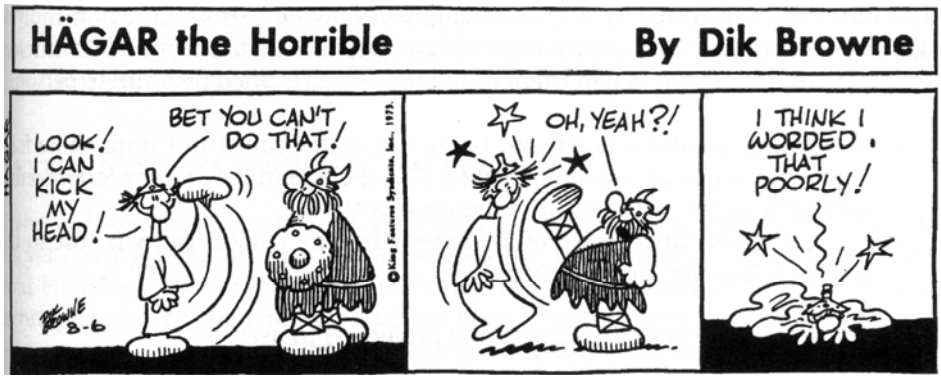

Figure 1 Cartoon illustrating ambiguity in wording

The role of systems engineer in the SLC

"Modern information technology products, even the software-intensive ones, are complex enough to require application of techniques from both disciplines. Accordingly, it becomes important to understand the relationships between relevant standards for systems and software engineering" (Moore, 1998).

(ISO 12207) depicts the link between system and software as: *"This standard establishes a strong link between a system and its software. It is based upon the general principles of systems engineering. The basic components of systems engineering (e.g., analysis,*

design, fabrication, evaluation, testing, integration, manufacturing, and storage/distribution) form the foundation for software engineering in the standard. This standard provides the minimum system context for software. Software is treated as an integral part of the total system and performs certain functions in that system. This is implemented by extracting the software requirements from the system requirements and design, producing the software, and integrating it into the system".

While mentioning systems engineering, the standard is silent as to the role of systems engineers. A search through engineering textbooks and other sources for the role of systems engineers showed that systems engineering is defined in several ways. A representative sample of definitions was shown in Table 1. These different perceptions of systems engineering illustrate the point that systems engineers themselves had different perceptions as to their role in the SLC (Kasser, 1996). For example, the systems engineer pointed out that systems engineers might direct or perform:

- The high level design.
- Requirements and interface management.
- Activities that are not performed by other specialty disciplines.
- Inter-group coordination.
- The advocacy for the customer during the design and test phases of the task and whenever the customer is not present.

After viewing the list the software engineer pointed-out in disdain "with so many definitions no wonder you get us confused, as to your role".

Table 1 A selection of definitions of systems engineering as published in chronological order

The combination of advanced chemical engineering science with the tool of electronic computers and the viewpoint of considering the process as an entity (Williams, 1961).
Systems engineering considers the content of the reservoir of new knowledge, then plans and participates in the action of projects and whole programs of projects leading to applications. It considers the needs of its customers and determines how these can best be met in the light of all knowledge both old and new. Thus systems engineering operates in the space between research and business, and assumes the attitudes of both. For those projects which it finds most worthwhile for development, it formulates the operational, performance and economic objectives, and the broad technical plan to be followed (Hall, 1962) page 4).
The science of designing complex systems in their totality to ensure that the component sub-systems making up the system are designed, fitted together, checked and operated in the most efficient way (Jenkins, 1969).
Systems engineering covers the comprehensive aspects of engineering practice, and the application of the modern rational approach to the formulation and solution of technical problems (Au and Stelson, 1969) page 1)
The application of scientific and engineering efforts to transform an operational need into a description of system performance parameters and a system configuration through the use of an iterative process of definition, synthesis, analysis, design, test, and evaluation; integrate related technical parameters and Ensures compatibility of all physical, functional, and program interfaces in a manner that optimises the total system definition and design; integrate reliability, maintainability, safety, survivability, human engineering, and other such factors into the total engineering effort to meet cost, schedule, supportability, and technical performance objectives.(MIL-STD-499A, 1974).
The transforming of an operational need into a description of system performance parameters and a system configuration. (FM_770-78, 1979)
A hybrid methodology that combines policy analysis, design and management. It aims to ensure that a complex man-made system, selected from the range of options on offer, is the one most likely to satisfy the owner's objectives in the context of long-term future operational or market environments (M'Pherson, 1986) pages 330-331).
An iterative process of top-down synthesis, development, and operation of a real-world system that satisfies, in a near-optimal manner, the full range of requirements for the system (Eisner, 1988) page 17).
The management function which controls the total system development effort for the purpose of achieving an optimum balance of all system elements. It is a process which transforms an operational need into a description of system parameters and integrates those parameters to optimise the overall system effectiveness (DSMC, 1989; SEMG, 1990) pages 1-2).
A robust approach to the design and creation of systems to accomplish desired ends (Chamberlain and Shishko, 1991) page 23).
An interdisciplinary approach to evolve and verify an integrated and life cycle balanced set of system product and process solutions that satisfy customer needs. Systems engineering: a) encompasses the scientific and engineering efforts related to the development, manufacturing, verification, deployment, operations, support, and disposal of system products and processes, b) develops needed user training equipments, procedures, and data, c) establishes and maintains configuration management of the system, d) develops work breakdown structures and statements of work, and

e) provides information for management decision making (MIL-STD-499B, 1992).

A management technology (Sage, 1992) page 1).
The design, production, and maintenance of trustworthy systems within cost and time constraints (Sage, 1992) page 10).
The intellectual, academic and professional discipline the principal concern of which is the responsibility to ensure that all requirements for a bioware/hardware/software system are satisfied throughout the life of the system (Wymore, 1993) page 5)
Integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. systems engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs (Sage, 1995) .
A discipline created to compensate for the lack of strategic technical knowledge and experience by middle and project managers in organizations functioning according to Taylor's "Principles of Scientific Management" (Chapter 2).
Comprises systems analysis, systems integration and human factors including human-computer interaction.(Anderson and Dibb, 1996)
The activity of specifying, designing, implementing, validating, installing and maintaining systems as a whole (Sommerville, 1998).
A complex collection of interactive units and subsystems within a single product, jointly performing a wide range of independent functions to meet a specific operational mission or need (Shenhar and Bonen, 1997).
A set of activities which control the overall design, development, implementation and integration of a complex set of interacting components or systems to meet the needs of all the users (DERA, 1997).
An interdisciplinary approach and means to enable the realisation of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem. (INCOSE, 2000).
The design and analysis process which decomposes an application into software and hardware (Brodie, 2001) page 249)
The process that identifies the technical characteristics and operating rules of that system that best achieves the objectives in question (Westerman, 2001) page 6).
The art and science of creating systems (Hitchins, 2003)
Deals with the planning, development, and administration of complex systems, particularly of computing systems (Endres and Rombach, 2003) page 1).
Provides a framework, within which complex systems can be adequately defined, analysed, specified, manufactured, operated, and supported (Faulconbridge and Ryan, 2003).
Guides the engineering of complex systems (Kossiakoff and Sweet, 2003)

"In addition", he said, "in the 20th and 21st Century paradigms, which are based on building hardware, systems engineers convert customer needs to system requirements", and he then quoted from (Kasser, 1996).

Most of today's systems engineers really appear (work as) to be Requirements and Interface Engineers. They have the responsibility to validate the requirements since there's little point in building a system which conforms to requirements if the requirements are incorrect".

"In the 21st Century paradigm, the system engineer should lead the Integrated Process Team (IPT)". He added, to which she pointed out, "some modern software-intensive systems can be developed effectively without systems engineers as long as the software engineers perform the requirements elicitation function".

To which he replied "(Pfleeger, 1998) begins the process of describing a system by naming its parts and then identifying how the component parts are related to each other. The system is analysed in terms of objects and activities. There is a role for requirements engineering to capture the system level requirements, but the book is silent as to how the requirements are allocated between the software and hardware subsystems".

The use of concepts

As an example of the different interpretations of concepts this section addresses the differences in the following concepts:

- Inheritance
- Blueprints
- Models
- Objects and classes
- Architecture

- Adaption
- Use of viewpoints

Inheritance

Systems and hardware engineers use inheritance in the form of physical and domain knowledge as shown in the following examples:

- When they build a printed circuit board they inherit mechanical aspects (size and shape) from other printed circuit boards. The board may also inherit connectors and interface circuits from previous boards.
- Spacecraft inherit environmental properties i.e. thermal vacuum, thermal, and vibration.
- Aircraft inherit attributes to make them airworthy.
- Ships inherit attributes to stop them from sinking and protect them from the long term effects of sea water.

However they do not generally realize that they are employing the concept of inheritance, other than the obvious case of reuse, so they have no methodology for using the concept.
Software engineering uses inheritance as an integral part of object-oriented techniques. In modern software engineering you can inherit any classifier: interface, class, use case etc., thus creating a powerful way of expressing and implementing ideas for specialization and generalization, accompanied by rules of refinement.

Blueprints

"We have always used Blueprints in the form of engineering drawings" he said.
"We software engineer use blueprints as a metaphor claiming that software is always a blueprint even when implemented," she countered.

Models

"In my view a model is a representation of a product" he said. She replied "a model is a way of expressing knowledge in an abstract way, yet exact, without showing unnecessary details. We can use a model during analysis or design. We can use a model to generate implementation. Software engineers model as a way of communicating knowledge"

Objects and classes

In software engineering:

- An object is a discrete entity with a well-defined boundary and identity that encapsulates state and behaviour. An object is a role-centred entity with internal data and a set of operations provided for that data. An object is an instance of a class.
- A class is the type of an object. A class is a descriptor for a set of objects that share the same attributes, operations (methods), relationships and behaviour. Examples are a set of people, places, things or transactions that share common attributes and perform common functions (Dewitz, 1996). A class might capture real-world concept or design concept. A class can inherit another class's operation, relationship and behaviour either for expressing commonalties or as a way of making a more specific class. Objects can be instantiated from every class (not just from the more specialized one).

Objects and Classes originally were used in Object-Oriented Programming (OOP) to achieve encapsulation, reuse, inheritance and abstraction, later migrated to Object-Oriented Analysis (OOA) as a way of capturing the real world (the holistic approach).

In systems engineering an object is a subsystem. There is no concept of class. "It goes back to the fact that you are actually not inheriting" she pointed out.

Architecture

In software engineering an Architecture is
"*The organizational structure and associated behaviour of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems*" (UML, 1999).
or/and
"*The set of design decisions about any system (or smaller component) that keeps its implementers and maintainers from exercising needless creativity*" (D'Souza and Willis, 1998).

In systems engineering, the architecture of a system consists of the structure(s) of its parts (including design-time, test-time, and run-time hardware and software parts), the nature and relevant externally visible properties of those parts (modules with interfaces, hardware units, objects), and the relationships and constraints between them. There are a great many possibly interesting relationships.

"On the first reading they both look the same" she said, then added, "the software is adding a layer of abstraction that allows the developer to extend his problem dimensions".

"And later definitions include system evolution", he replied. "For example, the United States Department of Defense Architecture Framework (DODAF) provides the following definition of an Architecture – '*the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time*' (DoDAF, 2004)".

Adaptation

The concept of Adaptation was addressed above.

Use of viewpoints.

Systems engineers' views tend to be constrained in the physical realm. So they tend to mix the physical and abstract views (Lykins, Friedenthal and Meilich, 2000). This tends to result in a one-to-one mapping between the functional and the physical. Software engineers may pick any number of abstract views and try to separate concerns by splitting them to different yet related viewpoints.

Discussion

There are several reasons for the gulf separating software engineering from the hard engineering disciplines including:

- Systems engineers tend to think physically, and move rapidly to solutions. They do this using the reductionist approach of partitioning the big problem into a number of smaller problems on the assumption that if all the small problems are solved, then the big problem will also be solved. They also have a wide range of all sorts of components to assemble into their architectures (resistors, computers, tanks, aircraft, etc.)
- Hardware components generally evolved from one specific application to other applications. Systems and hardware engineers tend to focus on solving the problem, if they have to build a component, it tends to be a special purpose component optimised for that application. Hardware engineers constructed their own computer cards until vendor components become available. Similarly, companies developed proprietary network protocols until standards were adopted. Software engineers tend to think in abstractions and try to find an elegant solution to the problem (sometime staying abstract for a longer period than the system engineer feels comfortable with). In general, software does not have the benefit of thousands of years of component development. Thus when developing applications, software engineers may try to develop them in a manner that allows them be reused in the same or in other yet to be specified applications.
- A major barrier is the semantic barrier due to the different perceptions of the use of words. Both sides must be made aware that the situation is akin to Humpty Dumpty telling Alice that when he uses a word it means just what he chooses it to mean — neither more nor less (Carroll, 1872).

Consequently, in an exchange of information, each side should use active listening techniques to minimize loss of meaning across the gap in their common interface.

Bridging the communications gap between systems and software engineers

In recent years advances have been achieved in two areas that have the potential of dealing with the explored gap. These two areas are:

1. CMMI – the extension of CMM to address both software and systems engineering issues became wide spread
2. SysML – the adaptation of UML to address system views was defined

However while exploring the effect of these on the communication gap, the known symptoms still emerge:

1. Most practitioners and supporting personal of the CMMI come from the software arena causing repeated comments from the traditional systems engineering community – "it must fit software projects".
2. The additions made to the SysML are mainly of the physical diagrams – and the amazing requirements diagram that tries to model things even if it does not need modelling.
3. The use of a semi common language to describe system and software leads to expectations of direct decomposition which of course does not follow.

Therefore while improving the situation the gap still remains.

The key element in bridging the communications gap between systems and software engineers is to use active listening enhanced by "pattern matching" during discussions. For example, during a meeting discussing a scenario, design issue or requirement, they can often be seen to be similar to previous encounters. Thus when faced with a problem, the project team should first review this chapter which will sensitise them to the issues and the potential barriers to communications. Only then should the discussions take place. During these discussions different people may recognize different similarities to previous encounters. Let each take a turn in explaining what pattern they recognize and why. Thus for example

- Systems engineering may recognize a Type A scenario.
- Hardware engineering may recognize a Type B situation.
- Software engineering may see it as a Type C.
- Reliability engineering may see it as a cross between a Type B and a Type C.

Participants in the meeting should use active listening techniques enhanced by pattern matching to apply feed back to the communications process to maximize the probability of sharing the meaning. Active listening is a standard technique for applying the feedback principle to inter-personal communications to minimize errors in conveying the meaning from one person to another. Active listening first recognizes that during a conversation, most people do not listen to what the other person is saying. They are too busy planning what they will say when the other person pauses. Standard active listening comprises the following multi-step process:

When the other person speaks gives them your full attention and look them straight in the eyes. Then begin iteration loop.

1. Listen to everything the other person says and try to understand it fully.
2. Ask questions to clarify anything you don't understand and analyse the response.
3. Rephrase what you have heard in your own words and ask the speaker if they meant what you are about to say. Use words such as "if I understand you, then …….", or "Do you mean….." This is the principle of applying feedback.
4. If, after you have rephrased what has been said and the person says, "No that's not it!" or the equivalent, then go back to step 2. You may need to invoke the STALL technique at this time (see below).
5. When the speaker finally agrees with you, then you have (most probably) actually communicated and shared meaning.
6. In modifying active listening by the use of pattern matching, change Step 3 to incorporate the pattern by adding words such as "this reminds me of the [Type A Scenario]", and "isn't this similar to [Type B]" and explain why you find a similarity in the current situation. Use a metaphor appropriate to the other party such as sport.

During the conversation use the STALL approach to regulate matters[8]. STALL is an acronym for

- **S**tay calm
- **T**hink
- **A**sk questions and analyse
- **L**isten
- **L**isten

You have two ears and one mouth, use them in that ratio.

"It sounds a bit like marriage counselling" said the tired software engineer.

Summary

This chapter has provided some insight that the failure of systems and software engineers to communicate, and more importantly their failure to understand that they are not communicating, may be a hitherto undetected cause of the failures of today's complex projects. The chapter then explored some of the reasons for the communications failure and recommended enhancing active listening with pattern matching as approach to bridging the communications gap. Active listening is a well-established technique for bridging communications problems and sharing meaning.

Conclusion

Systems and software engineers think differently, come from different backgrounds and use different vocabularies, hence stretching the point, it can be stated that systems engineers are from Mars and software engineers are from Venus.

[8] Stalling is a good way to initially deal with most problem situations.

References

Anderson, K. and Dibb, P., "Strategic Guidelines for enabling research and development to support Australian Defence, paragraph 121," 1996.

Arthur, L. J., *Improving Software Quality An Insider's Guide to TQM*, John Wiley & Sons, Inc., 1993.

Au, T. and Stelson, T. E., *Introduction to Systems Engineering Deterministic Models*, Addison-Wesley Publishing Company, 1969.

Bass, L., Clements, P. and Kazman, R., *Software Architecture in Practice*, Addison Wesley, 1998.

Berg, J. M., Levia, O. and Rouillard, J., *Object-Oriented Modelling*, Kluwer Academic Publishers, 1996.

Blanchard, B., *System Engineering Management*, John Wiley & Sons, Inc., 1998.

Boehm, B., "*Unifying SWE and SE*", Computer (2000), Number March.

Britton, C. and Doake, J., *Software System Development*, McGraw-Hill Education (UK) Ltd., 2003.

Brodie, E. J., *Software Engineering An Object-Oriented Perspective*, John Wiley & Sons, Inc., 2001.

Budd, T., *An Introduction to Object-Oriented Programming*, Addison-Wesley, 1996.

Buede, D. M., *The Engineering Design of Systems*, John Wiley & Sons, Inc., 2000.

Carroll, L., *Through the Looking Glass*, 1872.

Chamberlain, R. G. and Shishko, R., "*Fundamentals of Systems Engineering at NASA*", INCOSE/ASEM Proceedings1991.

Cox, C., *Welcoming Immigrants to a Diverse America: English as our Common Language of Mutual Understanding*, 1996, http://policy.house.gov/documents/statements/english.htm, last accessed 10 August 2000

DeMarco, T., *Why Does Software Cost So Much? And Other Puzzles of the Information Age*, Dorset House Publishing, 1995.

DeMarco, T. and Lister, T., *Peopleware Productive Projects and Teams*, Dorset House Publishing, 1999.

DERA, "DERA Systems Engineering Practices Reference Model," 1997.

Dewitz, S. D., *Systems Analysis and Design and the Transition to Objects*, Irwin/McGraw Hill, 1996.

DoDAF, *DoD Architecture Framework Version 1.0, 9 February 2004*, 2004.

Donaldson, S. E. and Siegel, S. G., *Cultivating Successful Software Development A Practitioner's View*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1997.

DSMC, *Systems Engineering Management Guide*, Defence Systems Management College, 1989.

D'Souza, D. F. and Willis, A. C., *Objects, Components, and Frameworks With UML: The Catalysis Approach*, Addison-Wesley, 1998.

Eisner, H., *Computer Aided Systems Engineering*, Prentice Hall, 1988.

Eisner, H., *Essentials of Project and Systems Engineering Management*, John Wiley & Sons, Inc., 1997.

Endres, A. and Rombach, D., *A Handbook of Software and Systems Engineering*, Pearson Education Ltd., 2003.

Faulconbridge, R. I. and Ryan, M. J., "*A Framework for Systems Engineering Education*", Systems Engineering Test and Evaluation (SETE) Conference, Adelaide, Australia, 1999.

Faulconbridge, R. I. and Ryan, M. J., *Managing Complex Technical Projects: A Systems Engineering Approach*, Artech House, Boston, 2003.

FM_770-78, "Field Manual: System Engineering," Headquarters, Department of the Army, 1979.

Fowler, M., *Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1997.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J., *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

Hall, A. D., *A Methodology for Systems Engineering*, D. Van Nostrand Company Inc., Princeton, NJ, 1962.

Hitchins, D. K., *Putting Systems to Work*, John Wiley & Sons, 1993.

Hitchins, D. K., *Advanced Systems Thinking, Engineering and Management*, Artech House, 2003.

Humphrey, W., *A Discipline for Software Engineering*, Addison-Wesley, Reading, MA, 1995.

INCOSE, *What is systems engineering?*, 2000, , (http://www.incose.org/whatis.html, last accessed 9th November 2000

ISO 12207, "ISO 12207."

Jacobson, I., Christerson, M., Jonsson, P. and _vergarrd, G., *Object-Oriented Software Engieering A Use Case Driven Approach*, Addison Wesley, 1993.

Jenkins, G. M., "The Systems Approach," *Systems Behaviour*, J. Beishon and G. Peters (Editors), Harper and Row, London, 1969, p. 82.

Kasser, J. E., *Applying Total Quality Management to Systems Engineering*, Artech House, Boston, 1995.

Kasser, J. E., "*Systems Engineering: Myth or Reality*", The 6th International Symposium of the INCOSE, Boston, MA., 1996.

Kasser, J. E., "*The Certified Systems Engineer - It's About Time!"* The Systems Engineering, Test and Evaluation (SETE) Conference, Brisbane, Australia, 2000.

Kendall, K. E. and Kendall, J. E., *Systems Analysis and Design*, Prentice Hall, Upper Saddle River, NJ, 1997.

Kossiakoff, A. and Sweet, W. N., *Systems Engineering: Principles and practice*, John Wiley & Sons Inc., 2003.

Lykins, H., Friedenthal, S. and Meilich, A., "*Adapting UML for an Object Oriented Systems Ewngineering Method (OOSEM)"*, The 10th Annual Symposium of the INCOSE, Minneapolis, MN., 2000.

Marcotty, M., *Software Implementation*, Prentice Hall, 1991.

Martin, J. N., *Systems Engineering Guidebook: A process for developing systems and products*, CRC Press, 1997.

Metzger, P. W. and Boddie, J., *Managing a Programming Project Processes and People*, Prentice Hall PTR, Upper Saddle River, NJ, 1996.

MIL-STD-499A, *Mil-STD-499A Systems Engineering*, United States Depeartent of Defense, 1974.

MIL-STD-499B, *Mil-STD-499B Systems Engineering*, United States Depeartent of Defense, 1992.

MIL-STD_2167A, *Defense System Software*, 1998.

Moore, J. W., *Software Engineering Standards A User's Road Map*, IEEE Computer Society, Los Alamitos, CA., 1998.

M'Pherson, P., "*Systems engineering: A proposed definition"*, IEE Proce 133 (1986), Number September.

Perry, W. E., *Effective Methods for Software Testing*, John Wiley & Sons, 2000.

Peters, J. F. and Pedrycz, W., *Software Engineering An Engineering Approach*, John Wiley & Sons, Inc., New York, 2000.

Pfleeger, S. L., *Software Engineering Theory and Practice*, Prentice Hall, New Jersey, 1998.

Phillips, D., *The Software Project Manager's Handbook Principles that Work at Work*, IEEE Computer Society, Los Alamitos, CA., 1998.

Pressman, R. S., *A Manager's Guide to Software Engineering*, McGraw-Hill, 1993.

Pressman, R. S., *Software Engineering A Practitioner's Approach*, McGraw-Hill, 2005.

Prieto-Díaz, R. (Editor), *Classification of Reusable Models published in Software Reusability*, ACM Press, 1987.

Rechtin, E., *Systems Architecting, Creating & Building Complex Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

Sage, A., *Systems Management for Information Technology and Software Engineering*, Wiley, 1995.

Sage, A. P., *Systems Engineering*, John Wiley & Sons, Ltd., 1992.

SEMG, D., *DSMC Systems Engineering Management Guide*, 1990.

Shaw, G. B., "*England and America: Contrasts," a conversation between Bernard Shaw and Archibald Henderson"*, Reader's Digest (1925), Number.

Shenhar, A. J. and Bonen, Z., "*The New Taxonomy of Systems: Toward an Adaptive Systems Engineering Framework"*, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 27 (1997), Number 2, 137 - 145 137 - 145.

Shlaer, S. and Mellor, S. J., *Object Oriented Systems Analysis*, Yourdon Press, 1988.

Shumate, K. and Keller, M., *Software Specification and Design A Disciplined Approach for Real-Time Systems*, John Wiley & Sons Inc., 1992.

Sommerville, I., *Software Engineering*, Addison-Wesley, 1998.

Szyperski, C., *Component Software*, Addison Wesley, 1997.

Thome, B. (Editor), *Systems Engineering Principles and Practice of Computer-based Systems Engineering*, John Wiley & Sons, 1993.

UML, "OMG Unified Modeling Language Specification Version 1.3," OMG, 1999.

UML, "Unified Modeling Language: Superstructure," OMG, 2005.

Van Vliet, H., *Software Engineering Principles and Practice*, John Wiley & Sons Ltd., 2000.

Weinberg, G. M., *The Psychology of Computer Programming, Silver Anniversary Edition*, Dorset House Publishing, 1998.

Westerman, H. R., *Systems Engineering Principles and Practice*, Artech House, Boston, 2001.

Williams, T. J., *Systems Engineering for the Process Industries*, McGraw-Hill, 1961.

Wymore, A. W., *Model-Based Systems Engineering*, CRC Press, Boca Raton, 1993.

Yourdon, E., *Decline & Fall of the American Programmer*, Yourdon Press, 1993.

Yourdon, E. and Argila, C., *Object Oriented Analysis and Design*, Yourdon Press, 1993.

יום עיון על מחקר בהנדסת מערכות בטכניון

יום עיון על מחקר בהנדסת מערכות נערך בטכניון ביום 6.12.2006. האירוע נתמך במשותף ע" י מרכז גורדון להנדסת מערכות בטכניון, אשר נתן את התמיכה כספית, יחד עם הסניף הישראלי של INCOSE. היוזמה לקיום יום העיון הייתה של ד"ר אביגדור זוננשיין, ומארגני הכנס היו פרופ' מנחם וייס ופרופ' יוסי בן- אשר.
בפתיחה השתתף המשנה לנשיא הטכניון פרופ' אביב רוזן, שהיה היזם העיקרי של הקמת המרכז למחקר מערכות בטכניון. השתתפו כ 90 אישה ואיש. ביום העיון הוצגו מחקרים מהאוניברסיטאות ומהתעשייה, אשר נתמכו ע"י קרנות שונות, וביניהן קרן גורדון והקרן המשותפת של וות" ת עם IAEC. רמת המצגות הייתה גבוהה, נשאלו שאלות רבות וקויים רב שיח בנושאים השונים. רשימת ההרצאות מוצגת להלן. יום העיון התחלק לשלושה מושבים בתחומים הבאים:
שלבי הגדרת המערכת, התנהגות האדם בצוותי תכן מערכות ובכלים ותוכנות. בנפרד התקיים פנל על השאלה אם הנדסת מערכות היא כמותית מספיק? הפנל זכה לתגובות ערות מאד מצד המשתתפים. גם אווירה, המקום, הכיבוד ורוח יום העיון היו נעימים מאד וזכו להערכה רבה מכל המשתתפים. מסתבר שרבים בארץ מתעניינים בתחום המקצועי של הנדסת מערכות. יום העיון זכה להצלחה רבה מאד, בהרשמה וגם בהשתתפות המלאה. עקב מגבלות האולם היה על המארגנים לדחות כ 20 אנשים נוספים שרצו להשתתף. מעל 70 משתתפים מתוך כ 90 ס" ה, נשארו עד לסוף הכנס, וגם זה דבר חריג. ארבע מתוך ההרצאות שהוצגו בכנס יוצגו גם במושב מחקר מערכות בכנס הישראלי של INCOSE IL. לאור ההצלחה נשקלת האפשרות לקיים כנס מסוג זה אחת לשנה.

רשם: פרופ' מנחם וייס

Hari A., Kasser J.E. and Weiss M.P.: Lessons Learnt Led to a Modified QFD Approach for Translating Customer Needs into a Requirements Specification

Dori Dov: Aligning System Requirements and Implementation by Bridging Information Gaps Between System Development Stages

Shtub Avraham & Golany Boaz: A Model for Justifying Investment in Quality Function Deployment

Erez Miriam: Innovation in Design Teams **Arbel Iris and Erez Miria**m: Training new product Design Teams to improve Team Innovation

Shpitalni M., Molcho G. and Stiassnie E.: The Holistic Product and Service Life Cycle Approach and its influence on the Modern Manufacturing

Paradigm: Management, Manufacturing and End of Life Systems

פנל ורב שיח על ": **האם הנדסת מערכות היא כמותית מספיק"?**

הפנליסטים : **ד"ר אביגדור זוננשיין, ד"ר חיים מזרחי, ד"ר מוטי פרנק**

Bustnay Tamir: How Many Systems Are There? – Using the N2 Method for Systems Partitioning

Pariente Meidad & Tamir Raz: On Artificial Intelligence Methods for System Functional Testing Optimization

Sinreich David and Marmor Yariv: Efficient Design of Simulation Models Based on Model Reuse And Generic Processes

האיגוד הישראלי להנדסת מערכות INCOSE_IL
טופס הצטרפות 2007

1. רקע

INCOSE_IL הינו הסניף הישראלי של הארגון הבין לאומי INCOSE. ארגון אשר מטרתו לקדם את הידע, ההבנה, התאום והיישום של הנדסת מערכות בעולם. מטרות INCOSE_IL (כמו בארגון הבינלאומי) הן לקדם את נושאי הנדסת המערכות בישראל, בתעשייה, באקדמיה ובמוסדות ממשלתיים בארץ ע"י:

- קידום הדרכות וחינוך מהנדסי מערכת במפעלים, באקדמיה ובהשתלמויות כלליות.
- קידום שיתופי פעולה ושיתוף במידע וידע בין מהנדסי מערכת באמצעות מפגשים, קבוצות עבודה וכנסים ייעודיים.
- הפצת מידע לידע מקצועי בנושא הנדסת מערכות באמצעות אתר האיגוד.
- עידוד מחקר אקדמי בנושאי הנדסת מערכות.
- ייזום וייצור שיתופי פעולה עם איגודים מקצועיים רלוונטיים כמו PMI, ISQ.

ארגון INCOSE הוא ארגון דינמי הכולל כיום, כ - 17 שנה מיום הקמתו, למעלה מ - 6500 חברים מכל העולם ו - 53 סניפים מ- 36 מדינות. הסניף הישראלי, INCOSE_IL, הוקם לפני ארבע שנים בשיתוף עם אילטם וכולל חברים מחברות שונות אזרחיות וביטחוניות. בפעילויות הסניף הנערכות בארץ משתתפים למעלה מ – 600 מהנדסי מערכות.
החברים בארגון INCOSE העולמי מקבלים את הרבעון Systems Engineering של INCOSE, מקושרים לקהיליית מהנדסי המערכות העולמית דרך אתר INCOSE לרבות גישה לאלפי מסמכים ומאמרים הנמצאים באתר. לחברי INCOSE הנחה בכנסים הבינלאומיים של INCOSE.

2. בחירות ליו"ר INCOSE_IL:

אחת לשנתיים עורך הסניף הישראלי בחירות ליו"ר הסניף, זמן כהונת היו"ר הנבחר הינו שנתיים.

3. INCOSE_IL

3.1 חברים

כיום חברים ב-INCOSE _IL כ- 100 איש, בנוסף משתתפים בפעילויות INCOSE_IL למעלה מ – 600 מהנדסי מערכות החברים באילטם – איגוד משתמשים. ציבור זה מהווה חלק בקהילת המתעניינים בהנדסת מערכות בישראל המונה יותר מ- 1000 איש. קבוצה זו מהווה את הגרעין המשתתף בכנסים הלאומיים שארגן INCOSE_IL ב- 6 השנים האחרונות (ארבעה במספר) ואת המשתתפים בעשרות ימי עיון ומפגשים טכניים הנערכים על ידי INCOSE_IL.
רוב החברים שייכים לארגונים החברים באילטם – איגוד משתמשים, וחלקם הם מהנדסי מערכות בחברות שאינן חברות באילטם.

3.2 הנהלה

הנהלת INCOSE_IL מובילה את הפעילויות השונות בהנדסת מערכות בארץ וחבריה פועלים בהתנדבות. נשיא INCOSE_IL נבחר בבחירות, אחת לשנתיים, ע"י חברי INCOSE_IL.
צוות ההנהלה נבחר ע"י היו"ר והרכבו מאושר במפגשי צוות ההנהלה.

3.3 צוות ההנהלה הנוכחי (מ- 9/05)

- ד"ר אביגדור זוננשיין (רפאל) - יו"ר מכהן
- עוזי אוריון (אל-אופ) - יו"ר הבא
- ד"ר מיכאל וינוקור (תע"א) - יו"ר לשעבר
- מוביל חברים וחברות - ד"ר משה ויילר (צה"ל)
- מוביל תקשורת - סרגי טוזיק (צה"ל)
- מובילה שותפות עם חברות - רויטל גולדברג (אלתא)
- מוביל קשרים בינלאומיים - יעקב כגן (מל'מ)
- רכז מפגשים טכניים - עמיר רוה (אומניסס)
- גזבר - משה סלם (אילטם)
- עורך עיתון - ד"ר עמיהוד הרי
- מזכירות - רינת דורי פלד, חגית סמוחה
- חברים נספחים: ד"ר חיים מזרחי, שרון שושני, פרופ' יוסי בן-אשר.

4. שירותים

האיגוד הישראלי להנדסת מערכות INCOSE_IL מציע לחבריו ולקהילת מהנדסי המערכות ולחברות המיישמות הנדסת מערכות – מגוון פעילויות מקצועיות:

פעילות	מועד	הערות
כנס הנדסת מערכות לאומי	אחת לשנתיים הכנס הבא מרץ 09	ב-6 השנים האחרונות קיימנו 4 כנסים לאומיים
מפגשים טכניים	אחת לחודשיים	קיימנו עשרות מפגשים טכניים בנושאים מגוונים
ביקורים במפעלים/חברות	אחת לרבעון	
סמינרים מקצועיים	אחת לרבעון	הסמינרים ניתנים ע"י מומחים מחו"ל
קבוצות עבודה	פגישות שוטפות	קבוצות עבודה: מתודולוגיות וכלים-בהקמה ניהול אינטגרציה-פועלת CMMI- פועלת
מושבים מקצועיים בכנסים אחרים	במועדי הכנסים – אחת לשנה	PMI ISQ

INCOSE_IL מפעיל אתר מידע מקצועי בנושא הנדסת מערכות, האתר מופעל במסגרת אתר אילטם www.iltam.org וכולל:

- נתונים על ארגון INCOSE_IL
- נתונים על ארגון INCOSE העולמי
- מידע על פעילויות INCOSE_IL
- מאמרים ומצגות מקצועיות בנושאי הנדסת מערכות
- פורום מהנדסי מערכת לתקשורת בין מהנדסי מערכת בנושאים מקצועיים

אנשי INCOSE_IL פעילים גם בכנסים הבינלאומיים בנושא הנדסת מערכות המתקיימים בעולם, ובמיוחד בכנסים:

- כנס INCOSE המתקיים אחת לשנה בארה"ב או מחוצה לה.
- כנס EUSEC המתקיים אחת לשנתיים באירופה.

הצטרפות לאיגוד INCOSE העולמי

ניתן להצטרף לאיגוד INCOSE העולמי באמצעות הטופס שלהלן. התשלום מבוצע בכרטיס אשראי בדולרים. את קובץ ההצטרפות ניתן להוריד מהכתובת: https://secure.sbims.com/incose/incosememapp.asp

כמו כן ניתן להצטרף לאיגוד INCOSE העולמי באמצעות אילטם. התשלום להצטרפות באמצעות אילטם מבוצע בשקלים בתוספת מע"מ. להצטרפות באמצעות אילטם, שלח אלינו את טופס ההרשמה שלהלן ואנו נעביר אליך את המחיר בשקלים.

פרטים נוספים על החברות באתר INCOSE העולמי: www.incose.org

טופס בקשה לרישום לאיגוד INCOSE העולמי

לכבוד אילטם: **Email: incose_il@iltam.org** **פקס: (03) 5100622** **טלפון: (03) 5118112**

שם: _____ **חברה:** _____ **תפקיד:** _____

טלפון: _____ **פקס:** _____ **Email** _____

כתובת למשלוח מכתבים: _____

המרד 29 בית התעשיינים ת.ד. 50232 תל אביב 61500 טל': 03 - 5118112, פקס: 03 - 5100622
Email: iltam@iltam.org **www.iltam.org**

*** להצטרפות ישירה דרך INCOSE העולמי אנא מלא את הטופס המצורף.**

קול המערכות

The Voice of the Systems

INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING
2150 N 107th St #205 Seattle,WA 98133-9009 (USA)
Phone: (206) 361-6607 or (800) 366-1164
Fax: (206) 367-8777 Email: info@incose.org
World Wide Web URL=http://www.incose.org/

Individual Membership Application FOR NEW MEMBERSHIPS

1. Membership Type *Please circle the amount remitted* 1/07

Reduced membership entitles the member to soft copy only for The SE Journal and *INSIGHT*.
Please see the INCOSE web site for details and eligibility

based on country of residence, for reduced membership rates.
Soft copies of INCOSE publications are available to all members via the INCOSE web site.

Month of Joining *....	Jun Jul Aug (06/07)	Sep Oct Nov (06/07)	Dec Jan Feb (06/07)	Mar Apr May (07/08)
Regular Membership	$105	$80	$55	$130
Developing Country	$53	$40	$28	$65
Student Membership **	$20	$15	$10	$25

*** Membership year is from June 1 to May 31.**
**** Student members must be enrolled at least ¾ time in engineering or related fields and proof of class registration must be submitted with application. Student members will not received hardcopies of publications.**

2. Personal Information ***Please write address exactly as it should appear on a mailing label.***

Name FIRST M.I. LAST Nickname Check One ☐ Mr. ☐ Ms. ☐ Dr.

Mailing Address (NOTE - We prefer to mail to your home address.) Check One ☐ Home ☐ Office

Zone or City Code City State / Prov Zip/Postal Country

Office Phone Office Fax

Email Home Phone (Optional)

3. Professional Information

Company / Agency / Institution Position / Title

4. INCOSE Chapter Affiliation *Chapter list is available on our website*

☐ _____

☐ Please assign me to the closest chapter.

☐ I wish to be a Member-at-large.

5. Today's Date _____

☐ Please do not publish my information in the INCOSE Membership Directory.

☐ I do not wish to receive emails from INCOSE.

6. Amount Enclosed $ _____ **(U.S. Dollars Only)**

Check One: ____ Check from U.S. Bank (payable to INCOSE) ____ Money Order ____ Charge Card

Amex, MasterCard or Visa Card Only

Card Number _____ Expiration Date _____

Name as it appears on card_____ Signature_____