הנדסת תוכנה מבוססת שימוש חוזר היא סטרטגית הנדסת תוכנה שבה תהליך הפיתוח מיועד חוזר תוכנה קיימת. יחידות התוכנה מועברות לשימוש חוזרים עשויות להיות בגדלים שונים בתכלית. לדוגמה:

1. מערכת יישום חוזר כולה מערכת הבקשה ניתן לעשות בה שימוש חוזר על ידי שילוב זה מבלי לשנות לתוך מערכות אחרות
2. רכיבים חוזרים רכיב של יישום, החל בגודל תת לאובייקטים יחידים, ניתן לעשות בה שימוש חוזר.
3. רכיבי התוכנה אובייקט ותפקוד חוזר המיישמים פונקציה אחת, כגון פונקציה מתמטית, או מחלקת אובייקט ניתן לעשות בה שימוש חוזר.

**שימוש חוזר יתרונות:**

* *לאמינות משופרת* - השימוש חוזרת בתוכנה, אשר נוסתה ונבדקה במערכות עובדות, צריכה להיות יותר אמינה מאשר תוכנה חדשה.
* *סיכון בתהליך מופחת* - עלות התוכנה קיימת כבר ידועה, ואילו עלויות פיתוח הן תמיד עניין של

שיקול דעת. זהו גורם חשוב לניהול פרויקטים משום שהיא מקטינה את מרחב הטעות בהערכת עלות הפרויקט.

* *שימוש יעיל של מומחים* - במקום לעשות את אותה העבודה שוב ושוב, מומחי יישום יכולים לפתח תוכנה לשימוש חוזרת שעוטפת את הידע שלהם.
* *תאימות לתקנים* - אמות מידה מסוימות, כגון תקני ממשק משתמש, יכול להיות מיושם כמערכת של רכיבים הניתנים לשימוש חוזר. לדוגמה, אם התפריטים ממשק משתמש מיושמים באמצעות רכיבים הניתנים לשימוש חוזר, כל היישומים להציג את אותו פורמטים התפריט למשתמשים. השימוש בממשקי משתמש רגילים משפר מהימן כי משתמשים עושים טעויות פחות כשהוצגו ממשק מוכר.
* *פיתוח מואץ* - הבאת מערכת לשווק מוקדם ככל האפשר הוא לעתים קרובות יותר חשוב מאשר את עלויות הפיתוח הכולל. תוכנת שימוש חוזר יכולה להאיץ ייצור מערכת כי זמן הפיתוח ואימות הן עשוי להיות מופחתת.

**שימוש חוזר חסרונות:**

* עלויות תחזוקה מוגברות - אם את קוד המקור של מערכת תוכנה לשימוש חוזרת או רכיב אינו זמין, אז עלויות תחזוקה עשויות להיות גבוהות יותר משום שיסודות השימוש החוזרים של המערכת עשויים להיות תואמים יותר ויותר עם שינויים במערכת.
* חוסר תמיכת כלי - כמה כלי תוכנה אינם תומכים פיתוח עם שימוש חוזר. זה עלול להיות קשה או בלתי אפשרי לשלב כלים אלה עם מערכת ספריית רכיב.
* יצירה, שמירה, ושימוש ספריית רכיב - אכלוס ספריית רכיב לשימוש חוזרת והבטיח מפתחי התוכנה יכולה להשתמש בספרייה יכולה להיות יקרה. צריכים להיות מותאמים תהליכי פיתוח על מנת להבטיח כי הספרייה משמשת.
* מציאת, הבנה, והתאמת רכיבים הניתנים לשימוש חוזר - רכיבי תוכנה צריך להתגלות ספרייה, הבין, לפעמים, מותאם לעבודה בסביבה חדשה. מהנדסים חייבים להיות בטוחים למדי למצוא מרכיב בספרייה לפני שהם כוללים

גורמי מפתח שאתם צריכים לדעת לפני שימוש חוזר תכנון התחלה:

לוח הזמנים של הפיתוח של התוכנה אם התוכנה יש לפתח במהירות, אתה צריך לנסות לעשות שימוש חוזר של מערכות מוכנה ולא רכיבים בודדים.  
  
אורך חיי התוכנה ציפה אם אתם מפתחים מערכת ארוכת חיים, אתה צריך להתמקד על התחזוקה של המערכת. אתה לא צריך רק לחשוב על היתרונות המיידיים של שימוש חוזר, אלא גם על ההשלכות לטווח ארוכות.  
  
הקריטי של התוכנה ודרישותיו שאינו פונקציונליות עבור מערכת קריטית שחייבת להיות מאושרים על ידי רגולטור חיצוני  
  
הפלטפורמה שעליה המערכת להפעיל מודלים רכיבים מסוימים, כגון .NET, הן ספציפיות פלטפורמות מיקרוסופט.

**Application frameworks**

- Application framework אוסף של כיתות מופשטות וקונקרטיות מותאמות והאריכו ליצור מערכות יישום. מסגרות לספק תמיכה בתכונות גנרי צפויים לשמש את כל היישומים מסוג דומה. מסגרות לתמוך חוזר עיצוב בכך שהם מספקים ארכיטקטורת שלד עבור היישום, כמו גם את השימוש החוזר של שיעורים ספציפיים במערכת. הארכיטקטורה מוגדרת על ידי כיתות אובייקט ואת האינטראקציות ביניהם. השיעורים בהם שימוש חוזר ישירות והוא ניתן להארכה באמצעות תכונות כגון ירושה.

מסגרות מיושמות כאוסף של שיעורי אובייקט קונקרטיים ומופשט ב שפת תכנות מונחה עצמים. לכן, מסגרות מותאמות לשפות ספציפיות. ישנן מסגרות זמינה בכל שפות תכנות הנפוצות מונחה עצמים למשל, Java, C #, C++, כמו גם בשפות דינמיות כגון Ruby Python.

**שלושה סוגי מסגרות:**

מסגרות תשתית מערכת  
מסגרות שילוב Middleware  
מסגרות יישומים ארגוניים

מסגרות יישומי אינטרנט (WAFs) הן סוג אחרון יותר חשוב מאוד של מסגרת. WAFs התומכים בבניית אתרים דינאמיים זמינים כעת נרחב. מסגרות יישומי אינטרנט בדרך כלל לשלב מסגרות מיוחדות אחד או יותר שתומכות בתכונות יישום ספציפיות. למרות שכל מסגרת כוללת פונקציונליות שונה במקצת, רוב מסגרות יישום האינטרנט תומכות בתכונות הבאות:

האבטחה WAFS עשוי לכלול כיתות לעזור ליישם אימות משתמש (התחברות) ובקרת גישה על מנת

* להבטיח שמשתמשים יוכלו לגשת רק מותר פונקציונליים במערכת.

חוגים דינמי דפי אינטרנט ניתנים כדי לעזור לך להגדיר תבניות דף אינטרנט כדי לאכלס באופן דינמי אלה עם נתונים ספציפיים מבסיס הנתונים של המערכת.

* מסגרות מסד תמיכה לא כוללות בדרך כלל מסד נתונים אלא להניח בסיס נתונים נפרדים, כגון MySQL, ישמשו. המסגרת עשויה לספק כיתות המספקות ממשק מופשט אל מסדי נתונים שונים.
* חוגים וניהול מושב ליצור ולנהל פגישות (מספר אינטראקציות עם המערכת על ידי משתמש) הם בדרך כלל חלק WAFs.
* מסגרות אינטרנט רוב האינטראקציה משתמש כעת לספק תמיכה ב- AJAX, המאפשר דפי אינטרנט

אינטראקטיביים יותר להיווצר.

**קווי מוצרי תוכנה**

אחת הגישות היעילות ביותר לעשות שימוש חוזר הוא ליצור קווי מוצר תוכנה או משפחות יישום. קו מוצר תוכנה הוא קבוצה של יישומים עם ארכיטקטורה משותפת ורכיבים משותפים, עם כל יישום מתמחה לשקף דרישות שונות. מערכת הליבה נועדה להיות מוגדר ומותאמת כדי להתאים לצרכים של לקוחות שונים של המערכת. זו עשויה להיות כרוכה בהגדרת התצורה של רכיבים מסוימים, יישום מרכיבים נוספים, ושינוי חלק מהרכיבים לשקף דרישות חדשות. קווי מוצרי תוכנה בוקעים בדרך כלל מן היישומים קיימים.

מסגרות Application וקווי מוצר תוכנה יש כמובן הרבה מן המשותף. שניהם תומכים אדריכלות מרכיבים משותפים, ודורשים פיתוח חדש כדי ליצור גרסה ספציפית של מערכת. ההבדלים העיקריים בין גישות אלה הן כדלקמן:

1. . מסגרות יישום להסתמך על תכונות מונחות עצמים כמו ירושה פולימורפיזם ליישם רחבות במסגרת. באופן כללי, את קוד המסגרת אינו שונה ואת השינויים האפשריים מוגבלים מה מותר לפי המסגרת. קווי מוצרי תוכנה הם לא נוצרו בהכרח בגישה מונחה עצמים. רכיבי יישום מוחלפים, נמחקו, או לשכתב. אין גבולות, באופן עקרוני לפחות, לשינויים שניתן לעשות.
2. מסגרות Application מתמקדות בעיקר במתן טכני ולא תמיכת תחום ספציפי. לדוגמא, יש מסגרות יישום ליצור יישומים מבוססי אינטרנט. קו מוצר תוכנה בדרך כלל מטביע מידע תחום פלטפורמה מפורט. לדוגמא, יכול להיות שיש קו מוצר תוכנה עוסק יישומים מבוססי אינטרנט לניהול שיא בריאות.
3. קווי מוצרי תוכנה הם בדרך כלל לשלוט ביישומים עבור ציוד. לדוגמה, ייתכן שיש קו מוצר תוכנה עבור משפחה של מדפסות. משמעות הדבר היא כי קו המוצרים יש לספק תמיכה עבור התממשקות חומרה. מסגרות Application הם בדרך כלל בעלי אוריינטציה תוכנה והם כמעט ולא מספקים תמיכה עבור התממשקות החומרה.
4. קווי מוצר תוכנה מורכב משפחה של יישומים קשורים, שבבעלות אותו ארגון. בעת יצירת יישום חדש, נקודת המוצא שלך היא לעתים קרובות את החבר הקרוב של מש' היישום, לא יישום הליבה הגנרית

**חוזר מוצר COTS**

מוצר מוכן (COTS) היא מערכת תוכנה שיכולה להיות מותאמת לדרישות של לקוחות שונים מבלי לשנות את קוד המקור של המערכת. כמעט כל תוכנת שולחן העבודה ועוד מגוון רחב של מוצרי שרת הן תוכנות COTS. בגלל תוכנה זו מיועדת לשימוש כללי, זה בדרך כלל כולל תכונות ופונקציות רבות. לכן יש פוטנציאל לשימוש חוזר בסביבות שונות וכחלק יישומים שונים.

**יתרונות COTS**

1. כמו סוגים אחרים של שימוש חוזר, פריסה מהירה יותר של מערכת אמינה עשויים להיות  
   אפשרי.
2. אפשר לראות מה פונקציונלי מסופקת על ידי היישומים ולכן זה קל לשפוט אם הם צפויים להיות מתאימים. חברות אחרות כבר רשאי להשתמש ביישומים כה ניסיון של המערכות זמין.
3. כמה סיכוני פיתוח נמנעים באמצעות תוכנה קיימת. עם זאת, גישה זו יש סיכונים משלה, כפי שאדון בהמשך.
4. עסקים יכולים להתמקד בפעילות הליבה שלהם מבלי להקדיש הרבה משאבים לפיתוח מערכות הIT-
5. כמו פלטפורמות הפעלה להתפתח, עדכונים טכנולוגיים ניתן לפשט כמו אלה הנם באחריות ספק מוצר COTS ולא ללקוח.

**חסרונות COTS**

דרישות בדרך כלל צריכות להיות מותאמות על מנת לשקף את הפונקציונליות ואת אופן הפעולה של מוצר COTS. זה יכול להוביל לשינויים משבשים את תהליכים עסקיים קיימים.

המוצר COTS עשויה להתבסס על הנחות שאינן כמעט בלתי אפשרי לשנות. הלקוח חייב אפוא להתאים את העסק שלהם כדי לשקף הנחות אלה.

בחירת מערכת COTS המתאימה מיזם יכולה להיות תהליך קשה, במיוחד כמוצרי COTS רב אינם מתועדות היטב. עושה את הבחירה הלא נכונה עלולה להיות הרת אסון כפי שהוא עשוי להיות בלתי אפשרי לעשות את העבודה במערכת חדשות לפי הצורך.

יתכנו חוסר מומחיות מקומית לתמוך בפיתוח מערכות. כתוצאה מכך, הלקוח צריך לסמוך על הספק ויועצים חיצוניים עבור ייעוץ ופיתוח. עצה זו עלולה להיות מוטה, מכוון מכירת מוצרים ושירותים, ולא לענות על הצרכימים האמיתיים של הלקוח.

ספק מוצר COTS שולט תמיכה במערכת ואבולוציה. הם עשויים פושטות רגל, להיות השתלטו, או עשוי לבצע שינויים שגורמים קשיים על לקוחות

ישנם שני סוגים של שימוש חוזר מוצר COTS, כלומר מערכות COTS-פתרון ומערכות COTS משולבת. מערכות COTS-פתרון המורכב של יישום כללי  
מרוכל יחיד, המוגדר לדרישות הלקוח. COTS-מערכות משולבות לערב שילוב של שתיים או יותר מערכות COTS אולי מיצרנים שונים כדי ליצור מערכת יישום.