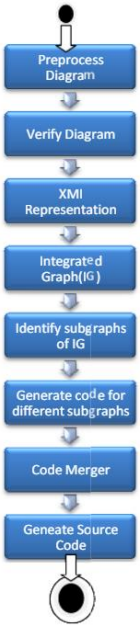


שלבי בסיס בתהליך יצירת קוד אוטומטי:

Preprocessor:

זהו התהליך הראשוני שעובר התרשים המקורי על מנת להיבנות מחדש בצורה שתהיה נוחה

יותר לעיבוד ולקריאה בתהליך יצירת הקוד.

Verifier:

בשלב זה, התרשים שנבנה מחדש עובר תהליך אימות על מנת לחפש שגיאות בתרשים, או לבדוק האם יש סכנה לגבי שגיאות עתידיות בתהליך קומפילציה.

XMI Representation:

XMI – XML Meta-data information, באה לתווך בין קבוצות תכנות שונות הפועלות עם תרשימי UML שמנוסחים בדרכים שונות ומשתמשים בכלים שונים, ולאפשר לצוותים אלו להעביר מידע מצוות אחד לאחר. במילים פשוטות XMI מגדירה באמצעות XML דרך סטנדרטית להעברת מטה-מידע ובנוסף מגדירה מהו סט של מידע, איך הוא בנוי ומה הוא מכיל.

בשלב זה נבנית תצוגת ה-XMI של המערכת המכילה את המטה-מידע עבור יצירת תתי התרשימים ה-Integrated Graphs, עבור תהליך יצירת הקוד.

Construction of IG: בשלב זה נבנים תתי התרשימים לפי גישות שונות הנדרשים לצורך יצירת הקוד המייצג את כל התרשים המקורי.

Code Merger: שלב זה הוא השלב הסופי, לאחר שכל הנתונים נותחו ותתי התרשימים קיימים בסדר הנדרש מופקים קטעי קוד המייצגים את תתי הגרפים באמצעות אלגוריתמים שונים ולבסוף מחוברים יחד והתוצר הוא קוד המתאר את התרשים המקורי.

גישות ליצירת קוד באופן אוטומטי:

ישנן שתי גישות מרכזיות ליצירת קוד באופן אוטומטי: Visitor-Based, Template-Based.

Visitor-Based: המטרה בגישה זו, היא לעבור באופן איטרטיבי על אלמנטים גרפיים של תרשימי ה-UML.

Template-Based: המטרה בגישה זו, היא ליצור מאגר של תבניות המהוות אוסף חוקים שלפיהם מייצרים קוד אוטומטי מתרשימים ספציפיים או ממודל ספציפי במערכת. גישה זו היא יותר שכיחה מאחר ובעזרתה ניתן לייצר קוד המקיף באופן רחב את המערכת המנוסח באופן קריא וברור.

אלגוריתמים ליצירת קוד באופן אוטומטי:

מצאנו שלושה אלגוריתמים שונים התורמים ליצירת קוד באופן אוטומטי:

1.Code-Munger: אלגוריתם זה הינו אלגוריתם פשוט אשר עובר על הקלט (תרשים UML) ומחפש אחר תגיות, מילות מפתח ומושגים נוספים המהווים חלקים מרכזיים במערכת ומייצר מהם פלט. אלגוריתם זה משמש על מנת לייצר תיעוד נרחב על המערכת.

Partial-Class Generator.2 : אלגוריתם זה, מקבל כקלט קובץ של הגדרות המכיל תבניות והגדרות קריטיות עבור יצירה של מחלקות. הפלט של האלגוריתם הוא קובץ של שלדים או בסיסים למחלקות שלאחר מכן נותר רק להשלים את הפונקציונליות הסופית.

Tier Generator.3: אלגוריתם מעט דומה ל-Partial-Class Generator רק להבדיל, אלגוריתם זה מקבל כקלט תרשים UML, ממלא עבורו קובץ קונפיגורציה המזכיר את קובץ ההגדרות אצל ה- Partial Class Generator, ובעזרת קובץ הקונפיגורציה הוא מפתח חלקים מסוימים מהמערכת ואף מוסיף פונקציונליות לעיתים מלאה של חלק מהשלבים או שלב במערכת.

<u>תבניות לייצור קוד באופן אוטומטי:</u>

על מנת להסביר את התבניות, תחילה נצטרך להסביר מהי תבנית – Template.

תבנית – Template היא קובץ המכיל קטעי קוד מסוימים אך במקום משתנים או שמות משתנים יש מקום להכניס נתונים.

מנוע תבניות – Templates Engine הוא תוכנה אשר יודעת למלא את התבניות במידע הנדרש כך שיפעלו לפי כללי לוגיקה המוגדרים בשפת התבניות.

Templates and Filtering.1: על פי תבנית זו, בהינתן תרשים UML, מבצעים ניתוח על התרשים על פי סינונים שמביאים לתתי תרשימים של התרשים המקורי בצורה של תתי קבוצות (כך שאם נחבר את תתי הקבוצות נקבל את הקבוצה הגדולה). תתי הקבוצות עוברות ניתוח ופירוקים נוספים עד שמתקבלים תתי קבוצות כך שלכל תת קבוצה (תת-תרשים) ניתן למלא תבנית – Template שתייצג את קטע הקוד עבור תת הקבוצה המתאימה.

הבעיות בתבנית זו, היא קושי בתחזוקה מאחר ואם נרצה להוסיף ולהגדיל את המערכת אז תהליך יצירת הקוד יקרה מחדש, והקוד החדש שנקבל יכול להיות שונה מהקוד לפני השינוי, מה שיגרום לקושי בתחזוקה.

קושי הוא נוסף הוא הסיבוכיות שיוצרות התבניות, במצב של מערכת גדולה ורחבה על מנת לייצר קוד לכל האלמנטים במערכת לעיתים התבנית תשתמש ביותר תבניות – Templates על מנת לייצר את הקוד, מה שיוביל להגדלת הסיבוכיות בקוד.

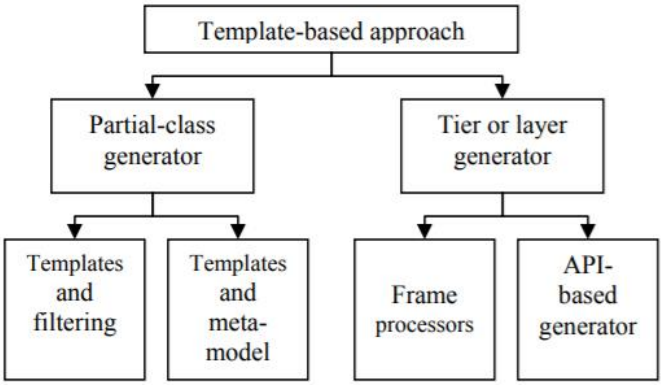
API-based Generator.2: תבנית זו בדרך כלל מזוהה עם שפת תכנות אחת, ועל כן נותנת למשתמש שלד המיוחד לשפת התכנות הספציפית ועוזרת למשתמש באמצעות כלים כמו קומפיילר לעבוד בצורה נכונה עבור השפה המבוקשת. תבנית זו משתמשת גם ב-Templates על מנת לממש את הקוד.

<u>זיהוי הגישות ליצירת קוד עם האלגוריתמים והתבניות ליצירת קוד:</u>

Template-based: מאחר וגישה זו מזוהה עם תבניות – Templates, לכן היא גם מזוהה עם אלגוריתם ה-Partial-class generator שפועל עם תבניות ה:Templates and Filtering ו-

Templates and Meta-model. אלגוריתם ה-Partial-class generator והתבניות שהוא פועל איתן מייצרים רק את הקוד של גוף המערכת, ללא תוספות פונקציונליות או שינויים מיוחדים.

אלגוריתם ה-Tier or Layer generator פועל בדרך מסובכת יותר ומייצר חלקי קוד של המערכת, לכן נזהה את אלגוריתם זה עם ה-API-based generator אשר נותן כלים נוספים למשתמש על מנת למנוע טעויות, כלים נוספים.



Visitor-based:

אלגוריתם ה-Code Munger מזוהה עם גישה זו מאחר והוא עובר באופן איטרטיבי על תרשים ה-UML ומחפש אחר אלמנטים טקסטואליים אשר מהווים תגיות ומילות מפתח ולפתח מהן תיעוד נרחב על המערכת.