УСТОЙЧИВОСТЬ СОВЕРШЕННЫХ АДАПТИРОВАННЫХ СЛОВ В МОДЕЛЯХ РАЗВИТИЯ СЛОИСТЫХ СТРУКТУР

Гурьянов В.И.

Региональный институт психологии и гуманитарных наук

Чебоксары, Россия

В последние годы заметное распространение получила архитектура программного обеспечения со слоистой структурой [1,2]. Это позволяет сформулировать относительно простые механизмы реструктуризации и построить целостную модель жизненного цикла систем автоматизации.

Процесс развития сложной системы имеет много общего с поведением динамических систем, хотя и имеет свои специфические особенности. Рассмотрим один из аспектов устойчивости процесса адаптации структур.

Пусть задан интерфейс бизнес-процесса $Is = (f_0, f_1, \dots, f_n)$, где $f \in Is$ идентификаторы элементов интерфейса (далее - прагматики). Особенность слоистых структур состоит в направленности связей со слоя i на слой $i+1$, но не наоборот. Композицию системы опишем как слово формального языка Lu^* с алфавитом $V_T = \{a_0, a_1, \dots, a_n\}$. Сопоставим каждой букве алфавита $V_T = \{a_0, a_1, \dots, a_n\}$ прагматику $f \in I_{S0}$ и определим фактор-множество $V_T/F = \{f_0|\{a_1, a_2, \dots a_k\}, f_1|\{b_1, b_2, \dots b_l\}, \dots, f_N|\{z_1, z_2, \dots z_m\}\}$, где $a_j, b_j, \dots, z_j \in V_T$. Структура $w_0 \in Lu$ является адаптированной, если имеет место $Is = I(w_0)$, где $Is = (f_0,f_1,\dots,f_N)$, $I(w_0) = (f_0',f_1',\dots,f_N')$ и $N' = N$. Слово $w_0 \in Lu$ назовем совершенной адаптированной структурой, если в адаптированной структуре w_0 насыщены все актуальные связи. Порядок сборки слов определяется некоторой решающей грамматикой $G(V_T,V_N,R,S)$, $Ls^+(G) \subseteq Lu^+$. Ограничимся решающей грамматикой, описываемой реляционной алгеброй.

Будем исходить из гипотезы о существование универсального закона развития сложных систем. Предположим, что процесс развития представляет собой совокупность трех взаимосвязанных процессов и включает: цикл регенерации, цикл перебора альтернатив и процесс генерации решающих грамматик. Совокупность правил, определяющих процесс генерации грамматик, назовем императивной логикой $\aleph$ конструирования решающих грамматик.

Типичный пример императивной логики содержит правила генерации адаптированной решающей грамматики по формуле соединения $R^{(k)} = R^{(k-1)} \Pi S_j$, где R- история разработки, S_j домен фактор-множества V_T/F по атрибуту f_j. Расширение алфавита по домену выполняется операцией $E_j(G)$ для f_j.

Перейдем к рассмотрению вопроса устойчивости процесса развития. Пусть совершенное адаптированное слово $w_0 \in Ls$ уже построено.

Рассмотрим причины, которые могут вызывать дефекты в структуре w_0. Одна, очевидная причина - изменение интерфейса Is в результате изменения бизнес-процесса, другая - ошибки сборки. Наконец, наиболее интересный случай – внесение случайных ошибок в процесс перебора альтернатив.

В отдельный класс следует выделить процессы, обусловленные особенностями $\aleph$- логики. Возможны ситуации, когда $\aleph$-логика успешно строит решающую грамматику, но приводит к циклам при малых нарушениях в структуре w_0 . Приведем пример такого рода.

Пусть известна решающая грамматика вида $V_T = \{a,b,b',c,d\}$, $V_N = \{S,A,B\}$, R: $(S \to aA,\ aA \to abB \mid ab'B,\ B \to c \mid d;\ A, B \to l)$; $w_0 = abc$. Расположим ветви дерева в том порядке, как они указаны в картеже R. Первая и вторая ветвь – закрыты, четвертая ветвь не исследована.

Ошибки 1-го рода. Пусть в результате ошибки использован шаблон $S|\text{-}\ aA_1|\text{-}\ abB|\text{-}\ abd$. Т.к. слово не совершенно, ветвь 4 закрывается и начинается поиск в списке альтернатив. В нашем случае находится ветвь $S|\text{-}\ aA_1|\text{-}\ abB|\text{-}\ abc$, которая приводит к совершенному слову.

Ошибки 2-го рода. Пусть в результате ошибки на текущем выводе ставится маркер «Закрыто». Исследование 4-ой ветви закроет ее и процесс вызывает генерацию грамматики, пытаясь расширить алфавит текущего домена. Расширение будет тождественным. После чего опять повторится цикл перебора альтернатив.

Таким образом, возникает бесконечный цикл обращения к $\aleph$- логике. В качестве шаблона используется последний вывод. Для выхода из этой ситуации в $\aleph$- логике должны быть предусмотрены дополнительные правила. Для активных систем подобные механизмы существуют. Если же используются базы опыта, то возможно возникновение бесконечных циклов.

Итак, требование устойчивости процесса развития структур накладывает дополнительное условие на императивную логику.

СПИСОК ЛИТЕРАТУРЫ:

1. Ксензов М.В. Рефакторинг архитектуры программного обеспечения, М.: ИСП РАН, Препринт 4. - Москва, 2004.
Вильямс Мартин Фаулер. Архитектура корпоративных программных приложений. - Москва, 2004.