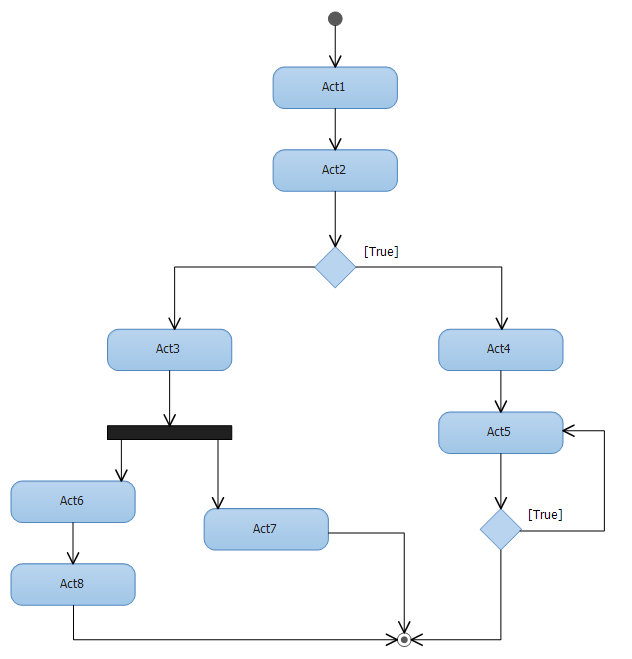
Aplikacja do analizy diagramów aktywności UML

1. Identyfikowane struktury diagramu aktywności

|  |  |  |
| --- | --- | --- |
| **typ** | **budowa** | **wynikowy zapis** |
| **przepływ** – reprezentuje przejście z jednej akcji do drugiej |  | *Flow (Act1, Act2)* |
| **warunek** – reprezentuje wybór pomiędzy kilkoma możliwościami |  | *Dec(Dec1, Act4, Act3)* |
| **rozwidlenie** – reprezentuje przejście wejściowego przepływu sterowania w co najmniej dwa wynikowe |  | *Fork(Fork1, Act6, Act7)* |
| **pętla** – reprezentuje akcję wykonywaną, dopóki spełniony jest warunek |  | *Loop(Dec1, Act5, END)* |

1. Przykład analizy

Rozpatrywany diagram:



Algorytm tworzenia zapisu sekwencji:

1. Identyfikacja przepływów
   * Flow(START, Act1)
   * Flow(Act1, Act2)
   * Flow(Act2, Dec1)
   * Flow(Act3, Fork1)
   * Flow(Act4, Act5)
   * Flow(Act5, Dec2)
   * Flow(Act6, Act8)
   * Flow(Act7, END)
   * Flow(Act8, END)
2. Identyfikacja Warunków
   * Dec(Dec1, Act4, Act3)
   * Dec(Dec2, Act5, END)
3. Identyfikacja rozwidleń
   * Fork(Fork1, Act6, Act7)
4. Połączenie przepływów ze sobą
   * Flow(START, Flow(Act1, Flow(Act2, Dec1)))
   * Flow(Act4, Flow(Act5, Dec2))
   * Flow(Act6, Flow(Act8, END))
5. Podstawienie w połączonych przepływach warunków i rozwidleń w miejscach ich nazw
   * Flow(START, Flow(Act1, Flow(Act2, Dec(Dec1, Act4, Act3))))
   * Flow(Act4, Flow(Act5, Dec(Dec2, Act5, END)))
   * Flow(Act3, Fork(Fork1, Act6, Act7)
6. Podstawienie przepływów w miejscach nazw aktywności docelowych
   * Flow(START, Flow(Act1, Flow(Act2, Dec(Dec1, Flow(Act4, Flow(Act5, Dec(Dec2, Act5, END))) , Flow(Act3, Fork(Fork1, Flow(Act6, Flow(Act8, END)), Flow(Act7, END)))))))
7. Identyfikacja pętli i uzyskanie ostatecznej postaci sekwencji
   * Flow(START, Flow(Act1, Flow(Act2, Dec(Dec1, Flow(Act4, Loop(Dec2, Act5, END)), Flow(Act3, Fork(Fork1, Flow(Act6, Flow(Act8, END)), Flow(Act7, END)))))))
8. Modelowanie diagramów aktywności i ich postać xml

Do modelowania diagramów aktywności użyliśmy komponentu Visual Studio 2012 Ultimate, który pozwala modelować architektury zgodnie ze standardem UML 2.0. Narzędzie to jest z jednej strony proste w użyciu, z drugiej umożliwia integrację modeli z cyklem tworzenia oprogramowania. Na uwagę zasługuje fakt, że tworzone diagramy składają się z dwóch plików xml – struktury i wyglądu osobno. Dzięki temu mogliśmy operować na przejrzystych plikach, których budowa była łatwa do zrozumienia.

Ogólna i pozbawiona większości atrybutów struktura pliku xml diagramu aktywności przedstawiona jest na poniższym listingu. Stworzony przez nas diagram UMLActivityDiagram1 przedstawia aktywność Activity1, której nazwa zawarta jest w korzeniu. Element package nie jest dla nas interesujący. Cała budowa diagramu aktywności zawarta jest we wnętrzu elementu nodes elementu activity.

Listing – ogólna struktura pliku xml diagramu aktywności:

<?xml version="1.0" encoding="utf-8"?>

<activityRootModel name="Activity1" >

<packagedElements>

<activityRootModelHasActivity>

<activity name="Activity1" >

<elementDefinition />

<nodes>…</nodes>

</activity>

</activityRootModelHasActivity>

</packagedElements>

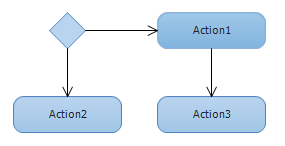
<package name="ModelingProject1">…</package>

</activityRootModel>

Poniższa tabela przedstawia sposób zapisu składników diagramu aktywności znajdujących się w elemencie nodes.

|  |  |  |
| --- | --- | --- |
| **typ** | **budowa** | **zapis xml** |
| początek przepływu |  | *initialNode* |
| koniec przepływu |  | *activityFinalNode* |
| akcja |  | *opaqueAction* |
| decyzja |  | *decisionNode* |
| rozwidlenie |  | *forkNode* |

Poniżej przedstawiono prosty przepływ sterowania oraz reprezentację xml. Pokazuje ona, że przepływy są zapisywane w węzłach, z których wychodzą. Dwa przepływy wychodzące z węzła decyzyjnego zdefiniowane są jako elementy controlFlow zawarte w flowNodeTargets. Przepływy controlFlow definiują też elementy docelowe, których nazwy są analogiczne do nazw podstawowych składników diagramu i zawierają w nazwie dodatkowo Moniker.

<nodes>

<decisionNode name="Decision1">

<elementDefinition />

<flowNodeTargets>

<controlFlow>

<opaqueActionMoniker LastKnownName="Action1" />

<elementDefinition />

</controlFlow>

<controlFlow>

<opaqueActionMoniker LastKnownName="Action2" />

<elementDefinition />

</controlFlow>

</flowNodeTargets>

</decisionNode>

<opaqueAction name="Action1">

<elementDefinition />

<flowNodeTargets>

<controlFlow>

<opaqueActionMoniker LastKnownName="Action3" />

<elementDefinition />

</controlFlow>

</flowNodeTargets>

</opaqueAction>

<opaqueAction name="Action2">

<elementDefinition />

</opaqueAction>

<opaqueAction name="Action3">

<elementDefinition />

</opaqueAction>

</nodes>

Wszystkie elementy diagramu zawierają też atrybut Id w formie GUID, który został pominięty ze względu na czytelność oraz nazwy: name lub LastKnownName. Sugeruje to, że najlepszym sposobem rozpoznawania elementów jest ich identyfikator. Nie napotkaliśmy jednak problemów posługując się nazwami.