利用情報解析によるコンポーネントウィザードの自動生成

4J−5

安達 孝夫　鷲崎 弘宜　深澤 良彰
早稲田大学理工学部

1　はじめに

現在、多くのソフトウェア開発ツールは、利用者（アプリケーション開発者）との対話により、複雑な設定を容易に定義するウィザード機能を提供する[1]。さらにコンポーネント指向開発において、ウィザード機能を個々のコンポーネント自身に持たせる試みがある[2]。コンポーネントウィザードは、コンポーネント開発者が想定する推奨利用法として、コンポーネントの持つプロパティの初期値、及び初期値の組合せ(設定セット)を提示し、そのビジュアルな選択により初期設定を行う。

しかし、ウィザード機能の実装には大きな開発コストがかかり、実際に同機能を提供するコンポーネントは少数である。本稿では、サンプルアプリケーションの解析により得られるコンポーネントの利用情報を用いて、ウィザード機能を自動付加する手法を提案する。

本手法では、サンプルアプリケーション中で、コンポーネント生成直後に設定されているプロパティの情報を元に、その利用情報解析を行う。複数のサンプルからプロパティの設定される値の組合せを導出し、その組合せを連続するダイアログ群で提示することによりコンポーネント利用の支援を行う。図1の例では、コンポーネントの背景色と前景色の組合せについて具体的な初期値のセットを3つ選択肢として提示している。

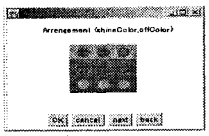

図 1: コンポーネントウィザードの実行例

2　利用情報解析

対象とするコンポーネントが実際に使用されている複数のサンプルアプリケーションを事前に収集し、コンポーネントが持つプロパティについて、以下のように解析を進めていく。まず、対象コンポーネントを利用しているサンプルアプリケーションの動作をトレースし、メソッド呼出しグラフを作成する。グラフ G は、メソッドを示すノードの集合 M と、メソッドからのメソッドの呼出しを示すエッジの集合 E から構成される。

グラフ $G=(M,E)$, メソッド集合 $M=\{m_0,...,m_n\}$
呼出し集合 $E=\{e_0,...,e_{n-1}\}$
呼出し $e_i=(m_{source},m_{target})(0<i<n)$

メソッド呼出しグラフ G の例を図2に示す。ノードは、メソッド情報とプロパティを表すラベルを持つ。

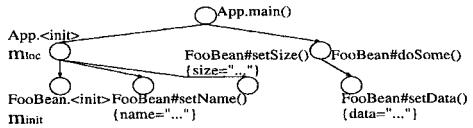

図 2: メソッド呼出しグラフ

対象コンポーネントの生成メソッド m_{init} を呼び出すメソッドを m_{loc} とする。メソッド m の初期設定度 $init(m_i)$ は、グラフ上でコンポーネントが生成された位置からみた m の近さを表す。$init(m)$ の定義を以下に示す。

$d(m_i,m_j)=$(グラフ上のm_iとm_jの間のノードの数)
$init(m)=\frac{1}{d(m_{loc},m)}$

FooBean のプロパティ操作 setSize(m_1) の初期設定度は1であり、プロパティ操作 setData(m_2) の初期設定度は1/3である。同じプロパティが複数設定されている場合は、同一プロパティの初期設定度の平均をとる。

プロパティについて、ウィザードで初期設定するには、頻繁に初期値として設定される値を得る必要がある。このような初期値の推定は、サンプルアプリケーションを実際に実行して、その実行時の各プロパティの値の変更状況を観測することで行う。

設定セットとは、コンポーネントを初期設定するにあたり、多くのサンプルアプリケーションで同一な複数のプロパティの初期値の組である。設定セットの導出には、以下に定義するプロパティ関連度を用いる。

プロパティ p を設定するメソッド m の集合を $a(p)$ とする。同じアプリケーション内でプロパティ p_i,p_j がそれぞれ設定操作であるメソッド m_i,m_j で X 回、Y 回設定されるとき、以下を得る。

$a(p_i)=\{m_{i1},m_{i2},\cdots,m_{ix}\}(1\leqq x\leqq X)$
$a(p_j)=\{m_{j1},m_{j2},\cdots,m_{jy}\}(1\leqq y\leqq Y)$

このときプロパティ関連度 $pr(a(p_i),a(p_j))$ を以下に定義する。

$$pr(a(p_i),a(p_j))=\frac{\sum_{y=1}^{Y}\sum_{x=1}^{X}\frac{1}{d(m_{ix},m_{jy})-1}}{XY}$$

Automatic Generation of Software Component Wizards by Usage Analysis
Takao Adachi, Hironori Washizaki and Yoshiaki Fukazawa.
School of Science & Engineering,Waseda University.

図2より、FooBeanのプロパティ設定操作setSizeとsetNameのプロパティ関連度は1であり、プロパティ設定操作setSizeとsetDataのプロパティ関連度は1/3である。

次に複数の設定セットPについて、提示する順序を測定する。全プロパティの全ての設定セットPの集合をSとする。例えば、プロパティ数n=3の時、$S=\{p_1,p_2,p_3,(p_1,p_2),(p_1,p_3),(p_2,p_3),(p_1,p_2,p_3)\}$である。これを全設定セットで設定セット推奨値$f(P)$を算出し、設定セットの提示は、この値が大きなものを優先する。なお、値が同じ場合は、設定セットPの要素数$|P|$が大きいほうを優先する。

$$\mathrm{f(P)}=\begin{cases}\dfrac{\sum_{i=1}^{|P|-1}\sum_{j=i+1}^{|P|}pr(a(p_i),(p_j))}{\frac{|P|(|P|-1)}{2}}\dfrac{\sum_{k=1}^{|P|}init(m_k)}{|P|} & (|P|>1)\\ init(m_i) & (|P|=1)\end{cases}$$

3 ウィザードの自動生成

コンポーネントシステムとしてはJavaBeansを対象とする。サンプルアプリケーションのトレースは、アプリケーション中で利用されるクラスにバイトコード編集[1]によってトレース機能を付加することで実現する。

ウィザードの自動生成について、設定セットなどをパラメータ化して変更可能としたウィザードの雛形(ウィザードフレームワーク)[3]を準備する。雛型には、ウィザードが持つ共通要素についてはあらかじめ定義し、対象毎に変化する設定項目について、XML形式の項目記述を外部から与えることで最終的なウィザードを作る。

4 解析例

文献[2]のWEbeanを使ったサンプルアプリケーションApp1 ~ 3に対して解析を行った。WEbeanは光が点滅するBeanである。App1は点滅をボタンで開始させるアプリケーション、App2は点滅速度を変更できるアプリケーション、App3はbackgroundを設定する以外はApp2と同じアプリケーションである。

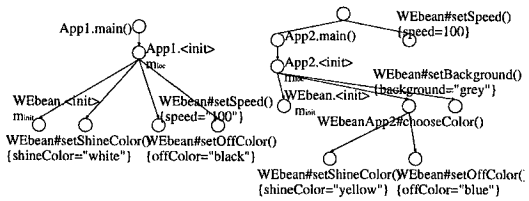

図3: メソッド呼出しグラフ(左App1, 右App3)

プロパティ設定に関連する部分のApp1、App3のメソッド呼出しグラフを図3に示す。メインメソッドとは別スレッドで実行されるメソッド群は、別のメソッド呼出しグラフを生成したのち、各グラフのノードの上に共通のノードを作成して繋げるものとする。App2のメソッド呼出しグラフはApp3のものからWEbean#setBackground()のノードを削除したものと同じである。

それぞれのアプリケーションで初期設定値とプロパティ関連度を計測した結果を表1、表2に示す。これらの情報からプロパティ提供の優先順位を決定する。以降、プロパティについて、shineColorはsc、offColorはoc、speedはsp、backgroundはbgと略記する。

表1: 初期設定度

init(m)	setShineColor	setOffColor	setSpeed	setBackground
App1	1	1	1	-
App2	0.5	0.5	0.25	-
App3	0.5	0.5	0.25	1

表2: プロパティ関連度

$pr(a(p_i),a(p_j))$	sc,oc	sc,sp	sc,bg	oc,sp	oc,bg	sp,bg
App1	1	1	-	1	-	-
App2	1	0.25	-	0.25	-	-
App3	1	0.25	0.5	0.25	0.5	0.33

表3は、$f(P)$の測定結果の抜粋である。まず、bgの値が1で、他にこれと同じ値を持つものがないので、コンポーネントと最も密接に関係したプロパティをして最初に提示する。次に、scと(sc,oc)の両セットの値が0.67で同じであるが、設定セットの要素数が大きいものを優先するので(sc,oc)を提示する。最後に残ったspeedを提示する。

表3: 抽出データ(抜粋)

P	sc	sp	bg	sc,oc	sc,sp bg	sc,oc sp,bg
f(P)	0.67	0.5	1	0.67	0.21	0.27

設定セットを提示する際に、組合せと共に初期値の候補を提示する。例えば、設定セット(sc,oc)について図3より、{sc="white", oc="black"}、{sc="yellow", oc="blue"}を候補として提示する。

5 おわりに

アプリケーション中のコンポーネントを利用する情報を解析することで、コンポーネントの初期設定に必要な情報を取得し、開発者の意図に沿ってコンポーネントを初期設定可能とするウィザード機能を自動生成する手法を提案した。現在、プロトタイプの実装を終えた段階であり、今後、実用可能なシステムの構築とその有用性を示す予定である。

参考文献

[1] S.Chiba, "Javassist – A Reflection-based Programming Wizard for Java," Workshop on Reflective Programming in C++ and Java, 1998

[2] 坂本衛, "JavaBeansプログラミング入門", オーム社, 1997

[3] C.Sheong, "Build Wizards Quickly Using a Swing-Based Wizard Framework," JavaReport, May. 2001