Wiesław WINIECKI, Piotr BOBIŃSKI
POLITECHNIKA WARSZAWSKA, INSTYTUT RADIOELEKTRONIKI

Automatyczna generacja interfejsu użytkownika dla mobilnych klientów pomiarowych serwisów sieciowych

Dr hab. Inż. Wiesław WINIECKI

Prof. nzw. na Wydziale Elektroniki i Technik Informacyjnych PW. Kierownik zespołu Komputerowej Techniki Pomiarowej. Aktualne obszary zainteresowań: systemy pomiarowe, przyrządy wirtualne, nowoczesne technologie komunikacyjne i programowe w skupionych i rozproszonych systemach pomiarowo-kontrolnych. Prezes Polskiego Stowarzyszenia Pomiarów, Automatyki i Robotyki POLSPAR, członek Sekcji Systemy Pomiarowe w Komitecie Metrologii i Aparatury Naukowej PAN, członek IEEE.

e-mail: W.Winiecki@ire.pw.edu.pl

Dr inż. Piotr BOBIŃSKI

Adiunkt na Wydziale Elektroniki i Technik Informacyjnych PW. Autor lub współautor ponad 20 publikacji naukowych. Aktualne obszary zainteresowań: nowoczesne technologie komunikacyjne i programowe w skupionych i rozproszonych systemach pomiarowo-kontrolnych, systemy multimedialne, cyfrowe przetwarzanie sygnałów audio.

e-mail: pbo@ire.pw.edu.pl

Streszczenie

W referacie przedstawiono koncepcję budowy aplikacji mobilnego klienta rozproszonego systemu pomiarowego. Zarysowano projekt systemu pomiarowego, który udostępnia swoje usługi jako serwis sieciowy, a więc umożliwia dostęp do świadczonych usług w sposób uniwersalny i niezależny od platformy klienta. Opisano aplikację klienta zrealizowaną na platformie J2ME i przeznaczoną dla przenośnych terminali MDA, która na podstawie specyfikacji GUI udostępnianej przez serwis sieciowy potrafi zbudować odpowiedni interfejs użytkownika. Podejście to pozwala na zaprojektowanie interfejsu użytkownika od razu w fazie projektowania jądra systemu pomiarowego, a jednoczenie nie ogranicza dostępu do systemu pomiarowego ze standardowych klientów zgodnych ze specyfikacją Serwisów Sieciowych.

Automatic generation of user interfaces for mobile clients of measurement web services

Abstract

The paper presents a concept of distributed measurement system's mobile client application. An example of measurement system available as a web service is shown. Such system enables access to its services in universal and platform independent way. Additionally, expect standard stricte "measurement" functions, service provides information that describes the graphical user interface for its management. The paper presents an implementation of mobile client application in Java 2 Micro Edition environment for MDA terminals. The client is able to build its own graphical user interface based on the definition received from the web service. It allows the measurement system developer to design the client user interface at the same time that the system kernel is designed. Such universal approach simplifies access to web service from terminals with build in developed software (GUI Generator) and also allows access from other clients, compatible with web services specification.

1. Wstęp

Popularność urządzeń typu PDA (*Personal Digital Assistant*) w dzisiejszych czasach rośnie z dnia na dzień. Wykorzystanie nowoczesnych technologii komunikacji bezprzewodowej, zarówno telefonii komórkowej GSM czy UMTS, jak i technologii Bluetooth czy Wireless Ethernet (WLAN), umożliwiło realizację aparatury pomiarowo-kontrolnej sterowanej zdalnie, np. z telefonu komórkowego czy przenośnego komputera. Integracja współczesnych zaawansowanych technologii komunikacyjnych oraz urządzeń klasy PDA stwarza jeszcze większe pole dla rozwoju nowoczesnych, bezprzewodowych systemów pomiarowych. Rola tych nowoczesnych urządzeń, określanych często skrótem MDA (*Mobile Digital Assistant*).

Jednym z ciekawszych narzędzi programowych, z punktu widzenia projektowania rozproszonych systemów pomiarowych, jest wykorzystanie języka Java, a dokładnie jego specjalnej edycji - platformy J2ME (*Java 2 MicroEdition*), przeznaczonej właśnie dla tego typu urządzeń.

Podstawowa wada platformy J2ME w kontekście projektowania rozproszonych systemów pomiarowych to brak pełnych, specjalizowanych bibliotek, dedykowanych zastosowaniom pomiarowym, a więc zawierających gotowe obiekty graficzne wykorzystywane do projektowania paneli RSP. Problem ten można rozwiązać poprzez implementację własnych komponentów graficznych i komunikacyjnych [1].

Kolejnym interesującym zagadnieniem jest wykorzystanie do projektowania i implementacji rozproszonych systemów pomiarowych technologii Serwisów (Usług) Sieciowych (*Web Services*) [2], które stanowią neutralną platformę do zdalnego wywoływania procedur poprzez sieć Internet. Połączenie technologii usług sieciowych wraz z narzędziami platformy J2ME pozwala na zbudowanie mobilne klienta systemu pomiarowego, udostępnianego jako serwis sieciowy, który potrafi automatycznie zbudować swój interfejs graficzny na podstawie specyfikacji dostarczonej przez serwer.

W rozdziale 2 omówiono w skrócie technologię serwisów sieciowych oraz wymieniono jej główne składniki. W rozdziale 3 zarysowano kluczowe elementy platformy J2ME, istotne z punktu widzenia niniejszej pracy. W rozdziale 4 przedstawiono realizację postawionego zadania. Opisano definicję specyfikacji graficznego interfejsy użytkownika oraz przykładowy serwis udostępniający żądaną funkcjonalność. Dalej zaprezentowano zrealizowane narzędzia programowe oraz mobilnego klienta automatycznie tworzącego swój własny interfejs graficzny.

2. Serwisy Sieciowe

Idea Serwisów Sieciowych jest prosta: używanie Internetu do publikowania i dostarczania oprogramowania, z którego może korzystać dowolna liczba zarejestrowanych użytkowników. Serwisy mogą działać w dowolnej dziedzinie - od przeliczania walut, sprawdzania wiarygodności kart kredytowych po zarządzanie produkcją. Aby mogły działać w praktyce, serwisy sieciowe muszą potrafić komunikować się ze sobą niezależnie od języka programu, czy systemu operacyjnego. Tylko wtedy będzie możliwe połączenie wzajemne oraz z już istniejącymi systemami informatycznymi.

Serwisy sieciowe stanowią kolejny krok ku dojrzałości rozproszonego przetwarzania w wielu dziedzinach. W skład serwisu wchodzą różne standardy komunikacyjne, które mogą być realizowane za pomocą dowolnych narzędzi oraz przez różnych dostawców. Główne cele stosowania usług sieciowych to:

- tworzenie aplikacji integrujących istniejące rozwiązania.
- integracja/agregacja usług oferowanych przez oddziały firmy lub całe firmy.
- integracja niezależnych od siebie usług opartych o odmienne podejścia.

Trzy główne składowe koncepcji serwisów sieciowych to [2]:
1. UDDI (*Universal Description, Discovery and Integration*) - protokół opisu dostępnych składników serwisu sieciowego umożliwiający rejestrację, reklamowanie i automatyczne wyszukiwanie usług.
2. SOAP (*Simple Object Access Protocol*) -protokół inicjalizacji konwersacji pomiędzy modułami opisanymi przez UDDI. Moduły (obiekty) rezydujące na zdalnych komputerach udostępniają pewne swoje metody (lub funkcje) do zdalnych wywołań, a aplikacja SOAP tworzy blok żądania dostępu w XML, dostarczając dane na potrzeby wywołania oraz identyfikując lokalizację modułu (obiektu).
3. WSDL (*Web Service Description Language*) - standard opisu serwisu sieciowego oraz charakterystyki implementacji w języku typu IDL (*Interface Definition Language*) opakowanego w XML.

Powiązania pomiędzy tymi składowymi są takie, że wpisy w języku WSDL w rejestrach UDDI opisują komunikaty SOAP definiujące konkretny serwis sieciowy.

Powiązania te można przedstawić opisując w kilku krokach schemat implementacji serwisu sieciowego:
- dostawca tworzy serwis
- dostawca opisuje swój serwis w języku WSDL (na potrzeby rejestru UDDI)
- dostawca rejestruje swój serwis w UDDI
- inny serwis lub użytkownik lokalizuje i żąda dostępu do serwisu przez zapytania do UDDI
- inny serwis lub użytkownik tworzy aplikację korzystającą z odnalezionego wcześniej serwisu (komunikacja przez SOAP)
- dane i komunikaty wymieniane są w postaci XML najczęściej po protokole HTTP.

3. Platforma Java 2 Micro Edition i serwisy sieciowe

Platforma Java 2 Micro Edition (J2ME) jest edycją technologii Java zaprojektowaną specjalnie z myślą o urządzeniach o ograniczonych zasobach i niewielkiej mocy obliczeniowej. W 1999 roku wyodrębniono ją z edycji Standard, a jej rozwój powierzono grupom ekspertów działających w ramach inicjatywy JCP (*Java Community Process*). Wynikiem ich pracy jest wiele standardów oraz propozycji standardów dotyczących oprogramowania urządzeń przenośnych. Z tej przyczyny platformę J2ME, w przeciwieństwie do jednolitych edycji Standard i Enterprise, należy rozpatrywać raczej jako zbiór podobnych, lecz niezależnych od siebie technologii przeznaczonych dla poszczególnych klas urządzeń. Dla urządzeń przenośnych przeznaczony jest profil MIDP (*Mobile Information Device Profile*) [3], który jest częścią platformy J2ME i definiuje środowisko aplikacji dla urządzeń, takich jak telefony komórkowe i urządzenia PDA. Jest on nadbudowany na konfiguracji CLDC (*Connected Limited Device Configuration*) [4]. Aplikacje uruchamiane w środowisku MIDP nazywane są midletami, które pakietowane są w paczki i w ten sposób osadzane na przenośnych terminalach.

Rosnąca popularność serwisów sieciowych zaowocowała utworzeniem opcjonalnych pakietów dla platformy J2ME umożliwiających tworzenie klientów usług sieciowych na przenośnych terminalach i zdefiniowaniem specyfikacji serwisów sieciowych dla platformy J2ME (*J2ME Web Services Specification*)[5, 6]. Specyfikacja definiuje dwa opcjonalne pakiety oprogramowania:
- JAXP - przetwarzanie dokumentów XML (*Java API for XML Processing*),
- JAX-RPC - zdalne wołanie procedur (*Java API for XML-based RPC*).

Pakiet JAXP pozwala na analizę (tzw. parsowanie) dokumentów XML i stanowi podzbiór odpowiedniego pakietu ze standardowej edycji języka Java (J2SE). Dostarczony parser działa w modelu SAX (*Simple API for XML*), jest zgodny ze specyfikacją XML 1.0, ale nie daje możliwości walidacji dokumentów XML.

Pakiet JAX-RPC jest implementacją technologii zdalnego wołania procedur (RPC) i stanowi podzbiór odpowiedniego pakietu z platformy J2EE (*Java 2 Platform Enterprise Edition*). Pakiet umożliwia tworzenie przenośnych klientów, którzy w łatwy sposób z wykorzystaniem protokołu SOAP, mogą wywoływać procedury serwisów sieciowych. Zdalne metody wołane są poprzez odpowiednie interfejsy i dzięki temu ich wołania nie różnią się niczym od wołań metod lokalnych.

Istotnym elementem pakietu JAX-RPC jest generator plików pośredniczących w zdalnych wołaniach sieciowych usług (*Stub Generator*). Generator na podstawie opisu serwisu sieciowego w języku WSDL generuje interfejsy i klasy (zręby - *stubs*) potrzebne do komunikacji z serwisem oraz tworzy odpowiednią strukturę komunikatów SOAP i połączeń sieciowych. Komunikacja mobilnego klienta z serwisem sieciowym przebiega w następujący sposób:
1. Lokalna aplikacja woła odpowiednią metodę wygenerowanego zrębu,
2. Klasa zrębu tworzy odpowiednie komunikaty SOAP,
3. Maszyna wirtualna J2ME otwiera połączenie z serwisem sieciowym i wymienia z nim komunikaty SOAP,
4. Klasa zrębu przetwarza odpowiedź z serwisu i przekazuje wynik do lokalnej aplikacji.

4. Realizacja midletu

Realizację postawionego zadania można podzielić na trzy etapy. W pierwszym zdefiniowano strukturę dokumentu XML opisującego wygląd oraz funkcjonalność interfejsu graficznego zapewniającego sterowanie systemem pomiarowym. W drugim etapie opracowano serwis sieciowy, stanowiący atrapę systemu pomiarowego, udostępniający metody symulujące realizację wybranych funkcji pomiarowych oraz metodę zwracającą definicję interfejsu użytkownika. Trzecim etapem była właściwa implementacja oprogramowania dla mobilnego klienta.

Definicja GUI

W celu umożliwienia mobilnemu klientowi zbudowanie graficznego interfejsu, rozmieszczenie kontrolek oraz utworzenie odpowiednich procedur obsługi zdarzeń, konieczne stało się utworzenie definicji GUI, opisującej rozmieszczenie kontrolek w interfejsie oraz wszystkie interakcje pomiędzy nimi. Dla tego celu stworzono następującą definicję dokumentu XML. Element główny `gui` zawiera definicje elementów interfejsu `item`. Każdy element `item` poza atrybutami opisującymi rodzaj i wygląd kontrolki może zawierać elementy `action` opisujące akcje związane z tym elementem. Akcja opisana jest poprzez atrybuty elementu `action` i parametry - elementy `param`. Dostępne funkcje, czyli możliwe wartości parametru `function` to:
- `getValue` - pobranie wartości z kontrolki
- `setValue` - ustawienie kontrolki
- `call` - wywołanie metody z serwisu sieciowego

DefinicjęDTD dokumentu przedstawiono poniżej.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT gui (item+)>
<!ELEMENT item (action*)>
<!ATTLIST item
      id NMTOKEN #REQUIRED
      type CDATA #REQUIRED
      label CDATA #IMPLIED
      content CDATA #IMPLIED
      length CDATA #IMPLIED
      width CDATA #IMPLIED
      height CDATA #IMPLIED
      newline (before|after|both|none) "none"
>
<!ELEMENT action (param*)>
<!ATTLIST action
      function (getValue|setValue|call)"getValue"
      method CDATA #IMPLIED
      return  CDATA #IMPLIED
>
```

```
<!ELEMENT param EMPTY>
<!ATTLIST param
      id NMTOKEN #REQUIRED
>
```

Serwis Sieciowy

Do celów testowania opisywanej koncepcji utworzono prosty serwis sieciowy symulujący pracę systemu pomiarowego oraz udostępniający poza podstawowymi funkcjami pomiarowymi definicję przykładowego interfejsu graficznego. Symuluje się system wyznaczający charakterystykę częstotliwościową badanego obiektu.

Podstawowe funkcje serwisu to:

- `configMeasurement` - funkcja konfigurująca następujące parametry pomiaru: kształt sygnału, początkową i końcową częstotliwość, krok zmiany częstotliwości,
- `startMeasurement` - funkcja wyzwalająca pomiar, zwracająca identyfikator pomiaru,
- `getResults` - funkcja zwracająca wyniki pomiaru o danym identyfikatorze

Dodatkowa funkcja `getGUILayout` zwraca dokument XML definiujący interfejs użytkownika obsługi systemu. Przykładową definicję dla dwóch kontrolek GUI przedstawiono poniżej.

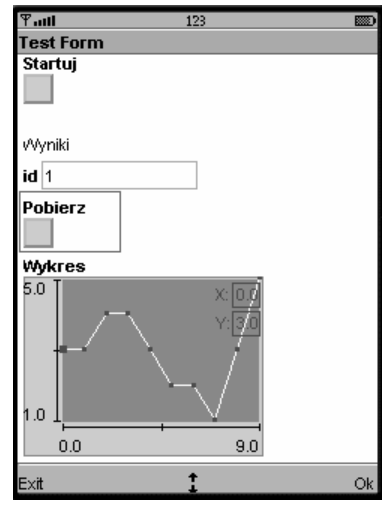
```
  <item id="get" type="Button" label="Pobierz" width="20"
  height="20">
        <action function="call" method="getResults"
        return="results">
              <param id="id"/>
        </action>
        <action function="setValue">
              <param id="chart"/>
              <param id="results"/>
        </action>
  </item>
  <item id="chart" type="Chart" label="Wykres"
  width="160" height="120"/>
</gui>
```

Mobilny Klient

Realizację oprogramowania rozpoczęto od wygenerowania odpowiednich interfejsów i klas zrębów na podstawie definicji funkcjonalności serwisu sieciowego dostarczonej przez sam serwis w postaci dokumentu WSDL. Skorzystano przy tym z narzędzia *StubGenerator* wchodzącego w skład zestawu narzędzi do implementacji mobilnych aplikacji J2ME WTK (*J2ME Wireless Toolkit*).

Kolejny etap stanowiła implementacja parsera XML, odpowiedzialnego za wczytanie dokumentu z definicją GUI i utworzenie obiektów reprezentujących poszczególne elementy panelu oraz obiektu zarządzającego zdarzeniami w systemie. Utworzono klasę *AutoTools*, w której zaimplementowano metodę odpowiednio rozmieszczającą elementy interfejsu na panelu oraz za metodę odpowiedzialną za obsługę zdarzeń. Ze względu na brak w J2ME bibliotek umożliwiających korzystanie z mechanizmu refleksji, który pozwoliłby między innymi na dynamiczne wołanie metod na podstawie ich nazwy, konieczne stało się zbudowanie narzędzia generującego odpowiednią klasę. Narzędzie nazwane *Caller Generator*, podobnie jak *StubGenerator*, na podstawie definicji serwisu w WSDL tworzy klasę *Caller*, która umożliwiającą wywołanie dowolnej metody z serwisu na podstawie jej nazwy i zbioru parametrów.

W ostatnim etapie zrealizowano mobilną aplikację (midlet) z wykorzystaniem zrealizowanych komponentów programowych. W konstruktorze midletu tworzone są wszystkie potrzebne obiekty, a w szczególności obiekty odpowiedzialne za komunikację z serwisem sieciowym, wołanie metod oraz automatykę interfejsu. W metodzie uruchamiającej midlet pobierana jest definicja interfejsu i tworzony jest panel czołowy aplikacji oraz obiekty zdarzeń, natomiast w metodzie nasłuchiwacza zdarzeń wołana jest tylko odpowiednia metoda ich obsługi. Przykładowy wygląd panelu przedstawiono na rysunku 1.



Rys. 1. Wygenerowany panel czołowy
Fig. 1. Generated front panel

5. Podsumowanie

Przedstawiono koncepcję budowy aplikacji mobilnego klienta rozproszonego systemu pomiarowego. Przedstawiono projekt systemu pomiarowego, który udostępnia swoje usługi jako serwis sieciowy, a więc umożliwia dostęp do świadczonych usług w sposób uniwersalny i niezależny od platformy klienta. Dodatkowo, oprócz dostępu do właściwych usług stricte „pomiarowych", serwis udostępnia także informacje opisujące graficzny interfejs użytkownika przeznaczony do jego obsługi. W referacie opisano aplikację klienta zrealizowaną na platformie J2ME i przeznaczoną dla przenośnych terminali MDA, która na podstawie specyfikacji GUI udostępnianej przez serwis sieciowy potrafi zbudować odpowiedni interfejs użytkownika, dający wprost dostęp do świadczonych przez ten serwis usług. Podejście to pozwala na zaprojektowanie interfejsu użytkownika od razu w fazie projektowania jądra systemu pomiarowego i znacznie upraszcza dostęp do funkcji sterujących systemem z terminali wyposażonych w zaprojektowane oprogramowanie (generator GUI), a jednoczenie nie ogranicza dostępu do systemu pomiarowego ze standardowych klientów zgodnych ze specyfikacją Serwisów Sieciowych.

Praca naukowa finansowana ze środków Komitetu Badań Naukowych w latach 2004-2006 jako projekt badawczy.

6. Literatura

[1] W. Winiecki, P. Bobiński: Wykorzystanie terminali PDA w bezprzewodowych systemach pomiarowych, Mat. VII Szkoły-Konferencji: Metrologia Wspomagana Komputerowo: MWK 2005, Waplewo, 17-20 Maja 2005.
[2] Web Services Architecture, W3C Working Group Note, February 2004.
[3] Mobile Information Device Profile for Java 2 Micro Edition Version 2.0, JSR 118 Expert Group, Java Community Process, Sun Microsystems, Inc., November 2002.
[4] Connected Limited Device Configuration Specification Version 1.1, Java 2 Platform, Micro Edition (J2ME), Sun Microsystems, Inc., March 2003.
[5] Jon Ellis & Mark Young: J2ME Web Services 1.0, Final Draft, Sun Microsystems, Inc., October 2003.
[6] Java 2 Platform, Micro Edition (J2ME) Web Services, A Technical White Paper, Sun Microsystems, Inc., July 2004.

Artykuł recenzowany