Java Bridge Module dan Java Component API untuk SID Simulator

Muhammad Hasrul Ma'ruf [a] ,dan Nurhayati [b]

[a), b)]*Dosen Program Studi Teknik Informatika*
Fakultas Sains dan Teknologi
Universitas Islam Negeri Syarif Hidayatullah Jakarta

e-mail : hasrulwho@gmail.com, nurhayatibuslim@gmail.com,

ABSTRACT

Simulation tools help creating a low cost and efficient development of embedded system. SID is an open source simulator software that consists library of components for modelling hardware and software components. A component can be written in C/C++ and Tcl/Tk. Currently, the SID simulation toolkit only provides support for C++ and Tcl/Tk. Tcl/Tk is used to create GUI-based components. However, we have observed that Tcl/Tk components causing slow simulation response. Developing GUI using Tcl/Tk is also inflexible. Thus it is not proper for developing the cutting-edge products with rich graphics. In this work, we introduced the idea of Java as an alternative platform for developing components in SID. We suggest two design approaches for Java Bridge module for SID. One is the approach based on socket communication, and the other is based on JNI. SID API for Java component development is also proposed to ensure the compatibility and simplicity of SID components in Java.

Key-Words: - *Embedded system, Simulator, SIDd, Bridge component, Java, Software*

1. Pendahuluan

Saat ini Embedded system semakin kaya fitur. Perangkat keras dan perangkat lunak dalam embedded system telah terjamin dalam pengembangannya sebelum di pasarkan. Waktu dalam memasarkan merupakan sebagai salah satu issue menarik sebagai tantangan pada produsen peralatan elektronik yang agar tetap kompetitif. Simulasi membantu pengembang sistem untuk meningkatkan waktu pemasaran, sebagai kompleksitas sistem dan tuntutan yang lebih ketat pada peningkatan kualitas sistem secara keseluruhan.

Simulator SID terdiri dari sebuah mesin yang menghubungkan beban dan simulasi komponen-komponen pada embedded system. Virtual Development Environment for Embedded Software (VDEES) adalah lingkungan virtual yang menyediakan alat konfigurasi, kode editor untuk menulis komponen simulasi, alat untuk membangun citra biner, debugger, dan monitor sistem untuk penyelidikan dari target virtual. VDEES didasarkan pada SID dan menggunakan Eclipse plug-in sebagai dasarnya.

SID menyediakan sistem monitor yang built-in ditulis dalam Tcl/Tk untuk memantau jalannya simulasi. Sistem memonitor daftar komponen yang aktif dalam platform virtual, menampilkan spesifikasi komponen atribut seperti pin, register, dll. Sejak VDEES didasarkan pada platform Eclipse, monitor sistem juga harus dilakukan untuk Eclipse plug-in, yang harus ditulis di Java. Namun, SID tidak dapat mendukung komponen yang ditulis di Java secara langsung tanpa komponen java bridge.

Di awali oleh kemungkinan mengintegrasikan sistem monitor untuk VDEES, kami mengusulkan ide menggunakan Java sebagai platform alternatif untuk mengembangkan komponen dalam SID. Selain motivasi pertama, saat SID komponen hanya dapat ditulis dalam C/C++ dan Tcl / Tk. Pengujian kami menunjukkan bahwa SID komponen yang menggunakan Tcl/Tk, berjalan sangat lambat. Untuk setiap input sentuhan memerlukan beberapa detik untuk evaluasi dan 1 detik untuk refresh tampilan. Selain itu, GUI di Tcl / Tk dianggap tidak layak untuk dikembangkan atau user-friendly, dibandingkan dengan bahasa lain seperti Java atau .NET.

Dalam pekerjaan sebelumnya, kami secara singkat memaparkan ide utama dari Java Bridge dan sistem prototipe yang menggunakan socket. Dalam tulisan ini, kami menyajikan gagasan Java Bridge dalam dua alternative yakni menggunakan socket dan JNI. Di sini, kami juga mengusulkan bahwa setiap komponen baru ditulis di Java harus mengikuti API tertentu, untuk memastikan kompatibilitas komponen baru dengan SID. Seperti Java mendukung pola desain Berorientasi Obyek, API juga akan dapat membantu pengembangan komponen SID di Java menjadi lebih cepat dan lebih mudah.

2. Penelitian yang berelasi

Penelitian kami didasarkan pada SID Simulator dan VDEES. Tujuan kami adalah untuk menciptakan alternatif untuk Tcl / Tk jembatan, yang Java. Bab ini akan memberikan penjelasan singkat tentang beberapa dasar dari pekerjaan ini.

2.1 SID Simulator

SID adalah suatu kerangka kerja untuk membangun sistem simulasi komputer dan SID dibuat untuk debugging, testing, dan verifikasi embedded sofware. Secara khusus, simulasi terdiri dari kumpulan pasangan komponen yang di perluas. Simulasi dapat berkisar dari instruksi CPU untuk multi-prosesorpada embeded system. SID memiliki fitur berikut:

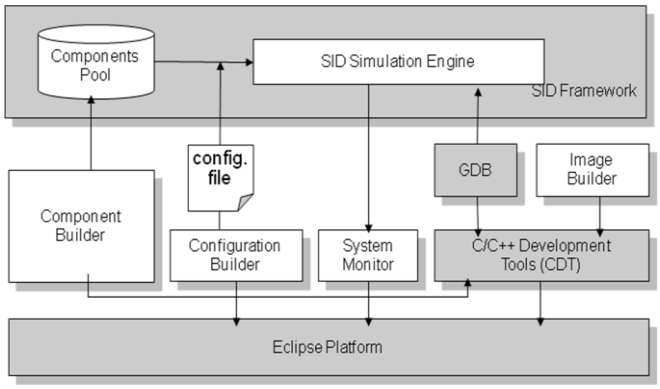
- Open source framework untuk membangun sistem simualsi komputer.
- Pertumbuhan library dari komponen-komponen untuk modeling bagian-bagian pada perangkat keras dan perangkat lunak, instrumentasi, kontrol, and antar muka eksternal.
- Dukungan GDB debugger
- Virtual Target Platform
- Embedded system software testing dan verification.

SID mendefinisikan antarmuka komponen kecil yang berfungsi untuk membungkus erat mereka. Komponen dapat ditulis dalam C++, C, Tcl atau bahasa lainnya yang termasuk dalam lingkungan API. Namun SID simulation toolkit hanya menyediakan dukungan untuk C++ and Tcl/Tk. SID didasarkan pada C++, karena C++ adalah bahasa utama dan untuk penambahan bahasa dibutuhkan komponen khusus berupa jembatan (bridge). Saat ini hanya Tcl/Tk bridge yang tersedia.

Biasanya, komponen-komponen secara terpisah disusun dan dikemas ke dalam library bersama. SID model komponen komunikasi melalui bus dan pin. SID juga mendukung komunikasi tingkat yang lebih tinggi dengan menggunakan atribut dan relasi. Selama simulasi start-up, komponen instansiasi, saling berhubungan, dan dikonfigurasi yang diperlukan untuk mewakili beberapa sistem tertentu. Semua konfigurasi ini ditulis dalam satu file konfigurasi, dan diperlukan sebelum runtime SID.

2.2 VDEES

VDEES adalah sebuah lingkungan pengembangan yang terintegrasi yang kami dibangun di atas *Eclipse* dan SID, lihat Gambar. 1. Menggunakan *Eclipse* dan *plug-in* yang bekerja sebagai mesin untuk mempermudah pembangunan SID. Sejauh ini, ada empat *plugin* yang ada di VDEES yakni VDEES *Binary Image Builder*, VDEES *Configuration Builder*, and VDEES *Custom Component Wizard*, and VDEES monitor. Kami menciptakan komponen Java bridge untuk SID menggunakan VDEES.



Gambar. 1 Arsitektur VDEES

2.3 C++ to Java Communication

SID engine dan native SID components ditulis dalam C++. Oleh karena itu kita membutuhkan sebuah protokol yang dapat memungkinkan komunikasi dari C++ ke Java, dan sebaliknya, jika kita ingin membuat komponen SID bekerja di dalam Java.

Sejumlah pendekatan alternatif juga memungkinkan aplikasi Java untuk beroperasi dengan kode yang ditulis dalam bahasa lain seperti C++. Java Native Interface (JNI) adalah suatu framework pemrograman yang memungkinkan kode Java berjalan dalam Java Virtual Machine (JVM) untuk memanggil dan disebut oleh aplikasi asli (program khusus untuk hardware dan platform sistem operasi) libraries yang ditulis dalam bahasa lain, seperti C, C++ dan assembly.

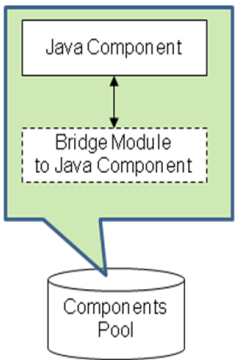
Selain JNI, aplikasi Java dapat terhubung ke database melalui JDBC™ API. Aplikasi Java juga dapat mengambil keuntungan dari teknologi objek terdistribusi seperti Java IDL API. Alternatif lain adalah komunikasi melalui koneksi soket TCP/IP atau melalui mekanisme lain inter-process communication (IPC).

3. Java Bridge Module

Untuk mendapatkan dukungan dengan bagian lain dari SID arsitektur, desain pada Java harus didasarkan pada arsitektur SID saat ini.

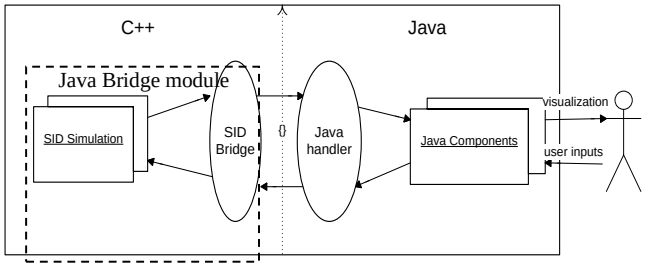
Berdasarkan pada hal ini, kita menyimpulkan bahwa Java Bridge untuk mematuhi persyaratan:

- Java Bridge Module adalah native (C++) SID komponen. Modul bridge module akan berlokasi di pusat komponen sebagai library yang di gunakan bersama dan di muat oleh SID engine pada runtime, lihat Gambar 2.
- Hal ini memungkinkan C++ to Java dengan komunikasi dua arah.
- Java Bridge module harus menjadi transparan untuk memanggil SID. Dengan demikian, Java Komponen di SID dapat diperlakukan seperti SID komponen lainnya.
- Mampu mengatur status Java component dan kemudian menggunakan kembali.



Gambar. 2 Java Bridge and Component location

Modul bridge terdiri atas C++ SID komponen dan bahasa asing. Pertama, SID engine memanggil bridge modul, dan kemudian daftar untuk komponen asing lainnya yang tersedia pada bridge, yaitu di Tcl/Tk atau Java Bridge. Kemudian daftar komponen-komponen yang tersedia dari bridge. Engine dapat melakukan berbagai pangilan ke/dari komponen asing melalui interpretasi dan komunikasi dengan berbagai cara. Di bawah ini, Gambar 3 adalah sebuah contoh bagaimana SID, bridge, komponen asing , dan user berinteraksi satu sama lain.



Gambar. 3 SID, Bridge, Java Component, and user interactions

3.1 Java Bridge Component using Socket

Dalam percobaan pertama kami, JNI itu tidak dapat digunakan dalam C++ library bersama. Oleh karena itu, tidak dapat digunakan sebagai mekanisme komunikasi bridge. Jadi kami memilih koneksi soket TCP/IP sebagai metode komunikasi antara C++ dan Java di SID-Java Bridge. Menggunakan soket memungkinkan kita untuk menganggap komponen Java yang akan terletak di host manapun.

Pada awalnya, kami menempatkan di kedua sisi single-server-multiple-client arsitektur, C++ dan Java, untuk mendukung komunikasi asynchronous. Secara khusus, kami menerapkan single TCP server pada Java dan C++ yang dapat menangani beberapa permintaan (panggilan prosedur) dan menciptakan beberapa penangan untuk setiap permintaan. Kemudian ternyata bahwa penggunaan thread memperlambat komunikasi untuk SID, bahkan lebih lambat dari Tcl/Tk. Meskipun demikian, sudah pasti cepat hanya pada pengiriman komunikasi, tidak pada pengembalian lagi. Sayangnya, tidak ada skenario pengembalian SID adalah hanya terbatas untuk komunikasi pin. Saat ini, arsitektur socket yang kami diusulkan adalah single threaded pengiriman pesan dan penanganan di kedua sisi.

Kami mengusulkan XML sebagai format pesan dalam panggilan dan pengembalian. Meskipun XML dapat membuat beberapa dari jenis kinerja, untuk alasan pembacaan dan juga portabilitas, kami memilih sintaks standar ini. Alasan lainnya adalah bahwa prosesor XML tersedia untuk Java dan C++.

SID Java Bridge tidak hanya membuat Java component, tetapi juga harus mampu berkomunikasi dengan komponen-komponen yang dibuat sesudahnya. Oleh karena itu penting untuk membuat bridge menjaga pointer ke Java object (komponen). Di dalam solusi berbasis socket, kami menggunakan Java object ID, memelihara id dan objek dalam hash table dan kemudian mengirim id ke sisi C++ , lihat Gambar.4. Kemudian, jika SID (C++) dibutuhkan untuk berkomunikasi dengan komponen, ia akan menggunakan id objek sebagai kunci untuk berinteraksi dengan objek yang tepat.

```
Hashtable gHash = new Hashtable();
String objId=Integer.toHexString(
       System.identityHashCode(component));
gHash.put(objId, component);
```

Gambar. 4 Object pointer's mechanism in socket-based Bridge

Di arah lain, kami menggunakan nama pin, bus, dll sebagai kunci untuk memanggil dari komponen Java ke komponen lainnya. Kami menggunakan tabel bijection lookup yang dapat dibuat, jika tidak ada, dari pointer string, atau mengambil pointer,

jika ada, dari pin atau bus di SID. Tabel lookup juga dapat digunakan dalam arah sebaliknya, mendapatkan nama dari pointer.

3.2 Java Bridge Component using JNI

Berbeda dengan soket berbasis komunikasi, JNI memungkinkan dengan dua cara yakni C + + dan Java untuk menjaga pointer dan melakukan panggilan ke objek asing. JNI mendukung penciptaan objek remote. Misalnya C++ dapat membuat objek Java, bersama dengan pointer objeknya. Kemudian, C++ dapat memanggil metode remote objek di Java dan sebaliknya.

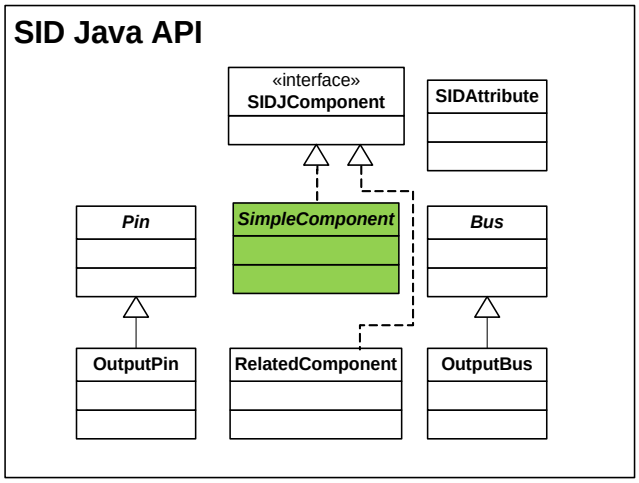
Untuk komunikasi C++ ke Java, kami menggunakan GIWS. GIWS adalah perangkat lunak gratis yang sangat menyederhanakan panggilan metode Java/objek dari C/C++. Ini memiliki kode generator lurus ke depan yang memungkinkan kita menggunakan XML sederhana untuk membuat C++ deklarasi class dan implementasi yang mengelola panggilan JNI, dan konversi data/objek. Setelah class C++ untuk setiap class Java yang ingin kita menggunakan di C + +, kita bisa membuat atau memanggil objek Java dalam cara yang sama seperti yang kita sebut di Java.

Untuk Java ke C++ komunikasi, kita menggunakan fungsi registerNatives. Jika GIWS menggunakan XML untuk membuat penafsiran antara bahasa yang berbeda, fungsi registerNatives mengambil nama metode dalam C++ dan Java bersama dengan metode tanda tangan sebagai parameternya. Fungsi ini juga memungkinkan kita untuk mendaftarkan batch native fungsi yang dapat dipanggil dari Java.

3.3 SID Java API

Sebuah Application Programming Interface (API) adalah seperangkat aturan tertentu dan spesifikasi bahwa sebuah program perangkat lunak dapat mengikuti untuk mengakses dan memanfaatkan layanan dan sumber daya yang disediakan oleh program lain perangkat lunak tertentu yang menerapkan API. Bagian ini menjelaskan tentang API yang digunakan di Java dalam rangka menciptakan komponen SID yang tepat. Sehingga komponen dapat disebut sebagai atau panggilan layanan SID.

Untuk membuat API, kami menciptakan class abstrak dan interface, lihat Gambar. 5. Tujuan dari class-class dan interface adalah untuk memastikan bahwa setiap komponen baru yang diciptakan di Java adalah mengikuti aturan SID yang ketat dan akan selalu kompatibel dengan panggilan SID, baik itu di pin, bus, atribut, atau hubungan dengan pemanggilannya.



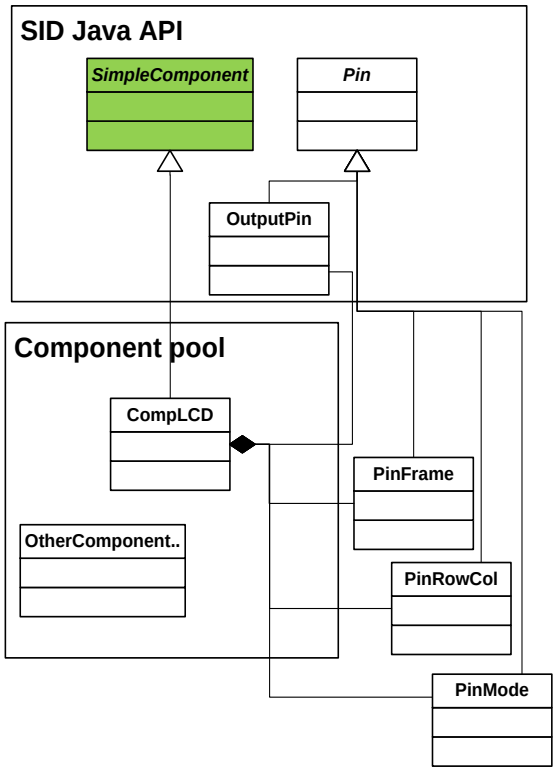
5 Java API class diagram

SIDJComponent adalah sebuah antarmuka yang memastikan semua fungsi komunikasi SID diimplementasikan dalam Komponen Java. Ini berisi template fungsi untuk pin, bus, atribut, dan relationship. Pin dan bus class abstrak yang menyerupai pin dan bus di SID. Pin memiliki metode abstrak yang akan mendorong nilai integer dari pin ke yang lain. Bus telah memiliki abstrak untuk menulis dan membaca metode yang memungkinkan bus untuk menulis dan membaca data dari alamat memori tertentu. OutputPin dan OutputBus digunakan untuk mendrive, menulis, atau membaca data ke / dari Tcl/Tk atau C++ komponen. SIDAttribute adalah membiarkan komponen memiliki atribut dengan nama, nilai, dan kategori.

SimpleComponent adalah class abstrak yang memiliki default implementasi untuk melampirkan pin, bus, dan komunikasi atribut. Hal ini dimaksudkan untuk membantu pengembangan komponen SID lebih mudah dan lebih cepat. Sementara itu, RelatedComponent adalah class abstrak yang memberikan abstraksi untuk komponen terkait yang digunakan ketika kita menggunakan relasi dan tidak berelasi dalam komunikasi SID. SID yang sama Java API yang digunakan pada bridge komunikasi socket-based dan JNI-based SID.

Sebuah komponen Java akan memperluas class SimpleComponent. Setelah diperluas, komponen tidak hanya memastikan bahwa ia memiliki fungsi seluruh SID komponen, tetapi juga memiliki beberapa dari mereka yang sudah diimplementasikan. Beberapa yang sudah diimplementaskan adalah untuk mendaftar, untuk menemukan, untuk menghubungkan dan memutuskan pin, bus, dan atribut pada jalannya. Komponen lengkap Java mungkin diperlukan untuk

melaksanakan jika perlu salah satu Bus atau Pin. Kelengkapan Bus atau Pin diimplementasikan di SID Java komponen akan memiliki petunjuk, apa yang harus dilakukan ketika sebuah pin atau bus mengirim atau mengambil data. Di bawah ini adalah contoh kelas diagram dari komponen lenkap Java bersama dengan pin, lihat Gambar. 6.



Gambar. 6 Example component using SID Java API

4. Kesimpulan

Dalam penelitian ini kami mengusulkan Java Bridge sebagai alternative untuk Tcl/Tk Bridge pada SID saat ini. Bridge antara SID and Java component akan di bangun menggunakan JNI dan socket. Kami juga mengusulkan Java API untuk memastikan kompatibilitas dari setiap SID Component di Java. API juga di maksudkan untuk membuat pengembangan komponen-komponen SID di Java lebih mudah dan lebih cepat. Keduanya dimaksudkan untuk memberikan komponen GUI yang lebih fleksibel dengan juga kinerja yang lebih baik daripada tampilan GUI dari pada Tcl/Tk komponen

Referensi

[1] SID. http://sourceware.org/sid/

[2] Hadipurnawan Satria, Baatarbileg Altangerel, Jin Baek Kwon, Jeongbae Lee, "Configurable Virtual Platform Environment using SID Simulator and Eclipse," In *Proc. of Software Technologies for Embedded and Ubiquitous Systems*, SEUS 2007, pp.394-398.

[3] Febiansyah Hidayat, Hadipurnawan Satria, Jin Baek Kwon, Software Verification of a Virtual Development Environment for Embedded Software, In Proceedings of the 9th *WSEAS international conference on Software engineering, parallel and distributed systems*, 2010.

[4] Hasrul Ma'ruf, Jin Baek Kwon, Developing a Bridge Module to Java Component for SID Simulator, In Proc of *The 34Th Korea Information Processing Society (KIPS) Fall Conference*, 2010

[5] VDEES. http://april.sunmoon.ac.kr/vdees

[6] The Java Native Interface Programmer's Guide and Specification, Sun Microsystems, chap. 1.

[7] GIWS. http://forge.scilab.org/index.php/p/giws/