Jurnal Teknologi, 48(D) Jun 2008: 59–74

MENGUKUR PARAS KESAHSAHIHAN SPESIFIKASI Z DALAM PENILAIAN SPESIFIKASI KEPERLUAN PELAJAR MELALUI PENDEKATAN PELAKSANAAN BERSIMBOL

ZARINA SHUKUR[1] & AHMAD RIZAL MOHD YUSOF[2]

Abstrak. Kertas ini mengesyorkan kaedah pengukuran paras kesahsahihan spesifikasi formal perisian. Pengesahsahihan dilakukan untuk memastikan spesifikasi menyatakan kelakuan program dengan tepat dan memenuhi keperluan pengguna. Paras kesahsahihan ini ingin diukur khususnya untuk kegunaan guru dalam menilai spesifikasi keperluan yang disediakan oleh pelajar. Pengesahsahihan dilakukan dengan menggunakan teknik pengujian, iaitu spesifikasi yang dihasilkan (oleh pelajar) dilaksanakan secara bersimbol terhadap sejumlah kes ujian yang disediakan secara bermakna dengan bantuan pelanggan (diwakili oleh guru). Seterusnya output ujian dalam bentuk predikat matematik antara pelanggan (guru) dan penspesifikasi (pelajar) dibandingkan. Hasil daripada perbandingan ini adalah suatu nilai numerik yang mencerminkan paras kesahsahihan spesifikasi berkaitan.

Kata kunci: Spesifikasi formal, pengesahsahihan spesifikasi, perlaksanaan bersimbol, paras kesahsahihan, ukuran kesahsahihan spesifikasi

Abstract. This paper proposes a method to measure the level of formal software specification validity. The validation of a specification is done in order to ensure that the specification correctly specify the program behaviour and fulfill the user requirements. We wish to measure the level of validity for teacher purposes in evaluating requirement specification which is written by students. The validation is done by using testing technique that is; the specification (which is prepared by the student) will be symbolically executed against several test cases which have been prepared earlier (by the teacher). Then, the output from the test, which is in the form of mathematical predicates will be compared against the expected answer. The result from the comparison is a numeric value which represents the level of the specification validity.

Keywords: Formal specification, specification validation, symbolic execution, validity level, specification validation measure

1.0 PENGENALAN

Analisis keperluan pengguna merupakan titik permulaan untuk suatu projek perisian. Ia mengandungi dokumen yang menyatakan keperluan tak formal ciri-ciri perisian

[1&2] Jabatan Sains Komputer, Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor. Tel: 03-89216720 Faks: 03-89216184
Email: zs@ftsm.ukm.my

yang akan dibangunkan [1]. Dokumen ini akan diproses dan dianalisis untuk menghasilkan dokumen spesifikasi keperluan. Penyataan dalam dokumen spesifikasi keperluan mestilah tepat, jelas, konsisten dan tidak taksa. Ini penting bagi menghasilkan perisian yang dapat memenuhi keperluan pengguna [2]. Spesifikasi yang tak formal adalah kabur dan tak jelas [3]. Ia mungkin difahami oleh pelanggan atau pembangun perisian tetapi ia mewujudkan interpretasi individu yang berbeza [4].

Untuk mengatasi masalah ini, kebanyakan saintis komputer menggunakan matematik untuk menspesifikkan dan membangunkan perisian [1-5]. Spesifikasi formal yang selesai dimodelkan dan didokumenkan perlu disahsahihkan. Ini penting untuk memastikan spesifikasi formal dapat menyatakan kelakuan program dengan tepat dan memenuhi keperluan pengguna [6]. Di antara teknik yang digunakan untuk mengesahsahihkan spesifikasi formal ialah teknik ulasan formal atau *formal technical review, viewpoint resolution technique,* teknik pembuktian formal [7] serta teknik pengujian [1]. Dua teknik yang pertama ini adalah adalah tidak berprosedur manakala teknik pembuktian formal adalah yang terbaik, namun sangat remeh. Teknik pengujian pula adalah berprosedur dan praktikal.

Pengujian untuk mengesahsahihkan spesifikasi formal bermaksud melaksanakan spesifikasi terhadap data ujian [1]. Idea ini telah dicadangkan oleh Kemmerer [8]. Namun begitu, kebanyakkan notasi spesifikasi formal adalah tidak berprosedur dan tidak boleh dilaksanakan secara terus [3]. Suatu cara yang digunakan untuk mentransformasikan spesifikasi formal kepada bentuk yang berprosedur supaya ia boleh dilaksanakan terhadap data ujian adalah melalui teknik animasi [1].

Terdapat beberapa usaha ke atas pengesahsahihan dan pengukuran spesifikasi formal secara berkomputer. Abdullah & Zarina [9] mengesyorkan pengesahsahihan dilakukan dengan menganimasi spesifikasi berdasarkan input ujian yang terpilih dan membandingkan dengan output ujian yang dijangka. Spesifikasi dilaksanakan dengan terlebih dahulu menukarnya ke dalam kod Prolog. Walaupun terdapat teknik untuk mengesahsahihkan spesifikasi, tidak terdapat perbincangan berkenaan pengukuran paras kesahsahihan suatu spesifikasi dalam kajian mereka. Namun begitu, pengukuran spesifikasi formal dari aspek kompleksiti ada disyorkan oleh Zarina [10] dan Wu & Yi [11], iaitu dengan menggunakan metrik perisian sedia ada yang diubah suai. Manakala Huang & Lai [12] mengesyorkan pengukuran kebolehsenggaraan protokol komunikasi dengan mendapatkan metrik kebolehsenggaraan daripada spesifikasi formal protokol tersebut.

Kertas ini membincangkan berkenaan pengukuran kesahsahihan bagi suatu spesifikasi formal Z. Pengesahsahihan dilakukan dengan menggunakan teknik pelaksanaan bersimbol, manakala pengukuran dilakukan dengan membandingkan output hasilan dengan output jangkaan. Salah satu aplikasi kajian ini adalah untuk digunakan dalam persekitaran pendidikan, iaitu pemarkahan spesifikasi pelajar secara berkomputer. Pemarkahan adalah satu proses yang memerlukan elemen konsistensi

dan konsentrasi. Sehubungan itu, dengan adanya pendekatan kepada penilaian yang boleh diautomasikan terhadap tugasan berasaskan spesifikasi formal Z, ia dapat membantu pengajar dalam menilai tugasan tersebut. Berbanding dengan aplikasi lain, skop dan saiz bagi spesifikasi formal dalam persekitaran pendidikan lebih kecil berbanding yang sebenar.

2.0 PELAKSANAAN BERSIMBOL DALAM PENGESAHSAHIHAN SPESIFIKASI FORMAL

Balzer *et al.* [13] telah mengimplemen pelaksanaan bersimbol untuk mengesahsahihkan spesifikasi *Generating Instructional Text* (GIST). Spesifikasi GIST dinyatakan melalui entiti dan hubungan. Sistem yang telah dibangunkan oleh Balzer *et al.* ini menganggap spesifikasi GIST sebagai set aksiom dan turutan logik tertib pertama. Ia akan menjana suatu teorem mudah dari aksiom-aksiom dalam spesifikasi GIST dengan menggunakan *Forward Inference Engine (*FIE). Teorem kemudiannya disemak dan dilaksanakan secara bersimbol oleh sistem sebelum dipaparkan kepada pengguna. Sistem ini disokong oleh parafrasa yang menterjemahkan spesifikasi GIST yang telah disahsahihkan ke bahasa Inggeris. Swartout [14] menyatakan penterjemahan ini bertujuan untuk memudahkan pengguna melihat keputusan pengesahsahihan spesifikasi GIST secara pelaksanaan bersimbol.

Seterusnya Kemmerer [8] telah membangunkan suatu sistem pelaksanaan bersimbol untuk membandingkannya dengan sistem prototaip bagi mengesahsahihkan spesifikasi *Ina Jo*. Spesifikasi *Ina Jo* adalah tidak berprosedur dan menggunakan predikat kalkulus tertib pertama bagi memodelkan sistem. Nilai-nilai pemboleh ubah keadaan spesifikasi *Ina Jo* dinyatakan sebagai peralihan keadaan. Kemmerer menganggap keperluan fungsian sistem yang sah perlu mematuhi spesifikasi *Ina Jo* dan maklumat yang terdapat dalam keperluan ini digunakan untuk membina kes ujian. Sistem pelaksanaan bersimbol yang dibangunkan oleh Kemmerer akan menjanakan kes ujian ke atas spesifikasi *Ina Jo*. Spesifikasi *Ina Jo* yang telah dijanakan dengan kes ujian akan disemak dahulu sebelum dilaksanakan secara bersimbol. Penyemakan ini bertujuan untuk memastikan sintaks dan jenis spesifikasi *Ina Jo* bebas dari ralat. Kemudian sistem akan melaksanakan spesifikasi *Ina Jo* secara bersimbol. Keputusan pengesahsahihan spesifikasi *Ina Jo* secara pelaksanaan bersimbol ini akan dipaparkan untuk menunjukkan proses pengesahsahihan berjaya atau gagal dijalankan. Kemmerer turut membandingkan hasil dari pengesahsahihan spesifikasi *Ina Jo* yang dijalankan melalui teknik pelaksanaan bersimbol ini dengan hasil pengesahsahihan spesifikasi *Ina Jo* melalui sistem prototaip. Kemmerer mendapati sistem yang dibangunkan untuk mengesahsahihkan spesifikasi *Ina Jo* melalui teknik pelaksanaan bersimbol dapat menghasilkan spesifikasi *Ina Jo* yang tepat dan memenuhi keperluan pengguna.

Kneuper [15] juga telah membangunkan satu sistem untuk mengesahsahihkan spesifikasi formal *Vienna Development Method* (VDM) dengan teknik pelaksanaan

bersimbol. Sistem ini dikenali sebagai *Symbolic Execution* (SYMBEX). Sistem ini dilarikan pada persekitaran Windows. Motivasi untuk membangunkan SYMBEX ini berpunca dari masalah yang dinyatakan oleh Kneuper untuk mengesahsahihkan spesifikasi VDM melalui teknik pelaksanaan bersimbol. Masalah yang dihadapi adalah menangani spesifikasi VDM yang tersirat serta pengenerikkan dalam VDM. Kedua-dua masalah ini dibincangkan untuk menjelaskan fungsi pembangunan SYMBEX dalam mengesahsahihkan spesifikasi VDM melalui pelaksanaan bersimbol. Masalah dalam menangani spesifikasi VDM yang tersirat ini diselesaikan dengan memperkenalkan nilai-nilai diskripsi. Masalah pengenerikkan pula diselesaikan dengan mendefinisikan pelaksanaan bersimbol dalam semantik VDM. Proses pengesahsahihan spesifikasi VDM melalui SYMBEX ini melibatkan input yang dimasukkan oleh pengguna ke atas skema-skema yang terdapat dalam spesifikasi VDM. SYMBEX akan membaca spesifikasi VDM dahulu. Kemudian input yang dimasukkan oleh pengguna akan dilaksanakan secara bersimbol. Seterusnya SYMBEX akan memaparkan keputusan yang dihasilkan. Jika SYMBEX memaparkan "*true*" ini menunjukkan skema dalam spesifikasi VDM itu telah disahsahihkan. Jika SYMBEX memaparkan "*false,*" skema yang diuji itu gagal disahsahihkan. SYMBEX telah diuji pada spesifikasi sistem transaksi perbankan. Dalam ujian yang telah dijalankan ini, Kneuper [15] menjelaskan SYMBEX berupaya mengesahsahihkan spesifikasi sistem transaksi perbankan.

Manakala Douglas and Kemmerer [16] membangunkan alat pelaksanaan bersimbol bagi menguji bahasa spesifikasi formal yang dinamakan Aslan. Aslan adalah bahasa spesifikasi berasaskan keadaan yang dibina atas kalkulus predikat tertib pertama. Alat yang dibina itu dinamakan Aslantest. Aslantest menganimasi spesifikasi Aslan dan membolehkan pengguna melaksanakan spesifikasi terhadap sejumlah kes ujian atau secara bersimbol.

Kajian-kajian yang telah dibincangkan di atas menunjukkan proses pengesah-sahihan spesifikasi formal melalui teknik pelaksanaan bersimbol dapat mencapai objektif untuk membuktikan suatu spesifikasi yang tepat dan memenuhi keperluan. Suatu ciri persamaan yang dapat dikenal pasti dalam kajian yang telah dijalankan oleh Balzer *et al*. [13], Kemmerer [8], Kneuper [15] dan Douglas and Kemmerer [16] ialah sistem yang dibangunkan memerlukan dua input spesifikasi yang ingin disemak atau dilaksanakan secara bersimbol, iaitu teorem [13] atau kes ujian [8][15-16] untuk menjalankan proses pengesahsahihan. Manakala perbezaan yang diperhatikan adalah teorem atau kes ujian yang digunakan oleh Balzer *et al*. [13] dan Kemmerer [8] dijanakan oleh sistem manakala kes ujian untuk SYMBEX dan Aslantest disediakan oleh pengguna. Melalui sistem yang telah dibangunkan, keempat-empat pengkaji ini telah menunjukkan bahawa pengesahsahihan spesifikasi formal boleh dilakukan secara pelaksanaan bersimbol.

Kebanyakan kajian selepas tahun 2000, menumpu kepada penggunaan pelaksanaan bersimbol terhadap spesifikasi formal bagi menghasilkan kes ujian,

dan seterusnya menggunakan kes ujian tersebut terhadap spesifikasi yang telah diperhalusi yang mana antaranya ialah aturcara yang berkaitan. Sebagai contoh, Manoranjan *et al*. [17] membincangkan pelaksanaan bersimbol ke atas model yang ditulis dalam bahasa spesifikasi formal berorientasi model bagi tujuan menghasilkan sejumlah kes ujian. Dalam kajian mereka, kes ujian yang terhasil digunakan bagi memperoleh satu aturcara Java, yang mana aturcara tersebut digunakan pula pemacu ujian. Le Gall *et* al. [18] pula menggunakan pelaksanaan bersimbol bagi menyemak sama ada spesifikasi yang diperhalusi setara dengan spesifikasi asal.

3.0 MENGESAHSAHIH SPESIFIKASI Z SECARA PELAKSANAAN BERSIMBOL

Berdasarkan kepada kajian-kajian yang telah dijalankan oleh King [19], Balzer *et al*. [13], Kemmerer [8] dan Kneuper [15], suatu metodologi yang bersesuaian dalam mengesahsahihkan spesifikasi Z melalui teknik pelaksanaan bersimbol boleh dilakukan seperti berikut:

(1) Menyediakan spesifikasi Z yang ingin disahsahih
(2) Menyediakan satu set kes ujian (input ujian dan output jangkaan) dengan mengekalkan prinsip pengesahsahihan
(3) Melaksanakan spesifikasi Z terhadap kes ujian secara bersimbol
(4) Membandingkan hasil pelaksanaan dengan output jangkaan

3.1 Penyediaan Kes Ujian

Singh *et al*. [20] menjelaskan kes ujian merupakan suatu input untuk proses pengujian yang mengandungi satu set data. Sommerville [21] juga menakrifkan kes ujian sebagai input dan output kepada proses pengujian (jangkaan). Dalam kajian ini, kes ujian merujuk kepada maklumat keadaan data yang disediakan oleh pelanggan [10]. Kes ujian terdiri daripada pemboleh ubah keadaan, pemboleh ubah input dan pemboleh ubah output yang disekutukan dengan suatu nilai untuk menguji skema dalam spesifikasi Z.

Pemboleh ubah keadaan disytihar di bahagian skema keadaan suatu spesifikasi sistem. Pemboleh ubah yang diakhiri dengan tanda soal (?) dikenali sebagai pemboleh ubah input. Manakala pemboleh ubah yang diakhiri dengan tanda seruan (!) dikenali sebagai pemboleh ubah output. Pemilihan kes ujian untuk mengesahsahihkan spesifikasi Z secara bersimbol ini adalah berdasarkan kepada tiga cara, iaitu [2]:

(i) Pembuktian Teorem Keadaan Awal

Teorem keadaan awal merupakan predikat yang memuaskan keadaan awal dan invarian keadaan. Skema *TelephoneBook* dan Skema *InitTelephoneBook* dalam

spesifikasi Sistem Buku Telefon digunakan untuk menjelaskan penghasilan kes ujian yang mewakili teorem keadaan awal. Skema *TelephoneBook* merupakan skema keadaaan yang menggambarkan kelakuan sistem secara keseluruhan. Ia terdiri daripada pemboleh ubah keadaan, *known* dan *telephone*. Predikat dalam skema *TelephoneBook*, iaitu *known* = *dom telephone* juga dikenali sebagai invarian keadaan. Ini kerana ia menyatakan ciri-ciri yang perlu ditangani dan ia juga sentiasa benar [22].

```
┌─ TelephoneBook_____________________
├→ known : Π NAME
├→ telephone : NAME ϕTELEPHONE
├──
├→ known = dom telephone
└____________________________________
```

Manakala skema *InitTelephoneBook* pula menunjukkan bahawa Sistem Buku Telefon bermula dengan dokumen yang kosong sebelum sebarang operasi dijalankan. Pemboleh ubah keadaannya dinyatakan sebagai set kosong.

```
┌─ InitTelephoneBook_________________
├→ TelephoneBook
├──
├→ known = {}
├→ telephone = {}
└____________________________________
```

Oleh kerana skema *InitTelephoneBook* menunjukkan keadaan awal sistem dengan pemboleh ubah keadaannya telah diberikan nilai, maka skema *InitTelephoneBook* ini akan digunakan terus sebagai kes ujian.

(ii) Pembuktian Ciri yang Perlu Ada

Ciri-ciri yang terdapat dalam suatu skema operasi dalam spesifikasi Z juga boleh disahsahihkan. Skema *AddTelephone* digunakan untuk menjelaskan penghasilan kes ujian yang mewakili ciri-ciri dalam suatu skema Z.

```
┌─AddTelephone_______________________
├→ΔTelephoneBook
├→ name?: NAME
├→ phone?: TELEPHONE
├──
└→ name? ™ {}
```

$$\begin{array}{l} \rightarrow known' = known \ Y \ \{name?\} \\ \rightarrow telephone' = telephone \ Y \ \{(name?, phone?)\} \end{array}$$

Skema *AddTelephone* menunjukkan operasi penambahan yang dilakukan ke atas pangkalan data sistem buku telefon. Penggunaan Δ menyatakan pangkalan data sistem akan mengalami perubahan semasa operasi penambahan dilakukan. Simbol ? menunjukkan pemboleh ubah tersebut adalah input kepada operasi penambahan. Pemboleh ubah yang bertanda ′ (perdana) pula menunjukkan keadaan selepas operasi dijalankan. Terdapat 2 input untuk operasi penambahan ini, iaitu *name?* dan *phone?*

Berdasarkan kepada skema *AddTelephone*, suatu kes ujian yang mewakili ciri-ciri dalam skema *AddTelephone* boleh dihasilkan. Satu ciri yang dikenal pasti bagi operasi penambahan ialah

> "Nama bagi pemilik nombor telefon yang akan ditambah ke dalam pangkalan data Sistem Buku Telefon mestilah belum wujud lagi dalam senarai pangkalan data Sistem Buku Telefon. Jika telah wujud, operasi penambahan tidak sepatutnya berlaku."

Input ujian untuk skema *AddTelephone* melibatkan pemboleh ubah keadaan *known* dan *telephone* dan pemboleh ubah input *name?* dan *phone?* Nilai untuk pemboleh ubah keadaan ini boleh dinyatakan sebagai set kosong. *ali* akan digunakan sebagai nilai bagi pemboleh ubah input untuk nama (*name?*) dan *N8782222* sebagai nilai bagi pemboleh ubah input untuk nombor telefon (*phone?*). Kes ujian ini boleh distrukturkan seperti berikut:

Input Ujian:
Pemboleh ubah keadaaan:
known = { }
telephone = { }
Pemboleh ubah input:
name? = ali
phone? = N8782222

Dalam skema *AddTelephone*, tiada pemboleh ubah ouput yang wujud. Oleh itu output ujian dalam skema *AddTelephone* ini hanya terdiri daripada pemboleh ubah keadaan selepas operasi penambahan dijalankan, iaitu *known'* dan *telephone'*.

Output ujian yang dijangkakan jika inputnya seperti di atas adalah *ali* dan *N8782222* berada dalam pangkalan data Sistem Buku Telefon.

Output Ujian:
Pemboleh ubah keadaaan selepas operasi:
known' = {ali}
telephone' = {(ali , N8782222)}

Daripada skema *AddTelephone* ini, satu kes ujian lagi boleh dihasilkan. Kes ujian yang seterusnya akan melakukan operasi penambahan ke atas pangkalan data Sistem Buku Telefon yang telah wujud dengan *ali* sebagai pemilik kepada nombor telefon *N8782222*. Kes ujian ini boleh distrukturkan seperti berikut:

Input Ujian:
Pemboleh ubah keadaan:
known = {ali}
telephone = {(ali , N8782222)}
Pemboleh ubah input:
name? = ali
phone? = N8782211

Output ujian tidak akan disediakan kerana input ujian yang digunakan sepatutnya akan menggagalkan operasi penambahan dalam pangkalan data Sistem Buku Telefon.

(iii) Penghitungan Prasyarat

Prasyarat menyatakan hubungan antara pemboleh ubah keadaan dan pemboleh ubah input sebelum suatu operasi dijalankan. Ia boleh disahsahihkan dan kes ujian yang mewakili prasyarat juga boleh dihasilkan. Berdasarkan kepada skema *AddTelephone*, iaitu daripada predikat $name \overset{TM}{} known$, suatu prasyarat yang perlu ada ialah

> Nama pemilik telefon yang akan ditambah ke dalam pangkalan data Sistem Buku Telefon mestilah belum wujud dalam senarai pangkalan data Sistem Buku Telefon.

Berdasarkan kepada prasyarat yang dinyatakan, dua kes ujian boleh disediakan, iaitu yang memenuhi prasyarat dan yang tidak memenuhi prasyarat. Kedua-dua kes ujian ini tidak melibatkan output ujian. Minat dalam pengujian ini hanyalah untuk melihat kejayaan atau kegagalan sesuatu operasi.

Untuk menjayakan operasi penambahan dalam Sistem Buku Telefon, nama dan nombor telefon yang akan ditambah mestilah belum wujud lagi dalam senarai pangkalan data Sistem Buku Telefon. Input ujian untuk skema *AddTelephone* melibatkan pemboleh ubah keadaan *known* dan *telephone* serta pemboleh ubah input

name? dan *phone?* Nilai untuk pemboleh ubah keadaan ini boleh dinyatakan sebagai set kosong. *ali* akan digunakan sebagai nilai bagi pemboleh ubah input untuk nama (*name?*) dan *N8782222* sebagai nilai bagi pemboleh ubah input untuk nombor telefon (*phone?*). Kes ujian ini boleh distrukturkan seperti berikut:

Input Ujian:
Pemboleh ubah keadaan:
known = { }
telephone = { }
Pemboleh ubah input:
name? = ali
phone? = N8782222

Untuk menggagalkan operasi penambahan, pemboleh ubah keadaan akan menggunakan *ali* sebagai pemilik kepada nombor telefon *N8782222*. Sekali lagi *ali* akan digunakan sebagai nilai bagi pemboleh ubah input untuk nama (*name?*) dan *N8782222* sebagai nilai bagi pemboleh ubah input untuk nombor telefon (*phone?*). Berikut adalah kes ujian yang sepatutnya menggagalkan operasi penambahan.

Input Ujian:
Pemboleh ubah keadaan:
known = {ali}
telephone = {(ali , N8782222)}
Pemboleh ubah input:
name? = ali
phone? = N8782211

Jika diperhatikan, input ujian yang digunakan untuk menghasilkan kes ujian dari prasyarat skema *AddTelephone* ini mempunyai persamaan dengan input ujian yang digunakan untuk menghasilkan kes ujian dari ciri-ciri yang perlu ada dalam skema *AddTelephone*.

Umumnya langkah-langkah yang perlu dijalankan untuk menyediakan kes ujian adalah seperti berikut:

[1]. Menggunakan skema awalan yang disediakan oleh guru sebagai kes ujian yang mewakili teorem keadaan awal: Skema awalan dalam spesifikasi Z terdiri dari pemboleh ubah keadaan dengan diawalkan dengan nilai-nilai tertentu. Kebiasaannya pemboleh ubah keadaan ini diawalkan sebagai set kosong. Oleh kerana pemboleh ubah keadaan ini telah diberikan nilai awal, maka skema awalan akan digunakan terus sebagai kes ujian.

[2]. Menggunakan skema operasi yang disediakan oleh guru untuk menghasilkan kes ujian yang mewakili:
- Ciri-ciri yang menjayakan operasi: Skema operasi terdiri daripada pemboleh ubah keadaan, pemboleh ubah input atau pemboleh ubah output. Untuk menghasilkan kes ujian yang mewakili ciri-ciri yang menjayakan sesuatu operasi, nilai-nilai pemboleh ubah keadaan, pemboleh ubah input atau pemboleh ubah output dalam skema operasi mestilah memuaskan predikat skema operasi.
- Ciri-ciri yang menggagalkan operasi: Bagi penyediaan kes ujian yang menggagalkan sesuatu operasi, nilai-nilai pemboleh ubah keadaan, pemboleh ubah input atau pemboleh ubah output dalam skema operasi mestilah tidak dapat memuaskan predikat skema operasi.

[3]. Menggunakan prasyarat dalam skema operasi untuk menghasilkan kes ujian yang:
- Memenuhi prasyarat: Bagi menghasilkan kes ujian yang memenuhi prasyarat, nilai-nilai pemboleh ubah keadaan, pemboleh ubah input atau pemboleh ubah output dalam skema operasi dirujuk semula. Nilai-nilai pemboleh ubah dalam skema operasi mestilah memuaskan predikat yang mewakili prasyarat skema operasi.
- Tidak memenuhi prasyarat: Bagi kes ujian yang tidak memenuhi prasyarat, nilai-nilai pemboleh ubah keadaan, pemboleh ubah input atau pemboleh ubah output mestilah tidak dapat memuaskan predikat yang mewakili prasyarat skema operasi.

Kes ujian yang telah disediakan berdasarkan kepada ketiga-tiga langkah di atas perlu diuji terlebih dahulu. Ini penting untuk memastikan kes ujian yang disediakan dapat memberikan keputusan yang tepat dengan spesifikasi yang diuji oleh guru. Jika tiada sebarang ralat, kes ujian yang telah disediakan dan diuji oleh guru akan digunakan. Jika wujud ralat, kes ujian perlu disemak semula bagi memastikan ia sesuai digunakan.

Daripada ketiga-tiga cara yang dijelaskan untuk penghasilan kes ujian, hanya kes ujian yang mewakili teorem keadaan awal dan kes ujian yang mewakili ciri-ciri skema operasi spesifikasi Z sahaja digunakan untuk pengesahsahihan secara pelaksanaan bersimbol. Prasyarat tidak dipilih kerana kes ujian yang dihasilkan dari ciri-ciri suatu skema operasi dalam spesifikasi Z turut merangkumi kesemua syarat yang telah ditetapkan dalam suatu skema operasi spesifikasi Z.

3.2 Pelaksanaan Bersimbol Spesifikasi Z Menggunakan Z/Eves

Dalam kajian ini, alatan Z/Eves version 2.1 [23] telah digunakan bagi melaksanakan spesifikasi Z secara bersimbol menggunakan kes ujian yang telah disediakan.

Spesifikasi Z boleh dilaksanakan secara bersimbol mengikut langkah-langkah berikut:

- sediakan spesifikasi Z yang ingin dilaksanakan dalam gaya format Fuzz (gaya yang dibaca oleh Z/Eves)
- sediakan input ujian yang telah dipilih dalam gaya format Fuzz juga
- laksanakan spesifikasi Z terhadap input ujian secara bersimbol menggunakan arahan-arahan Z/Eves
- dapatkan hasil pelaksanaan

Bagi mendemonstrasikan penggunaan Z/Eves, kes ujian yang telah dibincangkan dalam Bahagian 3.1 dinyatakan semula seperti dalam Jadual 1.

Jadual 1 Kes ujian

No	Jenis kes ujian	Input Ujian	Output Ujian(Jangkaan)
1	Teorem keadaan awal	Skema *InitTelephoneBook*	Berjaya
2	Ciri-ciri yang perlu ada (untuk kes yang berjaya)	*known = {}* *telephone ={}* *name? = ali* *phone? = N8782222*	*known' = {ali}* *telephone' = {(ali , N8782222)}*
3	Ciri-ciri yang perlu ada (untuk kes yang gagal)	*known = {ali}* *telephone = {(ali , N8782222)}* *name? = ali* *phone? = N8782211*	Operasi sepatutnya gagal

Berdasarkan Jadual 1, kes-kes ujian dibina mengikut salah satu bentuk yang berikut:

jika hanya satu skema tanpa input ujian;

```
test case | (schema name)
```

atau jika terdapat satu skema dengan input ujian;

```
test case | (schema name)[test input]
```

atau jika melibatkan lebih daripada satu skema;

```
test case | (schema name)[test input];
                (schema name)[test input];
                ...
```

Berdasarkan kepada bentuk yang disediakan, kes ujian yang pertama disediakan seperti berikut:

```
TestTelephone1 | InitTelephoneBook
TestTelephone2 | .InitTelephoneBook;
      AddTelephone[ali/name?,N8782222/phone?]
TestTelephone3 | .TestTelephone2;
      AddTelephone[ali/name?,N8782211/phone?]
```

Seterusnya kes ujian yang pertama dilaksanakan dengan menggunakan arahan Z/Eves berikut:

```
try TestTelephone1;
```

diikuti dengan arahan,

```
prove by reduce;
```

Output dari hasil pelaksanaan bersimbol ini adalah panjang. Dalam bahagian ini, penyataan output yang penting sahaja ditunjukkan. Bagi kes ujian pertama, hasil berikut diperolehi:

```
Proving gives ...
telephone = {}
∧ known = {}
```

Seterusnya, hasil pelaksanaan terhadap kes ujian kedua dipaparkan seperti berikut:

```
Proving gives ...
  N8782222 ∈ .TELEPHONE
∧ ali ∈ .NAME
∧ telephone = { }
∧ known = { }
∧ telephone' = {(ali, N8782222)}
∧ known' = {ali}
```

Manakala hasil pelaksanaan terhadap kes ujian ketiga adalah seperti berikut.

```
Proving gives ..
false
```

3.3 Perbandingan Hasil Pelaksanaan dengan Output Jangkaan

Seterusnya perbandingan dilakukan di antara output ujian (rujuk Jadual 1) dengan hasil pelaksanaan bersimbol dalam Bahagian 3.2. Secara ringkasnya, dapat dikatakan:

- Untuk kes ujian yang pertama, output ujian menjangkakan keadaan awal sistem berjaya dimulakan tanpa sebarang data. Jangkaan ini berjaya dibuktikan kerana output Z/Eves menunjukkkan keadaan awal sistem bermula dengan kosong.
- Seterusnya dalam kes ujian kedua, pemboleh ubah keadaan selepas operasi penambahan, iaitu *known'* dan *telephone'* yang menjangkakan *ali* sebagai pemilik nombor telefon *N8782222* terbukti. Ini kerana output Z/Eves

menunjukkan pemboleh ubah keadaan selepas operasi penambahan, iaitu *known'* dan *telephone'* dapat membuktikan *ali* sebagai pemilik nombor telefon *N8782222*.

- Dalam kes ujian ketiga, operasi penambahan sepatutnya gagal dilaksanakan. Penyataan ini juga berjaya dibuktikan oleh Z/Eves. Ini kerana Z/Eves memaparkan output sebagai *false*.

Keutamaan kajian ini bukan sahaja terhadap proses pengesahsahihan spesifikasi Z, malah yang lebih utama adalah proses pengukuran terhadap tahap kesahsahihan spesifikasi Z juga turut dilakukan. Walaupun King (1976), Balzer *et al*. [13], Kemmerer [8] dan Kneuper [15] tidak melibatkan sebarang proses pengukuran dalam pengesahsahihan spesifikasi formal melalui teknik pelaksanaan bersimbol, motivasi untuk mengukur spesifikasi Z bertujuan untuk melihat tahap kesahsahihannya. Tahap kesahsahihan spesifikasi Z ini akan diwakili oleh suatu nilai numerik.

4.0 MENGUKUR PARAS KESAHSAHIHAN

Pengukuran paras kesahsahihan ini dilakukan dengan membandingkan di antara output ujian yang disediakan oleh guru dan hasil pengujian ke atas jawapan yang didokumenkan pelajar. Proses perbandingan ini dijalankan dengan melihat kepada nilai pemboleh ubah keadaan selepas operasi dalam output ujian yang disediakan oleh guru dan hasil pelaksanaan bersimbol jawapan pelajar. Output ujian bagi kes ujian kedua yang telah disediakan oleh guru dalam Jadual 1 dirujuk dan ditunjukkan seperti berikut:

$known' = \{ali\}$
$telephone' = \{(ali , N8782222)\}$

Nilai-nilai bagi pemboleh ubah keadaaan selepas operasi penambahan dijalankan, iaitu $\{ali\}$ dan $\{(ali, N8782222)\}$ merupakan dua nilai yang akan dibandingkan dengan nilai pemboleh ubah keadaan selepas operasi dijalankan dalam hasil pelaksanaan bersimbol jawapan pelajar. Selepas dibandingkan, pengiraan markah dilakukan bagi mewakili paras kesahihan spesifikasi.

Satu elemen yang perlu ditentukan dalam pengiraan ini ialah skema pemarkahan. Andaikan markah penuh untuk kes ujian kedua adalah 6 dengan pembahagian markah ditentukan seperti berikut:

- Jika *known'* ialah $\{ali\}$ markah yang diperolehi adalah 2
- Jika *telephone'* ialah $\{(ali, N8782222)\}$ markah yang diperolehi adalah 4

Kita tulis semula hasil pelaksanaan bagi kes kedua (rujuk Bahagian 3.2) seperti berikut:

```
Proving gives ...
  N8782222 ∈ .TELEPHONE
∧ ali ∈ .NAME
∧ telephone = { }
∧ known = { }
∧ telephone' = {(ali, N8782222)}
∧ known' = {ali}
```

Dengan merujuk kepada dua baris terakhir yang mengandungi pemboleh ubah keadaan selepas operasi dijalankan, iaitu *known'* dan *telephone'*, didapati *ali* berada dalam pangkalan data Sistem Buku Telefon (*known' = {ali}*) dan *ali* juga sebagai pemilik kepada nombor telefon *N8782222 (telephone' = {(ali , N8782222)}).* Kedua-dua baris ini juga menunjukkan hasil pelaksanaan bersimbol dalam jawapan pelajar menepati output ujian yang disediakan oleh guru seperti dalam Jadual 1. Sehubungan itu, pelajar akan mendapat markah penuh, iaitu 6.

Untuk proses perbandingan yang melibatkan ouput ujian yang disediakan guru berbeza dengan hasil pelaksanaan jawapan pelajar, kes ujian kedua digunakan sekali lagi. Namun andaikan pelajar lain telah menyediakan skema *AddTelephone* yang salah seperti berikut:

```
┌─AddTelephone_____________
├→ΔTelephoneBook
├→ name?: NAME
├→ phone?: TELEPHONE
├──
├→ name? TM {}
├→ known' = known Y {name?}
└──────────────────────────
```

Dalam skema ini, pelajar tidak menyatakan pemboleh ubah keadaaan selepas operasi, iaitu *telephone' = telephone* Y *{(name?,phone?)}* di bahagian predikat skema *AddTelephone*. Apabila skema *InitTelephoneBook* dan skema *AddTelephone* dilaksanakan secara bersimbol terhadap kes ujian kedua (*TestTelephone2*), hasil pelaksanaan ditunjukkan seperti berikut (baris yang penting sahaja ditunjukkan):

```
Proving gives ...
...
...
∧ known = {}
∧ telephone = {}
∧ telephone' ∈ NAME × TELEPHONE
∧ known' = dom telephone'
∧ known' = {ali}
```

Jika diperhatikan dalam hasil pelaksanaan di atas, pemboleh ubah keadaan selepas operasi, iaitu *known'*, menghasilkan *known'* =.*{ali}*. Pemboleh ubah keadaan selepas operasi, iaitu *telephone'* tidak wujud dalam hasil pelaksanaan bersimbol jawapan pelajar ini. Apabila ouput ujian dibandingkan dengan output Z/Eves dalam hasil pelaksanaan bersimbol jawapan pelajar di atas, maka markah untuk hasil pelaksanaan bersimbol jawapan pelajar tersebut adalah 2 sahaja.

Proses pembahagian markah ini bukanlah berbentuk padanan terus, iaitu nilai pada pemboleh ubah keadaan selepas operasi dipadankan satu ke satu dengan output jangkaan. Sebaliknya predikat itu akan dianalisis. Setiap kelainan sama ada kelebihan atau kekurangan operator serta nilai simbol akan ditolak sebanyak satu bagi setiap kesalahan sehingga markah menjadi kosong.

Sebagai contoh, jika jawapan pelajar mengandungi

$$telephone' = \{(ali, N8782222)\} \cap .telephone$$

maka 4 markah yang diberikan akan ditolak 1.

Seterusnya, dengan cara yang sama, markah keseluruhan bagi mengukur kesahihan spesifikasi yang disediakan boleh dikira dengan menggunakan jadual 2.

Jadual 2 Skema pemarkahan pengukuran kesahihan spesifikasi

Kes Ujian	Markah Penuh	Pemberat	Markah diperolehi	Peratusan Markah Diperolehi
1	N_1	P_1	n_1	$100 \times P_1/(P_1 + P_2 + P_3) \times n_1/N_1$
2	N_2	P_2	n_2	$100 \times P_2/(P_1 + P_2 + P_3) \times n_2/N_2$
3	N_3	P_3	n_3	$100 \times P_3/(P_1 + P_2 + P_3) \times n_3/N_3$

Peratusan markah keseluruhan ialah;

$$= 100 \times ((P_1 \times n_1/N_1) + (P_2 \times n_2/N_2) + (P_3 \times n_3/N_3)) / (P_1 + P_2 + P_3)$$

5.0 KESIMPULAN

Dalam kertas kerja ini, satu kaedah bagi mengukur paras kesahsahihan spesifikasi Z telah dibincangkan. Pelaksanaan bersimbol bagi mengesahsahihkan spesifikasi telah dilakukan semenjak tahun 80-an lagi. Dengan berbantukan alatan Z/Eves, pelaksanaan bersimbol digunakan untuk mengesahsahih spesifikasi Z, dan seterusnya hasil pelaksanaan digunakan untuk mengukur paras kesahsahihannya. Contoh aplikasi bagi hasil yang dibincangkan dalam kertas kerja ini adalah dalam persekitaran pendidikan. Dengan adanya peralatan yang menyokong kaedah ini, paras

kesahsahihan spesifikasi boleh dinilai secara automatik, sehubungan itu dapat meringankan beban pensyarah selain daripada penyemakan yang konsisten dapat dilakukan.

RUJUKAN

[1] Salman, O. 1997. Animation of Z Specifications by Translation to Prolog. Laporan Teknikal. Department of Computer Science, University of Cairo, Egypt.

[2] Barden, R., S. Stepney dan D. Cooper. 1994. *Z in Practice*. New York: Prentice Hall.

[3] Hayes, I. J. dan C. B. Jones. 1991. Specifications are not (necessarily) executable. *Software Engineering Journal*. 4(6): 330-338.

[4] Cooke, D., E. Demirors, O. Demirors, A. Gates, B. Kraemer dan M. M. Tanik. 1996. Languages for the Specification of Software. *Journal of Systems and Software*. 32(3): 269-308.

[5] Mohd Zaki Haji Ghazali. 1996. Sistem Untuk Mentahkik Spesifikasi Formal Z. Tesis Sarjana Teknologi Maklumat. Universiti Kebangsaan Malaysia.

[6] Fuchs, N. E. 1992. Specifications are (preferably) executable. *Software Engineering Journal*. 7(5): 323-334.

[7] Fields, B. dan M. E. Goransson. 1992. A VDM Case Study in Mural. *Proceedings of IEEE Transaction of Software Engineering 1992*. 279-295.

[8] Kemmerer, R. A. 1985. Testing Formal Specifications to Detect Design Errors. *Proceeding of IEEE Transactions on Software Engineering '85*. 32-43.

[9] Abdullah Mohd Zin dan Zarina Shukur. 2004. Testing the Satisfiability of Formal Specification. *Malaysian Journal of Computer Science*. 17(1): 42-51.

[10] Zarina Shukur. 1999. The Automatic Assessment of Z Specifications. Tesis Ph.D. Univ. of Nottingham.

[11] Wu, F. dan T. Yi. 2004. *Measuring Z Specifications. ACM SIGSOFT Software Engineering Notes*. 29(5): 1-5.

[12] Huang, S.-J. dan R. Lai. 2003. Measuring the Maintainability of a Communication Protocol Based on Its Formal Specification. *IEEE Transaction on Software Engineering*. 29(4): 327-344.

[13] Balzer, R. M., N. M. Goldman dan D. S. Wile. 1982. Operational Specification as the Basis for Rapid Prototyping. *ACM Sigsoft Software Engineering Notes*. 7(5): 3-16.

[14] Swartout, W. R. 1983. The GIST Behaviour Explainer. Laporan Penyelidikan ISI/RS-83-3, USC Information Sciences Institute.

[15] Kneuper, R. 1989. Symbolic Execution as a Tool for Validation of Specifications. Tesis Ph.D. University of Manchester.

[16] Douglas, J. dan A. A. Kemmerer. 1994. Aslantest: a Symbolic Execution Tool for Testing Aslan Formal Specifications. *International Symposium on Software Testing and Analysis '94*. Seattle, Washington, United States. 15-27.

[17] Manoranjan, S., M. Butler, M. Leuschel dan S. Ramesh. 2007. Automatic Testing from Formal Specifications. *Lecture Notes in Computer Science*. 4454:95-113.

[18] Le Gall, P., N. Rapin dan A. Touil. 2007. Symbolic Execution Techniques for Refinement Testing. *Lecture Notes in Computer Science*. 4454:131-148.

[19] King, J. C. 1976. Symbolic execution and program testing. *Proceeding of Communication of the ACM 1976*. 385-394.

[20] Singh, H., M. Conrad dan S. Sadeghipour. 1997. *Test Case Design based on the Classification-tree Method*. Proc. of 1st International Conference on Formal Engineering Methods '97. 252-261.

[21] Sommerville, I. 1996. *Software Engineering*. Workingham: Addison Wesley Publishing Company

[22] Jacky, J. 1997. The Way of Z: *Practical Programming with Formal Methods*. Cambridge University Press, Australia.

[23] Saaltink, M. 1997. *The Z/EVES User Guide*. Ottawa. ORA Canada.