Jurnal Teknologi, 49(D) Dis. 2008: 1–12
© Universiti Teknologi Malaysia

PENDEKATAN FORMALISASI MODEL BERORIENTASI OBJEK DENGAN MODEL FORMAL: SATU TINJAUAN

NORAIDA HAJI ALI[1], ZARINA SHUKUR[2] & SUFIAN IDRIS[3]

Abstrak. Kaedah formal yang digunakan dalam pembangunan perisian atau sistem adalah berasaskan kepada notasi dan teorem matematik dalam reka bentuknya. Ia menggunakan model dan notasi matematik bagi spesifikasi, reka bentuk dan penentusahan perisian. Kaedah orientasi objek pula merupakan satu metodologi reka bentuk bermodular yang berasaskan kepada huraian satu sistem kepada koleksi objek yang berinteraksi antara satu sama lain. Untuk mempertingkatkan lagi analisis dalam kaedah orientasi objek; satu teknik dikenali sebagai proses formalisasi, iaitu mengintegrasikannya dengan notasi formal yang sesuai dicadangkan. Kertas ini akan membincangkan kajina-kajian lepas yang telah dilakukan dalam proses formalisasi ini dan fokus perbincangan mengenai pendekatan yang telah digunakan bagi tujuan di atas. Selain daripada itu, kertas ini juga membincangkan mengenai kebaikan atau faedah yang boleh diperolehi daripada proses ini dan mengapa ia menjadi satu keperluan dalam proses pembangunan sistem.

Kata kunci: Pemodelan berorientasi objek; pemodelan formal; UML; Z dan Object-Z

Abstract. Formal method for software or system development involves the exploitation of related mathematic notation and theorem. Object orientation on the other hand is a modular methodology in which it dissociate a system into related collection of objects. To enhance the object oriented analysis; a techniques known as formalization process that integrate the formal notation with selected object oriented method is proposed. As such, the paper will review previous work specifically focusing on the approaches taken. In addition, benefits obtained formalization process used in establishing the technique and the needs for this particular technique are also discussed further in this paper.

Keywords: Object-oriented model; formal method; UML; Z and Object-Z

1.0 PENDAHULUAN

Model merupakan satu perkara yang paling penting dalam prinsip kejuruteraan. Model digunakan oleh jurutera untuk menggambarkan bentuk atau kelakuan sesuatu binaan yang hendak dibina atau dibangunkan. Kebiasaannya, bentuk simbol grafik, hubungan dan penerangan dalam bentuk teks digunakan untuk menggambarkan

[1] Jabatan Sains Komputer, Fakulti Sains dan Teknologi, Kolej Universiti Sains dan Teknologi Malaysia, 21030 Kuala Terengganu, Terengganu
Tel: 09-6683260, Faks: 09-6694660. Email: aida@kustem.edu.my

[2&3] Jabatan Sains Komputer, Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia, 46300 Bangi, Selangor. Tel: 03-89216720, Faks: 03-89256723. Email: zs@ftsm.ukm.my; Tel: 03-89216181, Faks: 03-89256723. Email: si@ftsm.ukm.my

model tersebut. Antara faedah dan kebaikan pemodelan ini ialah dapat memerihalkan keperluan, kefungsian, reka bentuk serta prestasi bagi sesuatu binaan atau pun sistem. Pemodelan juga digunakan untuk mendapatkan jangkaan kos atau bajet dan masa yang diperlukan untuk menyiapkan sesuatu sistem.

Fasa keperluan pengguna merupakan satu perkara yang amat penting dalam pemodelan sistem. Spesifikasi keperluan merangkumi struktur sistem, kelakuan bagi setiap komponen fizikal dan juga komunikasi antara modul. Secara tradisionalnya, pemodelan satu sistem boleh digambarkan secara perkataan atau melalui grafik. Walau bagaimanapun, pernyataan melalui pendekatan ini tidak dapat menyelesaikan masalah dari segi kefahaman sesuatu keperluan sistem. Keadaan ini akan menimbulkan ketidakfahaman atau salah faham antara pengguna dan jurutera sistem. Keadaan ini akan menyebabkan sistem menjadi tidak konsisten (*inconsistent*). Oleh itu, kaedah formal dapat memberi gambaran yang begitu jelas mengenai satu sistem dan juga keperluan sistem. Melalui kaedah formal ini, pembangun sistem dapat mengesahkan beberapa elemen-elemen yang penting, bagi menyelesaikan masalah yang tidak jelas dan juga dapat mengesan ralat reka bentuk sebelum proses pembangunan sistem itu bermula. Tanpa penggunaan spesifikasi formal, satu sistem itu berkemungkinan akan diuji secara meluas atau berulang selepas perlaksanaannya. Proses ini akan menyebabkan kos pengujian akan meningkat. Ia dapat menyediakan satu perancangan atau rangka tindakan untuk pembangun sistem, atau pengaturcara, dan boleh didokumentasikan sebagai dokumentasi pengguna.

Penggunaan kaedah formal dalam pembangunan perisian dapat menjamin penghasilan perisian yang memenuhi keperluan pengguna [1]. Kaedah formal juga berpotensi dalam menjangkakan kelakuan suatu program tanpa menulis dan melaksanakan sebarang kod pengaturcaraan [2]. Keberkesanan penggunaan kaedah formal ini telah dibuktikan oleh Bowen & Hinchey [3]. Namun begitu, masih ramai penyelidik terutama di Malaysia masih meragui mengenai kenyataan ini disebabkan oleh penggunaan matematik yang agak sukar difahami dan ditafsirkan oleh pengaturcara [4]. Kaedah orientasi objek seperti OMT [5] dan kaedah Fusion [6] adalah kaedah yang popular berdasarkan kepada pembinaan pemodelannya yang agak menarik, kaya dengan mekanisma yang berstruktur di tambah dengan kestabilan dari segi kursus latihan dan pelbagai rujukan yang telah ada. Di samping itu, penggunaan alatan pemodelan dapat membantu jurutera perisian untuk membangunkan satu sistem yang lebih bersistematik dan berkesan.

Kertas kerja ini akan membincangkan beberapa pendekatan yang boleh digunakan dalam formalisasi model berorientasi objek dengan model formal. Turut dibincangkan juga kajian-kajian yang telah dilakukan sebelum ini dalam formalisasi ini.

2.0 KAEDAH FORMAL

Kaedah formal merujuk kepada pelbagai teknik permodelan matematik yang digunakan dalam pemodelan sistem komputer. Kaedah formal yang digunakan dalam

pembangunan perisian adalah berasaskan kepada teknik matematik untuk menerangkan fungsi dan kekangan sistem. Ia menggunakan model dan notasi matematik bagi spesifikasi, reka bentuk dan penentusahan perisian. Teori set dan notasi logik digunakan untuk membina fakta yang mempunyai pernyataan yang jelas. Spesifikasi matematik yang dibina menggunakan notasi matematik ini akan mengurangkan kekaburan seperti yang berlaku pada mod tidak formal. Melalui kaedah ini, spesifikasi dan model yang dibangunkan dapat menerangkan sebahagian atau kesemua bahagian perlakuan sistem pada paras yang abstrak.

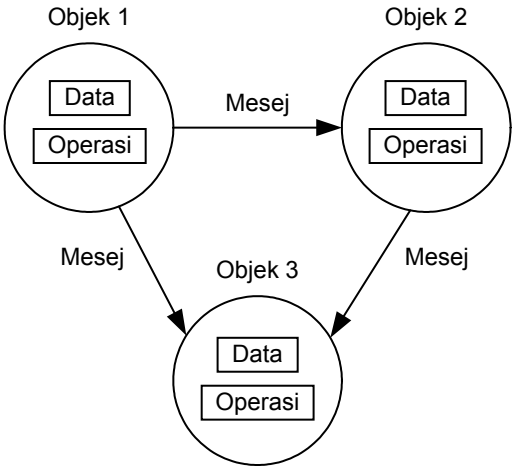
Melalui kaedah formal ini, pembangun sistem dapat mengesahkan beberapa elemen-elemen yang penting, bagi menyelesaikan masalah yang tidak jelas dan juga dapat mengesan ralat reka bentuk sebelum proses pembangunan sistem itu bermula. Kaedah formal dapat menghasilkan satu sistem yang mempunyai integriti yang tinggi dan ia disarankan untuk pembangunan sistem yang kritikal. Antara faedah atau kebaikan menggunakan spesifikasi formal dalam mewakilkan keperluan sistem ialah:

(i) Penggunaan spesifikasi formal dapat mengurangkan ralat keperluan.
(ii) Masalah ketidaklengkapan dan ketidakkonsistenan dapat diselesaikan.
(iii) Dapat mengurangkan risiko kerja-berulang yang disebabkan masalah keperluan dapat diselesaikan pada awal pembangunan sistem.
(iv) Mengurangkan kos pengujian sistem.

Antara bahasa spesifikasi yang telah wujud ialah LOTOS, VDM, CSP, B, Petri Net, RAISE, Promela/SPIN, VHDL, Z dan Objek-Z.

3. 0 MODEL BERORIENTASI OBJEK

Pemodelan berorientasi objek adalah satu metodologi reka bentuk yang bermodul berdasarkan kepada pembinaan satu sistem. Ia juga adalah satu koleksi komponen-komponen yang berinteraksi yang dikenali sebagai objek. Ia dimulakan dengan bahasa pengaturcaraan Simula '67, iaitu lanjutan Algol 60, di mana ia terlibat dengan pemodelan objek dalam dunia nyata bagi tujuan simulasi. Pemodelan berorientasikan objek adalah berdasarkan kepada konsep penyembunyian maklumat. Satu perisian berorientasikan objek boleh dikatakan sebagai satu kumpulan set objek yang dapat berinteraksi antara satu sama lain. Dengan kaedah ini, sesuatu perisian sistem difahami sebagai satu rangkaian objek-objek yang saling berkomunikasi di antara satu sama lain. Ciri-ciri ini dapat membezakan kaedah ini dengan kaedah fungsian atau berstruktur yang mana ia menggambarkan satu perisian terdiri daripada satu set fungsi yang berkongsi maklumat dan keadaan secara keseluruhan. Rajah 1 menggambarkan organisasi data dan fungsi di mana setiap objek mempunyai data dan fungsinya sendiri dan ia berkomunikasi antara satu sama lain melalui penghantaran mesej.



Rajah 1 Konsep asas pemodelan berorientasi objek

Satu objek mempunyai keadaan dan satu set operasi di mana ia bertindak dalam keadaannya. Keadaan pula mempunyai koleksi pemboleh ubah keadaan atau atribut, dan sebahagiannya boleh diisytiharkan oleh objek lain. Dengan cara ini, objek boleh digubah oleh objek itu sendiri. Objek boleh berkomunikasi antara objek yang lain melalui mesej yang dihantar dan diterima. Objek berbeza dari jenis data abstrak dengan kewujudan keadaan mereka. Ini akan memberi kesan ke atas kaedah pengabstrakan data. Orientasi objek semakin bertambah popularitinya sebagai satu paradigma pengaturcaraan dalam dekad yang lalu dengan terciptanya pengaturcaraan Smalltalk-80. Selepas itu, banyak bahasa berorientasikan objek mula dibangunkan termasuk EiffeL, POOL dan FOOPS. Juga terdapat banyak lanjutan orientasi objek ke atas bahasa pengaturcaraan yang sudah wujud dibangunkan. Antaranya adalah lanjutan-C, iaitu C++ dan Objective-C, dan juga lanjutan Lisp, iaitu LOOPS dan Flavors [4].

Hari ini, pemodelan berorientasikan objek telah pun menjadi satu pendekatan yang piawai di dalam proses pembangunan sistem. Dalam keadaan biasa, bahasa berorientasikan objek seperti C++ atau JAVA telah menjadi *de facto* piawai untuk bahasa pengaturcaraan. Kaedah berorientasi objek pula mempunyai pelbagai jenis notasi. Antara kaedah yang telah diterbitkan ialah OMT, OOSE, OOA/OOD, OOSA, DOOS, OOAD dan OOD. Kebanyakan daripada kaedah-kaedah ini mempunyai pendekatan analisis mereka sendiri. Kekuatan dan ciri keistimewaan bagi setiap kaedah digabungkan dan ini menerbitkan satu bahasa pemodelan, iaitu UML [7]. Di antara kebaikan atau kelebihan pemodelan berorientasi objek ialah:

(i) Fasa analisis dan reka bentuk sistem menjadi lebih mudah difahami – corak pengaturcaraannya yang menyerupai keadaan dunia sebenar.
(ii) Pengaturcaraan ini membenarkan kod diguna-semula.

(iii) Fasa penyenggaraan sistem menjadi lebih mudah dilakukan kerana ia hanya mengambil kira objek-objek yang terlibat sahaja.
(iv) Dapat mengurangkan pembinaan kod yang berulang kerana ia boleh diguna-semula.
(v) Atur cara yang dihasilkan lebih lasak dan selamat.

4.0 PROSES FORMALISASI

Unified Modeling Languange (UML) masih lagi dianggap tidak berjaya sepenuhnya dalam penumpuan masalah yang berkaitan dengan kekurangan dalam ketepatan notasi. Persoalannya apakah lagi yang boleh dilakukan untuk menjadikan UML lebih tepat? Melalui kajian-kajian lepas, isu ini telah banyak diberi penekanan di mana penggunaan kaedah formal dan hasilnya agak memberangsangkan. Kaedah atau proses fomalisasi ini telah dikenal pasti tentang penggunaannya dan keperluannya dalam bidang akademik, mahupun dalam perindustrian sebelum kewujudan UML lagi. Proses formalisasi ini bertujuan untuk menyokong kebolehpercayaan dan ketepatan bahasa pemodelan untuk digunakan dalam apa jua konteks. Meskipun pemodelan berorientasikan objek mempunyai kekuatannya sendiri, penggunaannya yang agak kompleks dalam pembangunan perisian boleh mengundang pelbagai masalah. Terdapat tiga pendekatan [8] yang boleh digunakan untuk formalisasikan pemodelan berorientasikan objek, iaitu:

(i) Tambahan (*supplemental*)
Kaedah ini merupakan sebahagian daripada model tidak formal di mana gambaran atau perwakilan secara bahasa tabii digantikan dengan pernyataan-pernyataan formal. Model seumpama OMT dapat ditambah atau diperluas dengan menggunakan pernyataan matematik. Penambahan dalam pernyataan formal boleh menjadikan model ini lebih tepat dan lebih jelas, tetapi penggunaan semantik bagi grafik sememangnya tidak dapat ditakrifkan dengan tepat dalam pendekatan ini. Hanya semantik bagi data yang statik yang boleh ditentukan. Untuk membolehkan proses-proses industri dapat digunakan untuk pembangunan berorientasikan objek yang lebih tepat dan untuk pembangunan berorientasikan objek yang agak kompleks, maka ia mungkin boleh menggunakan semantik formal dalam rajah sebagai asas untuk alat-sokongan transformasi seperti langkah-langkah pembaikan. Secara keseluruhannya, pendekatan ini tidak melibatkan kerumitan kaedah formal yang tersembunyi daripada pengguna.

(ii) Bahasa Formal Lanjutan-Orientasi Objek (*OO-extended formal languange*)
Dalam pendekatan ini, notasi formal yang sedia ada diperluaskan atau ditambahkan lagi dengan ciri-ciri Orientasi Objek. Kebanyakan tambahan

Orientasi Objek dalam notasi formal telah dicadangkan dalam kajian literatur seperti Z++ dan Objek-Z. Biasanya penambahan dalam konsep pemodelan Orientasi Objek kepada notasi formal lebih kepada penambahbaikan terhadap kualiti dalam asas bahasa formal. Pendekatan ini akan memberi hasil dalam perkayaan notasi formal. Hasilnya, dalam perwakilan kelakuan objek dan juga struktur kelas, akan digabungkan dengan notasi formal. Dari perspektif secara praktikal, masalah dalam pembangunan notasi menggunakan pendekatan ini ialah kekurangan alatan untuk menganalisis. Model yang menggunakan notasi ini agak sukar difahami, dibaca dan diubah suai. Ini adalah kerana masalah ruang semantik yang agak luas antara konsep dunia sebenar dengan perwakilan matematik dalam notasi formal. Usaha amat diperlukan untuk memetakan konsep antara dunia sebenar dan notasi formal.

(iii) Integrasi antara kaedah-kaedah (*Methods integration*)
Kebanyakan kajian integrasi ini lebih fokus kepada spesifikasi formal berbanding dengan model formal dalam Orientasi Objek. Tambahan lagi, penjanaan spesifikasi formal dapat melakukan proses analisis secara tepat dan menyediakan peluang-peluang lain untuk menyelesaikan masalah yang terhasil. Dalam pendekatan integrasi ini, kaedah spesifikasi formal akan dijana dari model tidak formal dengan tujuan untuk menggambarkan kaitan interpretasi/tafsiran formal dengan model tidak formal. Jika berlakunya pemetaan daripada struktur sintaksis dalam domain pemodelan tidak formal kepada artifak dalam domain yang dijelaskan secara formal, maka satu spesifikasi formal dari model tidak formal akan berlaku. Proses pemetaan ini digunakan untuk membina interpretasi model tidak formal.

Kewujudan pemetaan yang sedia ada tidak bermakna penjanaan bagi spesifikasi formal dari model Orientasi, janaan bagi spesifikasi formal memerlukan pengetahuan daripada pembangun sistem untuk membekalkan maklumat tambahan supaya ia dapat menyesuaikan format secara formal. Berpandukan kepada beberapa kajian literatur yang telah dilakukan, terdapat beberapa kaedah yang telah diperkenalkan untuk melaksanakan pendekatan ini. Antaranya ialah gabungan (*combining*), integrasi (*integrating*), terjemahan (*translation*), tafsiran (*interpreting*), transformasi (*transformation*) dan pemetaan (*mapping*). Kaedah yang berbeza ini mempunyai tujuan yang sama, iaitu untuk menggabungkan atau menggunakan kedua-dua jenis pemodelan ini, iaitu spesifikasi formal dan pemodelan berorientasikan objek untuk mewakilkan keperluan sistem.

4.1 Faedah/Kebaikan Formalisasi

Beberapa penyelidik luar negara telah membahaskan tentang kebaikan menerapkan pemodelan formal dalam model berorientasikan objek [9]. Antaranya pemodelan formal dapat:

(i) Menyokong kaedah pembangunan perisian.
(ii) Membantu pembangun sistem memahami perjalanan sistem dengan lebih jelas terutama sekali dalam spesifikasi keperluan.
(iii) Membenarkan komponen dalam perisian diguna semula (*reuse*).
(iv) Membuat pengesahan walaupun untuk sistem yang kritikal.
(v) Memperbaiki kaedah analisis dan reka bentuk dalam pembangunan perisian yang modelnya adalah jelas/koheren.
(vi) Digunakan untuk mengesahkan penggunaan semantik bagi suatu model.
(vii) Menjadikan fasa keperluan dan reka bentuk spesifikasi lebih tepat dan lengkap/rapi.

Untuk menghasilkan satu sistem yang lebih berkualiti dalam pemodelan berorientasikan objek, ia memerlukan satu usaha untuk memformalisasikan pemodelan berorientasikan objek dengan menggabungkan objek-objek yang lazim dengan ketepatan dalam formaliti [9]. Antaranya ialah:

(i) Pemodelan berorientasikan objek dapat diperkukuhkan dengan melakukan integrasi ke atas beberapa aspek dalam reka bentuk berorientasikan objek.
(ii) Pembaikan dalam pemodelan berorientasikan objek dapat dilakukan melalui sifat pewarisannya dengan menyediakan pendekatan-pendekatan baru.
(iii) Penggubahan terhadap dasar dalam struktur objek menjadikan proses pengesahan dapat distrukturkan dengan lebih baik.

4.2 Perlaksanaan Proses Formalisasi

Hasil daripada tinjauan terhadap kajian-kajian lepas didapati pendekatan ketiga, iaitu integrasi antara kaedah-kaedah, banyak digunakan dalam memformalisasikan pemodelan berorientasi objek ini. Perbezaan yang ketara di antara penyelidikan yang telah dilakukan adalah terhadap kaedah yang digunakan dan juga pemilihan bahasa pemodelan yang dikaji. Untuk mendapat gambaran yang lebih jelas Jadual 1 menunjukkan hubungan antara penyelidik dan penyelidikan yang telah dilakukan.

Antaranya ialah Ledang dan Souquieres [10], telah menggunakan kaedah pengintegrasian untuk memformalisasikan model UML dengan spesifikasi formal B. Selain daripada itu, kajian yang dilakukan oleh McUmber dan Cheng [8] pula

Jadual 1 Hubungan antara penyelidik dan penyelidikan

Penyelidik (Tahun)	Proses Formalisasi yang Dikaji	Jenis Kaedah Formalisasi		
		Integrasi	Gabungan	Pemetaan
Moreira & Clark (1996)	Formalisasi rajah pekerjasama UML dengan spesifikasi objek-Z.			✓
Weber. M. (1996)	Menggabungkan diagram kelas dan diagram keadaan (*statechart*)		✓	
Bertino *et al.* (1996)	Menterjemahkan diagram OMT kepada spesifikasi B			✓
France *et al.* (1997)	Menjana spesifikasi Z daripada Model Objek Fusion secara automatik	✓		
Wang *et al.* (1997)	Formalisasi model OMT dengan menggunakan LOTOS	✓		
Meyer *et al.* (1999)	Menterjemahkan diagram OMT kepada spesifikasi B			✓
Ledang dan Souquieres (2001)	Formalisasi model UML dengan spesifikasi formal B	✓		
McUmber dan Cheng (2001)	Formalisasi model UML dengan spesifikasi formal, iaitu Promela	✓		
Kim dan Carrington (2000)	Formalisasi model UML dengan spesifikasi objek-Z			✓
Bruel & France (2000)	Memetakan FMT (Fusion Modeling Techniques) kepada notasi spesifikasi formal Z			✓
Hvannerg (2001)	Menggabungkan UML dan spesifikasi formal Z		✓	
Bittner *et al.* (2003)	FMT (Fusion Modeling Techniques) kepada notasi spesifikasi formal Z		✓	

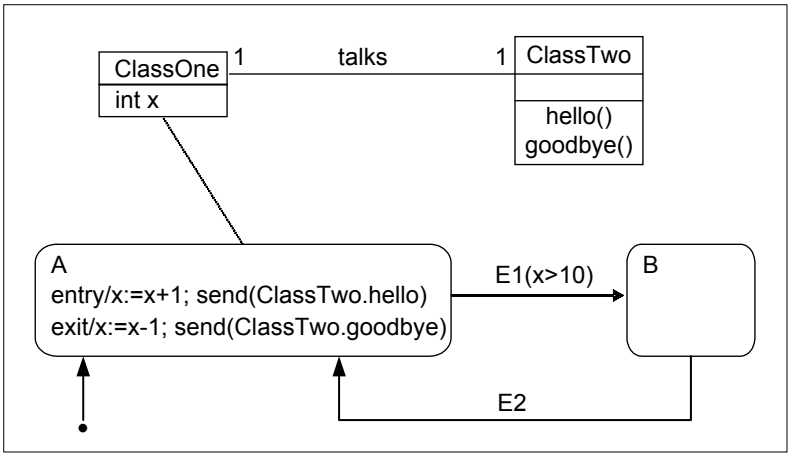
lebih fokus kepada memformalisasikan model UML dengan spesifikasi formal, iaitu Promela. Tujuan proses formalisasi ini ialah untuk mengaitkan semantik formal dengan diagram UML. Satu kajian lain pula oleh Bittner [11], menggunakan kaedah gabungan dengan menggabungkan Fusion[C+94] dengan spesifikasi formal Z. France *et al.* [12] pula telah kaedah pemetaan dalam membangunkan satu prototaip, iaitu FuZE, yang dapat menjana spesifikasi Z daripada Model Objek Fusion secara automatik.

Selain itu, Bruel *et al.* [13] pula memetakan FMT (Fusion Modeling Techniques) kepada notasi spesifikasi formal Z. Wang *et al.* [14] pula telah memformalisasikan

model OMT dengan menggunakan LOTOS melalui kaedah integrasi. Manakala Weber [15] telah menggabungkan diagram kelas dan diagram keadaan (*statechart*). Meyer *et al*. [16] dan Bertino *et al.* [17] juga menterjemahkan diagram OMT kepada spesifikasi B. Satu lagi kajian dilakukan oleh Moreira [18] yang menerapkan kaedah pemetaan dalam proses formalisasi tetapi lebih fokus kepada rajah pekerjasama UML yang lebih khusus kepada kes-guna yang terlibat dalam model UML. Kajian yang dilakukan oleh Kim dan Carrington [19, 20] telah mengaplikasikan kaedah pemetaan untuk memformalisasikan model UML dengan spesifikasi objek-Z. Mereka memberi fokus kepada pemetaan ke atas rajah turutan UML dan rajah carta keadaan UML. Penyelidik lain, iaitu Hvannerg [21] pula menggunakan kaedah penambahan elemen Z dalam UML supaya spesifikasi Z dapat disesuaikan dan boleh digunakan dalam kitar hayat pembangunan sistem. Secara keseluruhannya, terdapat beberapa kaedah yang telah diterapkan oleh para penyelidik untuk memformalisasikan pemodelan berorientasikan objek dengan pemodelan formal ini seperti yang telah dijelaskan sebelum ini. Perbezaan antara kajian mereka tertumpu dari segi pilihan model orientasi objek dan model formal yang dikaji. Selain daripada itu juga, terdapat perbezaan dari segi fokus kajian mereka.

5.0 CONTOH PERLAKSANAAN PROSES FORMALISASI

Untuk memahami lebih lanjut lagi mengenai proses formalisasi ini, satu contoh dipilih, iaitu formalisasi UML dengan spesifikasi formal, Promela, yang diutarakan oleh McUmber dan Cheng [8]. Kajian ini lebih fokus kepada memformalisasikan model berorientasikan objek,UML, dengan spesifikasi formal Promela. Pada akhir kajian, satu alatan telah dibangunkan, dan ia dikenali sebagai Hydra. Fungsi utama Hydra ini ialah menjana satu rangka kerja dalam bahasa Promela dari rajah UML. Rajah 2 menggambarkan satu model ringkas UML yang mempunyai 2



Rajah 2 Rajah kelas dan rajah pekerjasama UML

kelas, iaitu *ClassOne* dan *ClassTwo*. Model dinamik (rajah pekerjasama) bagi kelas *ClassOne* digambarkan dalam rajah yang sama dan digambarkan oleh garis putus-putus.

```
1   chan evq=[10] of {mtype, int};
2   chan evt=[10] of {mtype,int};
3   chan wait=[10] of {int mtype};
4   mtype={goodbye, hello, E1,E2}'
5   typedef ClassOne_T {
6                       int x;
7                       }
8   ClassOne_T, ClassOne_V;
9   chan ClassTwo_q=[5] of {mtype};
10  chan t=[1] of {mtype};
11  mtype={free};
12  proctype ClassOne()
13  {
14  mtype m;
15  int dummy;
16                  ClassOne_V.x=1;
17  /*Init state */
18                  goto A;
19  /*State A */
20  A:              printf("in state ClassOne.A");
21                  Atomic {classOne_V.x=ClassOne_v.x+1;
22  ClassTwo_q!hello;; }
23  A_G:
24                  evq!E1,_pid
25                  atomic{if :: !t?[free] o
26  t!free  ::  else skip fi ; }
27                  if
28                  ::  evt??E1,eval(_pid)          t?free; if
29                     :: ClassOne_V.x>10
30                     ClassOne_V.x=ClassOne_V.x-1;
31  ClassTwo_q!goodbye ;; goto B
32                   ::  else          goto A_G
33                   fi;
34                   fi;
35  /*State C */
36  B:              printf("in state ClassOne.B");
37                  evq!E2,_pid;
38                  atomic {if ::   !t?[free]
39  t!free   ::  else skip  fi; }
40                 if
41                 ::  evt??E2,eval(_pid)
42  t!free; goto A
43                  fi;
44  exit:     skip
45  }
46  proctype ClassTwo()
47  {
48  mtype m;
49  int dummy;
50  exit:       skip
51  }
```

Rajah 3 Penjanaan spesifikasi formal, Promela

Terdapat 2 kelas, iaitu *ClassOne* dan *ClassTwo* yang mempunyai hubungan *talks*. Dalam *ClassOne* terdapat satu atribut, *x* dan dalam *ClassTwo* terdapat dua operasi, iaitu *hello()* dan *goodbye()*. Dalam rajah pekerjasama terdapat 2 keadaan, iaitu keadaan A dan keadaan B. Keadaan A mempunyai 2 mesej, iaitu *entry* dan *exit* di mana setiap mesej ini akan dihantar kepada *ClassTwo*. Penghantaran mesej ini akan mengubah keadaan A kepada keadaan B melalui peristiwa *E1*. Keadaan B akan bertukar kepada keadaan A melalui peristiwa *E2*. Model kelas ini boleh dipetakan kepada pemodelan formal Promela dengan mengaplikasikan beberapa peraturan yang telah ditetapkan [8]. Dengan menggunakan alatan yang dibina, iaitu Hydra, satu spesifikasi formal, iaitu Promela akan dijana seperti dalam Rajah 3.

6.0 KESIMPULAN

Berpandukan kepada tinjauan yang telah dilakukan, kebanyakan kajian yang telah dilakukan lebih fokus kepada pemilihan bahasa pemodelan orientasi objek dan bahasa pemodelan formal dan juga fokus kepada kaedah yang digunakan oleh mereka. Ia juga memberi tumpuan kepada diagram tertentu sahaja dalam kajian mereka. Namun begitu, dalam kebanyakan kertas kerja yang telah dibincangkan ini tidak membincangkan mengenai kesahihan dan prestasi bagi setiap kaedah formalisasi yang telah dilakukan. Sejauh mana keberkesanan dan penggunaannya bagi setiap kaedah formalisasi yang diutarakan tidak dibincangkan dengan terperinci. Kebanyakan kertas kerja ini menjelaskan bagaimana kaedah formalisasi ini dapat dilakukan dengan membicarakan kaedah dalam proses formalisasi yang digunakan untuk memformalisasikan model berorientasi objek dengan model formal. Contoh formalisasi yang disertakan dapat memberi gambaran bagaimana proses formalisasi tersebut berlaku. Kebaikan dan faedah proses ini juga dijelaskan dalam kertas ini.

RUJUKAN

[1] Terwilleger, R. B., M. J. Maybee dan L. J. Osterweil. 1989. An Example of Formal Specification as an Aid to Design and Development. *ACM SIGSOFT Software Engineering Notes*. 14(3): 266-272.

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

[3] Bowen J. P. dan M. Hinchey. 1995. Seven More Myths of Formal Methods. *IEEE Software*.12(4): 34-41.

[4] Zarina Shukur dan A. Rizal M.Yusof. Satu Eksperimen Dalam Pengaturcaraan Berasaskan Spesifikasi Formal Z . Bengkel C, Ftsm, UKM.

[5] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy dan W. Lorensen. 1991. *Object-Oriented Modeling and Design.* New York: Prentice Hall.

[6] Coleman, D., P. Arnold, S. Bododff, C. Dollin, H. Gilchrist, F. Hayes dan P. Jeremaes. 1994. *Object-Oriented Development: The Fusion Method.* Object Oriented Series. New Jersey: Prentice Hall.

[7] Evans, A., R. France, K. Lano dan B. Rumpe. 1999. The UML as a Formal Modeling Notation. *Lecture Notes in Computer Science Issue*. 1618: 336-348.

[8] McUmber, W. E dan B. H. C. Cheng. 2001. A General Framework for Formalizing UML with Formal Languages. International Conference on Software Engineering. 23: 433-442.

[9] Champeaux, D., P. America, D. Coleman, R. Duke, D. Lea dan G. Leavens. 1991. Formal Techniques for OO Software Development. Proceedings of the OOPSLA. 166-170.

[10] Ledang, H. dan J. Souquieres. 2001. Integrating UML and B Specification Techniques. Proceedings of Informatik 2001. Vienna, Austria: 53-58.

[11] Bittner, M. dan F. Kammueller. 2003. Translating Fusion/UML to Object-Z. Proceedings of the 1st ACM and IEEE Int. Conference on Formal Methods and Model for Codesign, (MEMOCODE 2003). France: 49-50.

[12] France, R. B., J. M. Bruel dan M. M. Larrondo-Petrie. 1997. An Integrated Object-Oriented and Formal Modeling Environment. *Journal of Object Oriented Programming.* 10(7): 25-34.

[13] Bruel, J. M. dan R. B. France. 1998. Transforming UML Models to Formal Specifications. First International Workshop UML'98. Mulhouse, France, LNCS 1618.

[14] Wang, E., H. Richter dan B. Chen. 1997. Formalizing and Integrating the Dynamic Model with OMT. Proceedings of the 19th Software Engineering International Conference. 45-55.

[15] Weber, M. 1996. Combining State Charts and Z for the Design of Safety-Critical Control Systems. FME'96: Industrial Benefit and Advances in Formal Methods, LNCS 1051. 307- 326.

[16] Meyer, E. dan J. Souquieres. 1999. A Systematic Approach to Transform OMT Diagrams to a B Specification. Proceedings of the Formal Method Conference. 1: 875-895.

[17] Bertino, E., D. Castelli dan F. Vitale. 1996. A Formal Representation for State Diagrams in the OMT Methodology. Proceedings of the SOFSEM: Theory and Practice of Informatics. 1175: 328-334.

[18] Moreira, A. dan R. Clark. 1996. Adding Rigorous to Object-Oriented Analysis. *Software Engineering Journal.* 11(5): 270-280.

[19] Kim, S. K. dan D. Carrington. 1999. Formalizing the UML Class Diagram using Object-Z. Proceedings of the 2nd IEEE Conference on UML. 1723: 83-98.

[20] Kim, S. K. dan D. Carrington. 2000. A Formal Mapping between UML Models and Object-Z Specifications. Proceedings of the ZB2000. 1878: 2-21.

[21] Hvannerg, E. T. 2001. Combining UML and Z in a Software Process. Proceedings of Informatik. Vienna, Austria. 635-640.