RANCANG BANGUN SIMULATOR ATM SEBAGAI PERANGKAT AJAR REKAYASA PIRANTI LUNAK

Ivransa Zuhdi Pane
BBTA3, BPPT
izpane@gmail.com

ABSTRAK

Rekayasa piranti lunak bermanfaat dalam pengembangan piranti lunak yang mendukung proses bisnis suatu organisasi. Agar tercipta piranti lunak yang sesuai dengan kebutuhan, proses tipikal rancang bangun piranti lunak selayaknya dimengerti dengan benar, tidak hanya oleh pihak pengembang namun juga oleh pihak kustomer, yang pada umumnya kurang memiliki pengetahuan tentang rekayasa piranti lunak. Pengembangan materi ajar yang berbasis pada kasus yang mudah dimengerti oleh pihak kustomer, seperti anjungan tunai mandiri, diharapkan dapat mendorong terbentuknya pemahaman di pihak kustomer, yang diperlukan untuk menyusun spesifikasi piranti lunak.

Kata kunci *:* rekayasa piranti lunak, pseudocode

ABSTRACT

Software engineering is useful in developing software that supports an organization's business processes. In order to create software that suits the needs, the typical software design process should be properly understood, not only by the developer but also by the customer, who are generally less knowledgeable about software engineering. The development of user-friendly, case-sensitive learning materials, such as automated teller machines, is expected to encourage customer-side understanding, which is required to develop software specifications.

Keyword*: software engineering, pseudocode*

PENDAHULUAN

Rekayasa piranti lunak merupakan disiplin rekayasa yang berhubungan dengan rancang bangun piranti lunak mulai dari tahap awal hingga tahap dimana piranti lunak dioperasikan. Produk piranti lunak yang dihasilkan dari proses rekayasa piranti lunak selanjutnya dapat dimanfaatkan untuk mendukung proses bisnis suatu organisasi. Untuk membangun piranti lunak yang tepat guna, maka dibutuhkan spesifikasi piranti lunak yang sesuai dengan kebutuhan pihak kustomer. Pada prakteknya, proses penggalian kebutuhan piranti lunak acap kali mengalami kendala yang disebabkan oleh sejumlah alasan. Salah satu alasan utama yang sering ditemukan adalah kurangnya pemahaman pihak kustomer terhadap proses rekayasa piranti lunak. Meski bukan menjadi domain tanggung jawab pihak kustomer, namun pemahaman ini sesungguhnya berpengaruh cukup signifikan dalam penyusunan spesifikasi piranti lunak, yang pada gilirannya dapat mendukung keterlibatan pihak kustomer dalam proses piranti lunak demi terciptanya piranti lunak yang sesuai dengan keinginan pihak kustomer.

Berdasarkan argumen yang dimaksud dalam alinea sebelumnya, maka makalah ini berupaya untuk mengembangkan suatu materi ajar yang diharapkan dapat membantu pihak kustomer, khususnya yang memiliki pengetahuan terbatas di bidang rekayasa piranti lunak, memahami proses tipikal pengembangan piranti lunak melalui studi kasus pengembangan piranti lunak simulator anjungan tunai mandiri (ATM). Adapun adopsi ATM sebagai kasus studi didasarkan pada tingginya popularitas penggunaan ATM yang diharapkan mendukung pemahaman pihak kustomer, bahkan orang yang sama sekali awam pada rekayasa piranti lunak, terhadap proses rekayasa piranti lunak. Makalah ini terlebih dahulu menguraikan teori dasar tentang proses rekayasa piranti lunak, untuk selanjutnya membahas studi kasus simulator ATM menurut proses

tipikal rekayasa piranti lunak, yaitu analisis, perancangan, implementasi dan operasi, serta menutup dengan kesimpulan dan saran.

PROSES REKAYASA PIRANTI LUNAK

Gambar 1 menunjukkan model air terjun (*waterfall model*), yang merupakan model klasik dan umum digunakan untuk menggambarkan tahapan dalam proses pengembangan piranti lunak. Adapun uraian mengenai masing-masing tahap adalah sebagai berikut :

A. Tahap Analisis

Tahap awal dari pengembangan piranti lunak, dimana pengembang melakukan penggalian kebutuhan (*requirement elicitation*) dan kajian terkait. Kegiatan tipikal dalam tahap ini adalah wawancara terhadap pihak kustomer tentang fungsionalitas dan data yang dibutuhkan, observasi kondisi atau lingkungan kerja dimana piranti lunak akan digunakan, dan kajian kelayakan realisasi kebutuhan menjadi piranti lunak yang nyata.

B. Tahap Perancangan

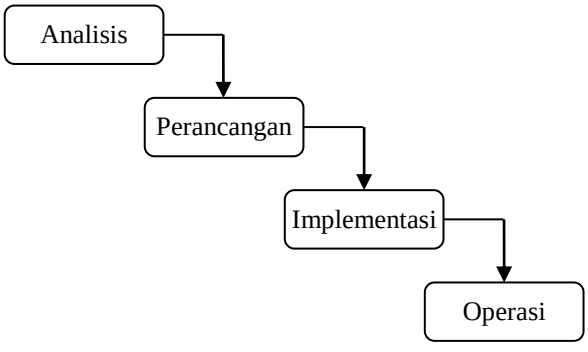
Tahap penuangan hasil analisis ke dalam ‘cetakan’ dasar piranti lunak, yang terdiri dari antarmuka pengguna, basis data dan algoritma. Pada pola pengembangan piranti lunak dengan bahasa pemrograman visual, antarmuka pengguna berbentuk sebagai jendela form (*form window*) yang berisi komponen interaktif, seperti tombol (*button*), teks, dan kotak daftar (*list box*), yang merupakan realisasi fungsionalitas yang digali dalam tahap analisis. Basis data merupakan entitas pengelola data, sebagai realisasi data yang disusun dalam tahap analisis. Sedangkan algoritma berperan sebagai pengendali alur eksekusi, dan penghubung antara fungsionalitas dan data.

C. Tahap Implementasi

Tahap realisasi gagasan konseptual yang dikembangkan dalam tahap analisis dan perancangan, menjadi piranti lunak nyata. Kegiatan tipikal dalam tahap ini adalah penyusunan kode program (*coding*) dan pengujian kualitas piranti lunak (*testing*). Dewasa ini, *coding* umumnya dilaksanakan dengan bahasa pemrograman visual yang lebih interaktif dan mudah dipahami. Sedangkan *testing* dilaksanakan terhadap hasil *coding* yang bertujuan untuk memastikan keterpenuhan kebutuhan fungsionalitas dan data yang didefinisikan dalam tahap analisis.

D. Tahap Operasi

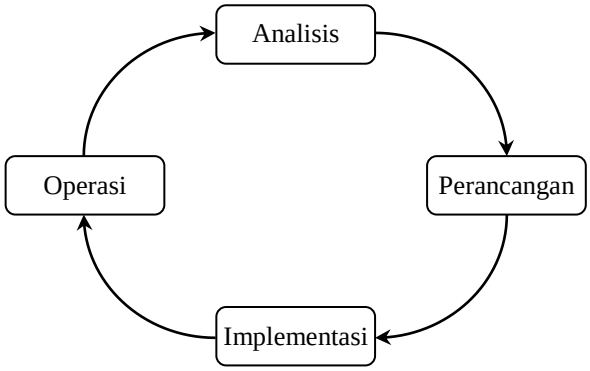
Tahap penggunaan piranti lunak mulai dalam kondisi uji coba hingga kondisi operasional sesungguhnya. Tahap ini juga meliputi kegiatan perawatan piranti lunak, seperti perbaikan piranti lunak dari kesalahan (*defect*) yang tidak terdeteksi dalam tahap implementasi, atau penambahan fungsionalitas dan data baru, baik sebagai pelengkap maupun sebagai peningkat kemampuan (*upgrade*).



Gambar 1. Model air terjun (*waterfall model*).

Secara ideal, pengembangan piranti lunak berlangsung secara berkesinambungan, dimana setiap kali suatu piranti lunak selesai dibangun dan mulai dioperasikan, maka piranti lunak tersebut selanjutnya dikembangkan kembali pada tahapan berikutnya dan diadaptasikan sesuai tuntutan kebutuhan baru yang semakin berkembang. Dalam hal ini, piranti lunak dikatakan berevolusi sesuai siklus hidup pengembangan piranti lunak yang ditunjukkan dalam Gambar 2. Konsep ini

sesungguhnya mendorong suatu organisasi untuk mengembangkan sumber daya manusia yang memiliki kompetensi di bidang rekayasa piranti lunak untuk dapat mengembangkan piranti lunak keperluannya sendiri, ketimbang mengadakan piranti lunak komersial atau meminta pihak ketiga untuk melakukan pengembangan piranti lunak yang jelas memakan biaya lebih tinggi.



Gambar 2. Siklus hidup pengembangan piranti lunak.

STUDI KASUS

Kasus pengembangan piranti lunak yang dibahas dalam makalah ini adalah simulator ATM, dengan beberapa batasan berikut :

- Simulator ATM merupakan simulasi dari mesin ATM sesungguhnya di lingkungan kerja komputer personal (mesin ATM virtual);
- Transaksi yang disimulasikan tidak melibatkan barang atau produk fisik, seperti mesin ATM, buku rekening, kartu ATM atau uang tunai, secara nyata;
- Jenis transaksi yang dikembangkan terbatas pada transaksi sederhana untuk membantu pemahaman proses rekayasa piranti lunak secara sederhana;
- Interaksi antara pengguna dan simulator ATM diasumsikan sama dengan interaksi yang sesungguhnya terjadi dengan beberapa modifikasi.

ANALISIS KEBUTUHAN PIRANTI LUNAK

Penggalian kebutuhan dalam tahap analisis piranti lunak dapat diawali dengan menguraikan fungsionalitas yang harus dilaksanakan, dan data yang harus tersedia dan diolah oleh piranti lunak, melalui wawancara terhadap kustomer dan atau observasi terhadap objek rujukan dan lingkungan kerjanya. Kajian kelayakan terhadap fungsionalitas dan data kemudian dilaksanakan untuk menghasilkan spesifikasi kebutuhan yang dapat direalisasikan pada tahap rekayasa piranti lunak selanjutnya.

Pada kasus pengembangan simulator ATM, dengan observasi terhadap mesin ATM nyata, maka fungsi-fungsi yang diharapkan terealisasi dapat diuraikan dalam Tabel 1. Kajian sederhana terhadap delapan fungsi dalam Tabel 1 dapat dilakukan dengan memberikan peringkat prioritas dari berbagai aspek terkait untuk menentukan perlu tidaknya fungsi tersebut diadakan. Fungsi setor tunai dalam mesin ATM di Indonesia secara *de facto* belum terealisasi di seluruh penjuru negeri, sehingga kurang realistis untuk diadakan. Fungsi penggantian PIN ditunda karena memerlukan konsiderasi tentang algoritma enkripsi/dekripsi yang lebih mendalam, meski sesungguhnya diperlukan dipandang dari aspek sekuritas. Fungsi beli pulsa dianggap kurang diperlukan karena kegiatan ini sesungguhnya lebih banyak dilakukan secara nyata di gerai transaksi pulsa atau tercakup dalam fungsi *smart phone* pengguna dewasa ini. Namun, dengan diadakannya fungsi bayar tagihan (telepon, air atau listrik) dan fungsi bayar ZIS (Zakat, Infaq dan Sadaqah) yang memiliki banyak kemiripan fungsional, maka fungsi beli pulsa dapat pula diadakan. Sehingga secara keseluruhan terdapat empat fungsi yang layak untuk diadakan dan dilanjutkan pengembangannya ke tahap perancangan, seperti yang terangkum dalam Tabel 2.

Tabel 1. Uraian fungsi-fungsi simulator ATM

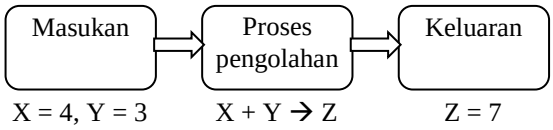
No	Nama Fungsi	Prioritas	Keterangan
1	Tarik Tunai	Sangat Perlu	Diadakan
2	Periksa Saldo	Sangat Perlu	Diadakan
3	Transfer	Perlu	Diadakan
4	Setor Tunai	Kurang Realistis	Ditiadakan
5	Ganti PIN	Perlu	Ditunda
6	Bayar Tagihan	Perlu	Diadakan, digabung
7	Beli Pulsa	Kurang Perlu	
8	Bayar ZIS	Sangat Perlu	

Tabel 2. Spesifikasi kebutuhan simulator ATM

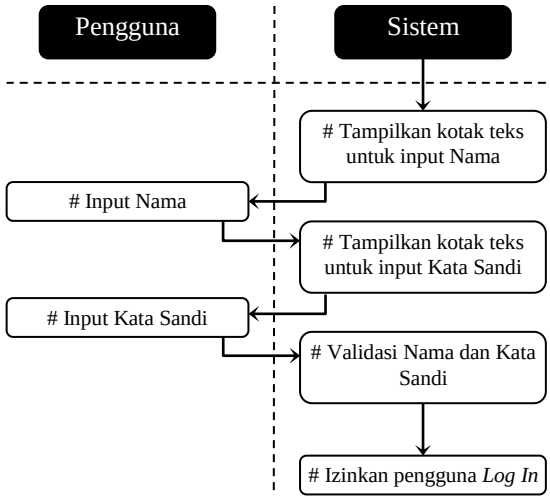
No	Nama Fungsi	Uraian
1	Tarik Tunai	Penarikan uang tunai (virtual)
2	Periksa Saldo	Pemeriksaan saldo rekening
3	Transfer	Pemindahan uang ke rekening lain
4	Pembayaran	Pembayaran tagihan, pulsa, ZIS

PERANCANGAN PIRANTI LUNAK

Dalam tahap perancangan piranti lunak, hasil dari tahap analisis dikaji dan diurai menjadi tiga entitas dasar pengolahan data, yaitu masukan (*input*), proses pengolahan (*process*) dan keluaran (*output*), seperti ditunjukkan dalam Gambar 3. Masukan merepresentasikan data sebagai bahan baku yang akan diolah. Proses pengolahan menguraikan operasi aritmatika dan logika yang diterapkan pada masukan untuk dikonversi masukan menjadi keluaran. Keluaran merupakan informasi hasil proses pengolahan, yang dapat berbentuk alfanumerik maupun grafis.



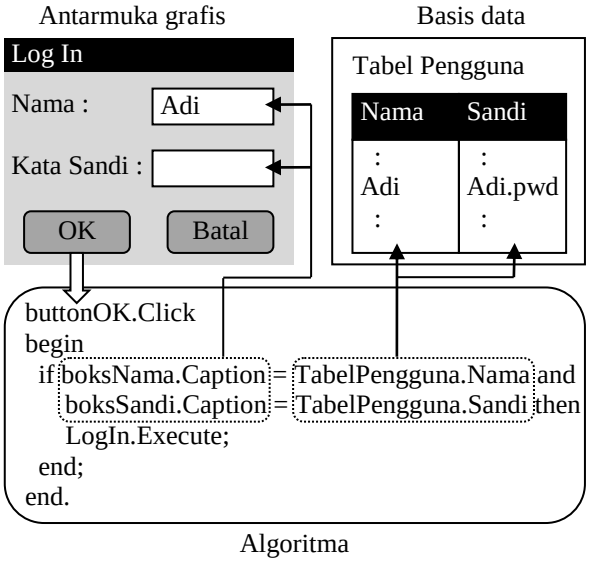
Gambar 3. Masukan, proses pengolahan dan keluaran.



Gambar 4. *Use case log in* pengguna ke sistem piranti lunak.

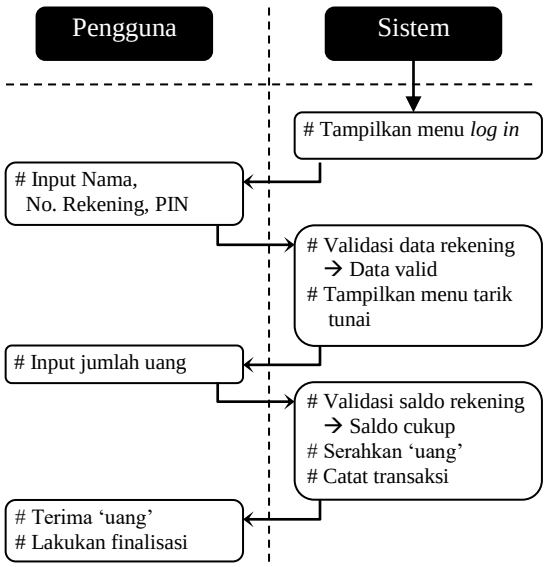
Pada umumnya, masukan dan keluaran merupakan data dan informasi yang selanjutnya menjadi acuan untuk penyusunan basis data, sementara proses pengolahan merupakan penghubung antara masukan dan keluaran yang menentukan antarmuka dan algoritma. Salah satu cara efektif

untuk menentukan masukan, keluaran dan proses pengolahan adalah menyusun skenario operasional dari piranti lunak dengan penggunaan *use case*. *Use case* adalah uraian perilaku sistem piranti lunak sebagai bentuk responnya yang terkait dengan suatu interaksi pengguna. Gambar 4 memperlihatkan *use case* sederhana untuk kasus *log in* pengguna ke suatu sistem piranti lunak.



Gambar 5. Antarmuka, basis data dan algoritma untuk *log in* pengguna ke sistem piranti lunak.

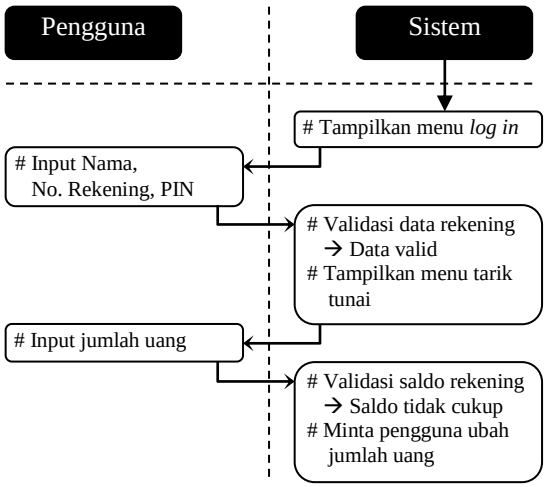
Kasus *log in* dalam Gambar 4 mengusulkan data Nama dan Kata Sandi sebagai masukan, validasi Nama dan Kata Sandi sebagai proses pengolahan, dan *Log In* sebagai keluaran. Implementasi kasus ini ke dalam rancangan antarmuka grafis, basis data dan algoritma ditunjukkan dalam Gambar 5. Dalam antarmuka, terdapat dua kotak teks untuk memungkinkan pengguna memasukkan Nama dan Kata Sandi, dan dua tombol untuk memungkinkan pengguna melakukan *Log In* (tombol OK) atau membatalkannya. Algoritma sistem yang diasosiasikan ke aksi tekan dari tombol OK (buttonOK.Click) selanjutnya akan mengakses basis data (Tabel Pengguna) dimana data nama dan kata sandi dikelola untuk keperluan validasi.



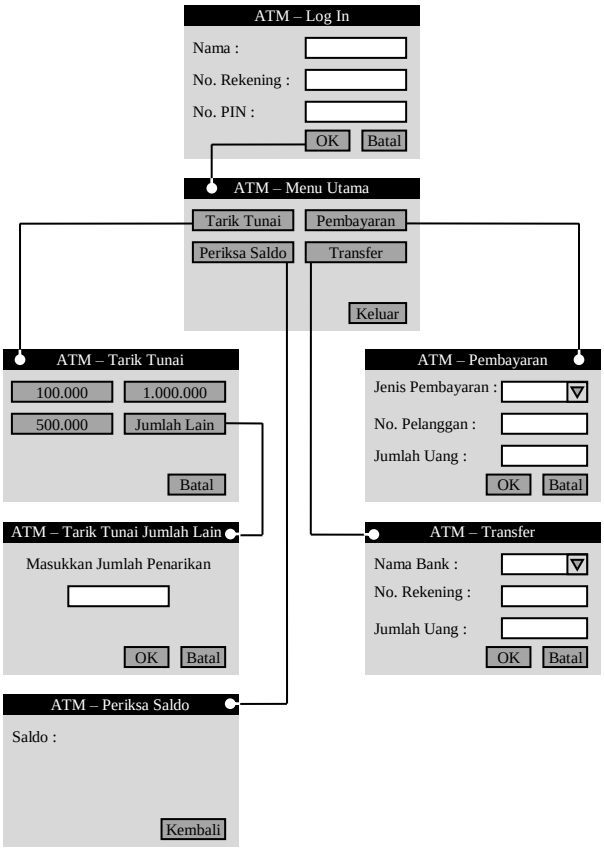
Gambar 6. *Use case* tarik tunai sukses.

Dalam kasus pengembangan simulator ATM, fungsi tarik tunai dapat diuraikan dengan *use case* dalam Gambar 6 untuk kasus sukses, dimana pengguna diasumsikan berhasil melakukan

penarikan uang tunai. Gambar 7 menunjukkan *use case* untuk kasus tarik tunai gagal akibat kurangnya saldo dalam rekening pengguna. Penguraian *use case* untuk kasus gagal perlu dilakukan untuk mencegah terjadinya kesalahan eksekusi piranti lunak dari seluruh kemungkinan yang diprediksi mungkin terjadi. Dengan kata lain, *use case* untuk kasus tarik tunai gagal akibat kesalahan input nama atau PIN, tidak tersedianya uang tunai dalam mesin ATM atau sebab lainnya juga patut dikembangkan. *Use case* untuk fungsi lainnya dapat diuraikan dengan cara yang sama.



Gambar 7. *Use case* tarik tunai gagal.



Gambar 8. Rancangan antarmuka grafis simulator ATM.

Dengan mempertimbangkan seluruh fungsi yang ada, maka antarmuka grafis simulator ATM dapat dirancang seperti yang ditunjukkan dalam Gambar 8. Aktivasi simulator ATM diawali

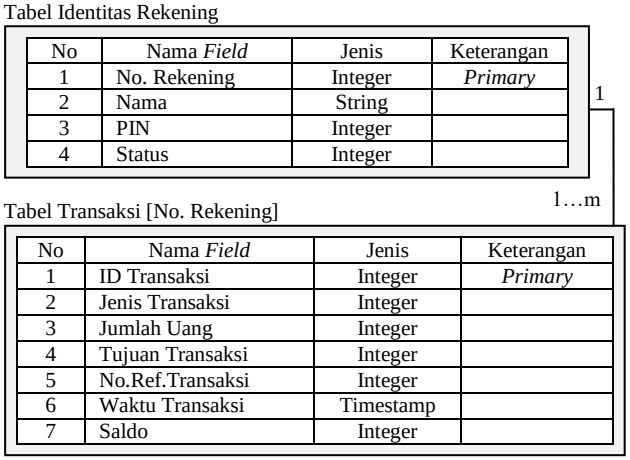
dengan pengisian identitas pengguna pada jendela *Log In*, yang diikuti dengan pengalihan ke jendela Menu Utama apabila identitas pengguna tervalidasi dengan benar. Pengguna selanjutnya dapat menentukan jenis transaksi yang diinginkan dengan menekan salah satu dari empat tombol transaksi (Tarik Tunai, Periksa Saldo, Pembayaran dan Transfer) di Menu Utama dan mengikuti alur eksekusi setiap transaksi, atau membatalkan transaksi dengan menekan tombol Keluar.

Seperti yang diperlihatkan dalam Gambar 8, menu Tarik Tunai menyediakan tiga jumlah penarikan tetap, yaitu 100.000, 200.000 dan 1.000.000,-, serta satu jumlah penarikan sesuai keinginan pengguna, yang bercabang pada menu Tarik Tunai Jumlah Lain. Menu Periksa Saldo memberikan informasi saldo rekening yang tersisa untuk membantu pengguna dalam mengambil keputusan pada saat melakukan transaksi jenis lainnya. Menu Pembayaran dilengkapi dengan kotak daftar kombinasi (*combo box*) yang berisi daftar jenis pembayaran, seperti listrik, pulsa atau ZIS, untuk mempermudah pengguna melakukan pemilihan dan mencegah kesalahan seandainya jenis pembayaran harus didefinisikan secara manual. Hal yang sama berlaku pula untuk menu Transfer, yang dilengkapi dengan combo box Nama Bank, untuk memudahkan pengguna memilih bank tujuan transfer.

Perancangan basis data untuk simulator ATM dapat dilakukan dengan merujuk *use case* dalam Gambar 6 dan 7. Informasi tentang nama, nomor rekening dan PIN pengguna harus diketahui oleh sistem untuk keperluan validasi identitas pengguna. Selanjutnya, *use case* untuk kasus tarik tunai sukses mengindikasikan perlunya pencatatan transaksi yang terkait dengan satu identitas pengguna tertentu dengan tujuan utama memperbarui saldo rekening pengguna yang bersangkutan. Sehingga, basis data untuk kasus simulator ATM dapat dirancang seperti yang ditunjukkan dalam Gambar 9.

Tabel Identitas Rekening dalam Gambar 9 berisi informasi dasar tentang rekening pengguna, yang digunakan untuk keperluan validasi identitas pada saat pengguna melakukan *log in* untuk memulai transaksi di simulator ATM. Tabel ini memiliki relasi *one-to-many* dengan tabel Transaksi [No. Rekening], yang berisi uraian transaksi yang terkait dengan suatu rekening pengguna. Dengan kata lain, bila terdaftar lima rekening pengguna di tabel Identitas Rekening dengan No. Rekening : 1, 2, …, 5, maka terdapat lima tabel Transaksi, yaitu Transaksi 1, Transaksi 2, … , Transaksi 5, yang masing-masing digunakan untuk mencatat setiap transaksi yang terjadi dengan rekening yang bersesuaian.

Field No. Rekening dalam tabel Identitas Rekening didefinisikan sebagai *primary key*, yaitu *field* unik yang menjadi acuan utama untuk mengakses tabel Transaksi yang bersesuaian. Demikian pula halnya dengan *field* ID Transaksi dalam tabel Transaksi, yang seyogyanya harus unik untuk membedakan satu transaksi dengan transaksi lainnya yang terjadi dalam satu rekening pengguna tertentu. Sebagian besar jenis *field* di kedua tabel didefinisikan sebagai *integer* atau numerik sesuai karakteristik masing-masing *field*. Baik tabel Identitas Rekening maupun tabel Transaksi dapat dikembangkan lebih lanjut, khususnya penambahan jumlah *field*, untuk lebih mencirikan identitas rekening, pengguna dan transaksi.



Gambar 9. Rancangan basis data simulator ATM.

IMPLEMENTASI PIRANTI LUNAK

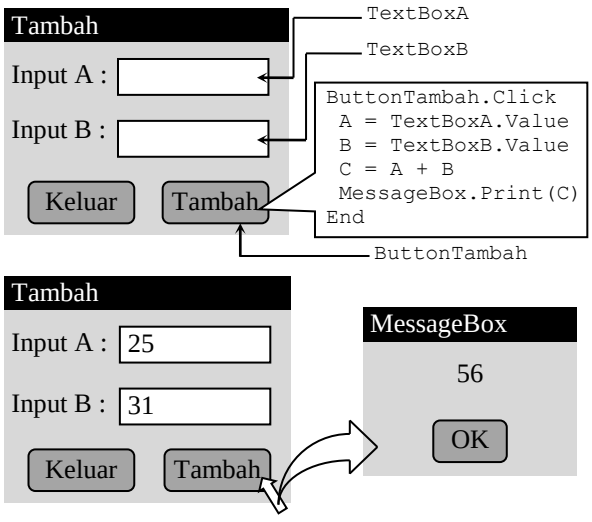
Penyusunan kode program dalam tahap implementasi dapat dilakukan menurut bahasa pemrograman sesuai definisi kebutuhan. Meski demikian, guna mempermudah pembahasan mengenai kegiatan *coding* ini, makalah ini menggunakan beberapa pendekatan berikut :

- Kode program disusun dengan *pseudocode* yang sifatnya universal dan mempermudah transisi dari perancangan algoritma ke kode program;
- Tiga struktur dasar pemrograman, yaitu struktur urut, struktur seleksi dan struktur repetisi, digunakan untuk membangun alur eksekusi sistem;
- Konsep pemrograman berbasis pemrogaman visual dengan *visual component* dan *component event-driven* yang ramah pengguna (*user friendly*).

Pseudocode merupakan representasi pernyataan atau instruksi pemrograman dalam bahasa baku yang sederhana dan memenuhi kriteria algoritma yang benar. Contoh *pseudocode* sederhana untuk empat jenis perhitungan dasar terhadap dua variabel adalah sebagai berikut :

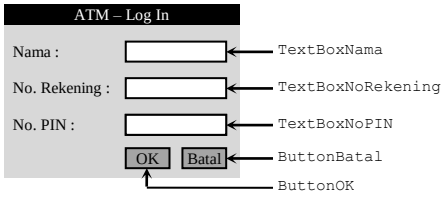
```
Program Hitung_Dasar
   Input A
   Input B
   Tambah = A + B
   Kurang = A – B
   Kali = A x B
   Bagi = A / B
   Print Tambah, Kurang, Kali, Bagi
End
```

Konsep pemrograman visual memungkinkan pemrogram menggunakan komponen visual, seperti tombol (*button*) atau kotak daftar kombinasi (*combo box*), dengan properti dan fungsi tertentu yang dapat diatur guna penyusunan program yang interaktif dan ramah pengguna. Salah satu fitur utama yang melandasi kemudahan ini adalah *component event-driven*, yaitu eksekusi program menurut instruksi yang didefinisikan terhadap 'kejadian' (*event*) dari komponen. Sebagai contoh, komponen *button* memiliki sejumlah *event*, di antaranya *click*, yaitu 'kejadian' yang timbul saat pengguna mengklik *button*. Bila sekumpulan instruksi didefinisikan terhadap *event click* dari *button* ini, maka instruksi-instruksi tersebut akan dieksekusi saat pengguna mengklik *button*, seperti yang ditunjukkan dalam Gambar 10. Contoh ini memperlihatkan serangkaian instruksi yang didefinisikan terhadap *event click* dari komponen `ButtonTambah`, yaitu `ButtonTambah.Click`. Instruksi pertama, `A = TextBoxA.Value`, merujuk kepada properti `Value` atau nilai bilangan yang dimasukkan pengguna ke dalam komponen `TextBoxA`, untuk selanjutnya nilai ini dimasukkan ke variabel `A`. Instruksi keempat, `MessageBox.Print(C)`, mengaktifkan fungsi `Print` atau pencetakan dari `MessageBox`, komponen yang bekerja untuk menampilkan pesan, yang dalam contoh ini adalah nilai variabel `C`.



Gambar 10. Definisi dan eksekusi *event click button*.

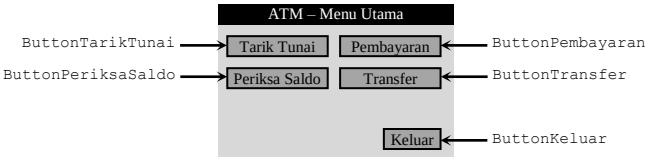
Dalam kasus pengembangan simulator ATM, kode program untuk Menu Log In dapat disusun sebagai berikut :




```
ButtonOK.Click
  Access_Granted = False
  Nama = TextBoxNama.Text
  NoRekening = TextBoxNoRekening.Text
  NoPIN = TextBoxNoPIN.Text
  Tabel = Tabel_Identitas_Rekening
  Open Tabel
  Find Record Where Field_NoRekening = NoRekening In
    Tabel
  If Record Is Found Then
    If Field_Nama = Nama And Field_NoPIN = NoPIN Then
      Access_Granted = True
  End If
  Close Tabel
  If Access_Granted Then
    MenuUtama.Display
  End If
End
```

```
ButtonBatal.Click
  Application.Terminate
End
```

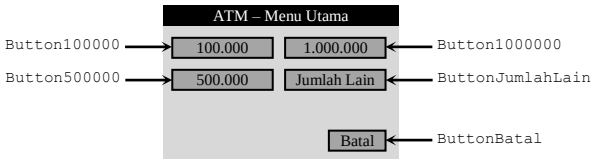
Gambar 11. Kode program untuk Menu Log In simulator ATM.




```
ButtonKeluar.Click
  BackToMenuLogIn
End

ButtonTarikTunai.Click
  MenuTarikTunai.Display
End

ButtonPeriksaSaldo.Click
  MenuPeriksaSaldo.Display
End
```

```
ButtonPembayaran.Click
  MenuPembayaran.Display
End

ButtonTransfer.Click
  MenuTransfer.Display
End
```

Gambar 12. Kode program Menu Utama.




```
Button100000.Click
  Tabel = Tabel_Transaksi
  Open Tabel
  GoTo Last_Record In Tabel
  Saldo = Field_Saldo
  If Saldo < 100000 Then
    MessageBox.Print("Saldo Tidak Cukup !")
    Close Tabel
    Exit
  End If
  Append Record In Tabel
  Field_IDTransaksi = Tabel.RecordCount + 1
  Field_JenisTransaksi = 2
  Field_JumlahUang = 100000
  Field_TujuanTransaksi = 0
  Field_NoRefTransaksi = 0
  Field_WaktuTransaksi = Time(Now)
  Field_Saldo = Saldo - 100000
  Post Record Into Tabel
  Close Tabel
 MessageBox.Print("Transaksi Berhasil !")
 BackToMenuUtama
End
```

```
ButtonJumlahLain.Click
  MenuTarikTunaiJumlahLain.Display
End

ButtonBatal.Click
  BackToMenuUtama
End
```

Gambar 13. Kode program Menu Tarik Tunai.

Kode program dalam Gambar 11 untuk `ButtonBatal.Click` hanya terdiri dari 1 instruksi, yaitu instruksi untuk terminasi aplikasi. Dalam kode program untuk `ButtonOK.Click`, variabel `Access_Granted` menyatakan *flag* diizinkan atau tidaknya pengguna masuk ke Menu Utama berdasarkan validasi identitas rekening. Variabel ini awalnya disetel `False` dan akan berubah menjadi `True` bila `Nama`, `NoRekening` dan `NoPIN` pengguna terdaftar di `Tabel_Identitas_Rekening`. Dengan pola perancangan algoritma yang sama, maka kode program untuk menu-menu representatif lainnya, seperti Menu Utama dan Tarik Tunai, dari simulator ATM dapat disusun seperti yang ditunjukkan dalam Gambar 12 dan 13.

Pengujian kode program dapat dilakukan dengan beberapa cara. Salah satu cara yang efektif adalah melakukan pengujian dua tahap, yaitu :

- Pengujian parsial :
  Pengujian terhadap setiap komponen pembentuk piranti lunak, seperti pengujian terhadap Menu Tarik Tunai, tanpa melibatkan menu lainnya, dari simulator ATM;
- Pengujian integrasi :
  Pengujian terhadap beberapa atau seluruh komponen pembentuk piranti lunak, seperti pengujian terhadap Menu Tarik Tunai dengan mengikutsertakan Menu Log In dan Menu Utama dari simulator ATM.

Pengujian parsial lebih bertujuan menghilangkan *defect* yang terdeteksi dalam tiap komponen selama tahap implementasi. Sedangkan pengujian integrasi lebih cenderung bertujuan menyempurnakan perilaku eksekusi dan batasan operasional dari piranti lunak. Pada kedua jenis pengujian, kegiatan yang umumnya dilakukan adalah mempersiapkan kasus uji dan data uji. Sebagai contoh, pengujian parsial terhadap Menu Log In dapat dilakukan untuk kasus uji berikut :

Tabel 1. Contoh kasus uji parsial Menu Log In.

No	Kasus Uji	Variabel Uji	Hasil	Tindak Lanjut
1	Nama kosong	TextBox Nama	Record tak ditemukan	Tampilkan pesan untuk input Nama
2	NoPIN > 8 digit	TextBox NoPIN	Pencarian ditolak	Batasi input hingga 8 digit

Tindak lanjut untuk kasus uji 1 dalam Tabel 1 dapat dilakukan dengan menambahkan kode program berikut pada `ButtonOK.Click` dari Menu Log In :

```
ButtonOK.Click
  Access_Granted = False
  Nama = TextBoxNama.Text
  If Nama = "" Then
    MessageBox.Print("Input Nama Dengan Benar !")
    Exit
  End If
  :
  :
```

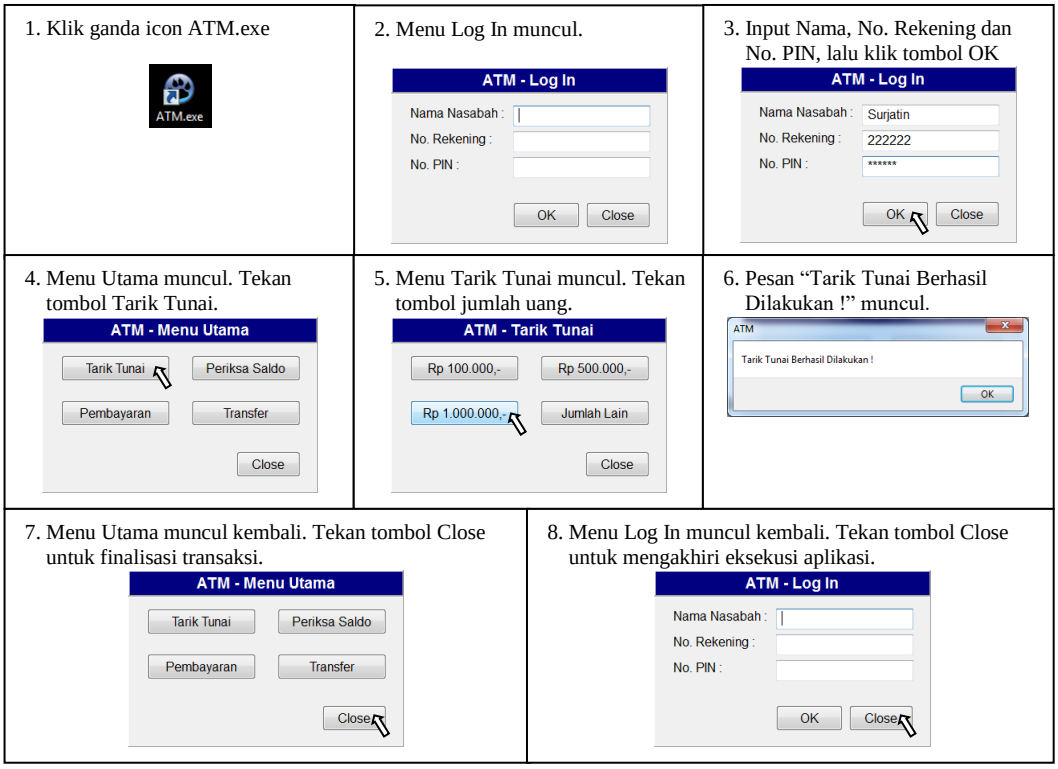
Gambar 14. Revisi kode program untuk `ButtonOK.Click` dalam Menu Log In sistem ATM.

Pola pengujian ini dapat diterapkan pada menu lainnya dari sistem ATM, baik untuk pengujian parsial maupun pengujian integrasi.

OPERASI PIRANTI LUNAK

Operasi piranti lunak dilaksanakan setelah piranti lunak lolos dari tahap implementasi. Pada tahap ini, pihak pengguna mulai menggunakan piranti lunak dan, seandainya terjadi (pada sebagian besar kasus, hampir dapat dipastikan terjadi), melaporkan kesalahan (*defect*) yang terjadi kepada pihak pengembang untuk ditindaklanjuti. Meski piranti lunak sulit dibebaskan secara sempurna dari *defect*, pihak pengembang selayaknya melakukan perencanaan resiko untuk mengantisipasi hal ini, seperti melakukan peninjauan ulang (*review*) secara berkala terhadap produk piranti lunak dan dokumentasi yang terkait, serta berkoordinasi dengan pihak klien dan pengguna untuk mendeteksi *defect* sedini mungkin. Sehingga biaya untuk penyempurnaan piranti lunak dapat ditekan serendah mungkin, ketimbang apabila *defect* ditemukan pada tahap operasi.

Pengoperasian piranti lunak selayaknya juga disertai dengan ketersediaan manual pengoperasian yang memadai. Dokumen seperti ini dapat disusun oleh pihak pengembang dan harus mencakup baik aspek teknis maupun aspek administratif dari pengoperasian piranti lunak. Dalam kasus pengembangan simulator ATM, manual pengoperasian untuk transaksi tarik tunai dapat disusun sebagai berikut :



Gambar 15. Manual pengoperasian simulator ATM untuk transaksi tarik tunai.

Manual ini mengacu pada piranti lunak nyata yang dibangun sesuai hasil analisis, perancangan dan implementasi yang dijelaskan sebelumnya, dengan piranti lunak pemrograman visual *open source* Lazarus versi 0.9.30.4 pada platform sistem operasi Windows 7 64-bit.

SIMPULAN

Pengembangan piranti lunak simulator ATM sebagai sarana pengajaran rekayasa piranti lunak telah dilaksanakan, dan diharapkan dapat memberikan pemahaman yang komprehensif kepada pihak kustomer dalam mendukung upaya untuk menyusun spesifikasi piranti lunak yang tepat sasaran dan sesuai dengan kebutuhan.

DAFTAR PUSTAKA

[1] I.Z. Pane, "Rekayasa Piranti Lunak Dengan Lazarus Pada Kasus ATM," Presentasi Ilmiah Forum Fungsional BBTA3., 18 Oktober 2016.

[2] R.S. Pressman, "Software Engineering, A Practitioner's Approach 6th Edition," McGraw-Hill, 2005.

[3] I. Sommerville, "Software Engineering 8th Edition," Pearson, 2006.