IMPLEMENTASI DOKUMEN *SOFTWARE REQUIREMENT SPESIFICATION* (SRS) UNTUK ANALISIS KEBUTUHAN FUNGSIONAL DAN PENGUJIAN *BLACK-BOX*

Ardiansyah
Program Studi Teknik Informatika Universitas Ahmad Dahlan
Lab. Mobile Technology Innovation Center UAD
Jl. Prof. Dr. Soepomo, Janturan, Yogyakarta
Email: ardiansyah@tif.uad.ac.id

Abstrak

Software Requirement Spesification (SRS) merupakan salah satu batian penting dalam proses analisis kebutuhan pada siklus pengembangan piranti lunak. SRS akan digunakan pada fase-fase berikutnya seperti perancangan sistem, implementasi hingga pelatihan ke pengguna. Akan tetapi walaupun sudah ada dokumen standar SRS yang dikeluarkan IEEE, fakta di lapangan jarang sekali bisa ditemukan di skripsi-skripsi mahasiswa Informatika. Akibatnya spesifikasi kebutuhan fungsional banyak yang ditulis secara ambigu serta hampir tidak ada yang bisa menunjukkan kaitan antara spesifikasi kebutuhan fungsional yang menjadi dasar atas pengujian black box.

Paper ini akan menunjukkan bagaimana penerapan dokumen spesifikasi kebutuhan sistem fungsional yang kemudian bisa menjadi dasar untuk pembuatan dokumen test case pengujian black box. Alat dan bahan yang digunakan pada penelitian ini menggunakan studi kasus pelayanan kasir di mini market, form spesifikasi kebutuhan fungsional, blok diagram dan form pengujian black box.

Penggunaan form spesifikasi kebutuhan sistem (SKS) memudahkan tim penguji piranti lunak (softwre tester) dalam membuat test case setiap fungsi. Bagi pengembang piranti lunak (programmer) dimudahkan dalam membuat setiap fungsi serta untuk pengujian tiap unit (unit testing). Berdasarkan semua kemudahan tersebut terjadi perbedaan yang nyata antara penggunaan SKS yang tidak tepat dan dengan penggunaan SKS yang tepat yaitu pada sisi pendefinisian spesifikasi kebutuhan fungsional yang jelas dan mengurangi ambiguitas.

***Kata kunci**: analisis kebutuhan, black box, fungsional, SRS, testing*

PENDAHULUAN

Software Requirement Spesification (SRS) merupakan salah satu bagian penting dalam proses analisis kebutuhan di dalam siklus pengembangan piranti lunak. SRS kerap digunakan dalam perancangan, implementasi, pemantuan proyek, verifikasi dan validasi hingga pelatihan ke pengguna. Ada dua jenis kebutuhan yaitu kebutuhan *user* dan kebutuhan sistem. Kebutuhan sistem terdiri dari kebutuhan fungsional dan non-fungsional (Sommerville, 2007). Hasil analisis kebutuhan didokumentasikan ke dalam dokumen SRS. Beberapa dokumen SRS yang bisa digunakan misalnya keluaran IEEE Standard 830 (IEEE Computer Society, 1998) dan Volere Requirements Specification Template (Pressman, 2010), (Preece, Rogers, & Sharp, 2002).

Kebutuhan sistem berisi deskripsi terhadap layanan yang disediakan oleh sistem dan batasan-batasan operasionalnya. Kebutuhan ini merefleksikan kebutuhan pengguna/konsumen akan sistem yang kelak bisa membantu menyelesaikan masalah yang dihadapinnya. Proses mencari, menemukan, menggali, menganalisis, mendokumentasikan dan memeriksa layanan serta batasan-batasan inilah yang disebut *requirement engineering* (Sommerville, 2007). Pengujian piranti lunak merupakan proses atau rangkaian proses yang dilakukan untuk menjamin bahwa kode program benar-benar melakukan apa yang sudah dirancang sebelumnya dan tidak melakukan hal-hal yang di luar rancangan (Myers, Badgett, Thomas, & Sandler, 2004).

Walaupun sudah ada dokumen SRS yang menjelaskan tentang kebutuhan fungsional, namun dalam prakteknya kebanyakan skripsi mahasiswa belum bisa menunjukkan secara detail isi kebutuhan fungsional dan juga kaitannya dengan pengujian *black box*. Padahal sudah jelas dinyatakan bahwa pengujian *black box* adalah pengujian fungsional. Sehingga sudah seharusnya bahwa dasar pengujian *black box* berasal dari kebutuhan fungsional yang tertuang pada dokumen SRS. Penelitian ini menunjukkan bagaimana membuat dokumen spesifikasi kebutuhan sistem

fungsional yang kemudian bisa menjadi dasar untuk pembuatan dokumen *test case* pengujian *black box*.

METODE

Alat dan Bahan

Untuk menjelaskan keterkaitan antara dokumen SRS dengan pengujian *black box* dan desain antarmuka diperlukan antara lain studi kasus yaitu aplikasi *point of sales* (PoS) atau kasir di mini market, form Spesifikasi Kebutuhan Sistem (SKS) fungsional, block diagram fungsional, dan form pengujian *black box*.

Proses dan Alur Kerja Aplikasi PoS

Secara umum diagram alir aplikasi kasir di mini market adalah (1) kasir mengambil barang yang diserahkan oleh pembeli, (2) kasir memindai gambar *barcode* di kemasan barang, (3) data barang tampil di layar, (4) kasir mengubah *quantity* pembelian, (5) tampil sub-jumlah dan total belanja, (6) kasir menginputkan jumlah uang pembayaran, (7) tampil jumlah uang kembalian, (8) mencetak struk belanja.

Form Standar Spesifikasi Kebutuhan Sistem (SKS)

Jika menggunakan form standar untuk menspesifikasikan kebutuhan sistem, maka beberapa informasi yang harus dituliskan antara lain kode dan nama fungsi, deskripsi, input, sumber input, output, destinasi output, aksi, kebutuhan, kondisi awal, kondisi akhir, dan pengaruh (Sommerville, 2007), seperti ditunjukkan pada tabel 1.

Tabel 1. Form Standar Spesifikasi Kebutuhan Sistem (SKS)

Kode dan Nama Fungsi	
Fungsi	Nama fungsi
Deskripsi	Penjelasan mengenai fungsi yang akan dispesifikasikan
Input	Deskripsi mengenai input yang akan dikirimkan ke fungsi
Sumber	Tempat asal input berada
Output	Deskripsi mengenai hasil output
Destinasi	Tujuan hasil atau tempat output akan diletakkan
Aksi	Deskripsi mengenai aksi-aksi apa saja yang akan dilakukan sesuai dengan kondisi-kondisi yang telah ditentukan.
Kebutuhan	Entitas lain yang terlibat agar fungsi bisa berjalan
Kondisi Awal	Suatu kondisi awal yang harus dipenuhi agar fungsi yang dijalankan bisa berjalan dengan benar
Kondisi Akhir	Suatu kondisi yang benar setelah fungsi dijalankan
Pengaruh	Dampak yang dihasilkan setelah fungsi dijalankan

Dokumen Pengujian Black Box

Dokumen pengujian *black box* menggunakan *template* dari Williams (2006) yang telah dimodifikasi seperti yang ditunjukkan pada tabel 2. Kolom pertama di tabel akan diisi nama atau nomor fungsi yang telah ditentukan pada form SKS. Kolom kedua berisi daftar semua input yang mungkin diberikan. Kolom ketiga hasil/output yang diharapkan (*expected result*) berdasarkan fungsi yang telah didefinisikan. Kolom keempat berisi hasil/output yang terjadi (*actual result*) yaitu "Lulus" kalau hasilnya sesuai yang diharapkan dan "Gagal" kalau tidak sesuai yang diharapkan.

Tabel 2. Dokumen Template Pengujian *Black Box*

Fungsi	**Input**	**Expected Result (Output)**	**Actual Result**

HASIL DAN PEMBAHASAN

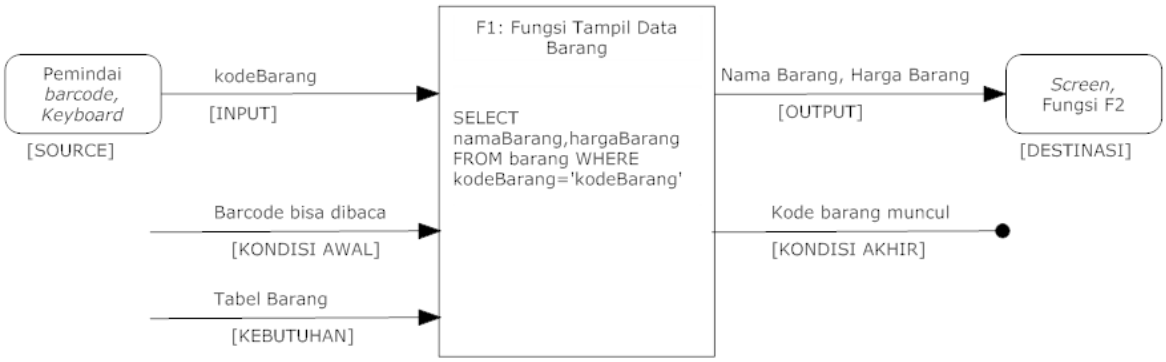
Analisis Kebutuhan Fungsional

Berdasarkan diagram alir aplikasi PoS sebelumnya, selanjutnya bisa dispesifikasikan kebutuhan fungsional dalam bentuk form standar Spesifikasi Kebutuhan Sistem (SKS). SKS yang pertama adalah fungsi untuk menampilkan detail barang seperti yang dijelaskan pada tabel 2.

Tabel 2. SKS Fungsi Tampil Detail Barang

F1: Tampil Detail Barang/Kasir	
Fungsi	Menampilkan detail data barang
Deskripsi	Menampilkan detail data barang yang dibeli oleh pembeli. Sehingga nanti bisa digunakan untuk menghitung sub-jumlah dan total belanja.
Input	Kode Barang
Sumber	Pemindai *barcode*
Output	Detail barang: nama dan harga.
Destinasi	*Screen*, Fungsi F2
Aksi	Ketika pemindai berhasil mengenali *barcode*, maka akan menghasilkan kode barang. Selanjutnya kode barang ini akan dikirimkan ke perintah SQL untuk menampilkan detail barang.
Kebutuhan	Tidak ada
Kondisi Awal	Barcode harus dalam kondisi baik sehingga bisa dibaca pemindai
Kondisi Akhir	Keluar kode barang
Pengaruh	Tidak ada

Bila divisualisasikan dalam bentuk blok diagram seperti di gambar 1, maka akan tampak seperti gambar 1. Pertama-tama pemindai memindai *barcode* yang ada di kemasan barang. Supaya bisa mengeluarkan kode, maka *barcode* harus dalam kondisi baik sehingga bisa dibaca/pindai. Namun bila suatu waktu pemindai tidak berhasil membaca *barcode*, kasir tetap bisa menginputkan kode barang lewat *kebyoard.* Kode barang yang berhasil dibaca atau diinputkan lewat *keyboard* akan dikirimkan ke fungsi F1 dengan tujuan untuk menampilkan data barang. Secara teknis, fungsi ini berisi perintah SQL untuk menampilkan detail barang, sehingga yang dibutuhkan adalah kode barang dan tabel barang. Jika *query* berhasil dijalankan maka otomatis akan muncul detail data barang ke layar dan juga sebagai sumber untuk menjalankan fungsi F2.



Gambar 1. Blok Diagram Fungsi F1 Tampil Data Barang

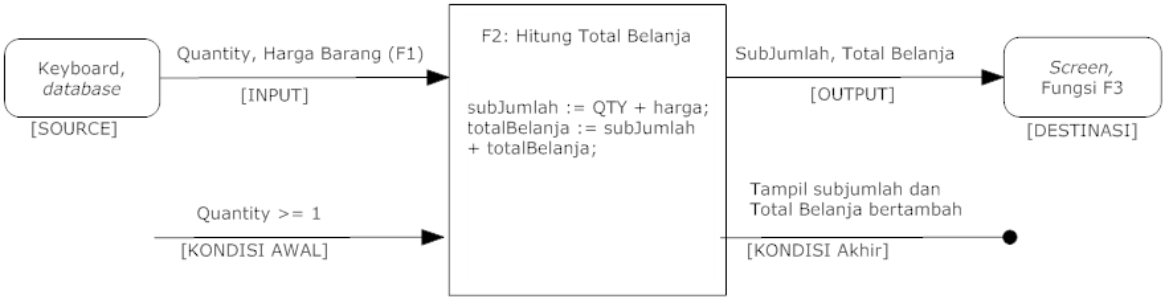
Setelah memanggil fungsi F1, maka dilanjutkan untuk menjalankan fungsi F2 yaitu menghitung total belanja. SKS fungsi ini ditunjukkan pada tabel 2.

Tabel 2. SKS Fungsi Hitung Total Belanja

F2: Hitung Total Belanja/Kasir	
Fungsi	Menghitung total belanja
Deskripsi	Setelah F1 berfungsi maka bisa dilakukan penghitungan sub-jumlah dan total belanja yang tampil secara urut di layar monitor.
Input	Quantity, Harga Barang

Sumber	Quantity diinputkan lewat keyboard, Harga Barang diambil dari *database* sesuai fungsi F1.
Output	Total Belanja
Destinasi	*Screen*, Fungsi hitung uang kembalian
Aksi	User menginputkan *quantity* barang minimal 1 pcs/unit sehingga bisa dihitung sub-jumlahnya. Setelah sub-jumlah dihitung maka otomatis bisa menghitung total belanja.
Kebutuhan	Tidak ada
Kondisi Awal	Quantity >= 1
Kondisi Akhir	Sub Jumlah tampil dan Total Belanja bertambah
Pengaruh	Tidak ada

Fungsi menghitung total belanja membutuhkan input berupa *quantity* yang berasal dari *keyboard* kasir dan harga barang yang berasal dari fungsi F1. Secara *default* kondisi awal *quantity* belanja harus minimal 1 *pcs*/unit agar bisa menghitung sub-jumlah terlebih dahulu. Setiap kali barang yang dibeli dihitung sub-jumlahnya otomatis juga menambah total belanja yang tampil di layar dan juga sebagai sumber untuk menjalankan fungsi F3 (perhatikan gambar 2).



Gambar 2. Blok Diagram Fungsi F2 Hitung Total Belanja

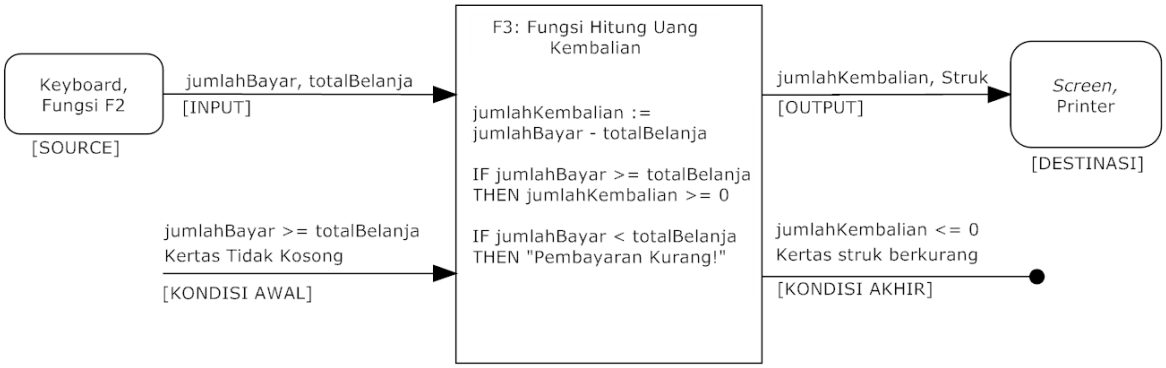
Tahap berikutnya setelah semua barang dihitung dan tahu berapa total belanja adalah menghitung jumlah uang kembalian sekaligus mencetak struk belanja. Adapun spesifikasi kebutuhan sistem untuk fungsi ini seperti yang digambarkan pada tabel 3.

Tabel 3. SKS Fungsi Hitung Uang Kembalian

F3: Hitung Uang Kembalian/Kasir	
Fungsi	Menghitung uang kembalian
Deskripsi	Fungsi ini adalah bagian dari tahap akhir proses PoS/kasir. Fungsi ini akan menghitung jumlah uang kembalian berdasarkan jumlah uang yang dibayarkan pembeli dengan total belanja dan sekaligus mencetak struk belanja.
Input	Jumlah Bayar, Total Belanja
Sumber	Jumlah Bayar diinput dari keyboard Total Belanja berasal dari Fungsi F2
Output	Jumlah Uang Kembalian, Struk Belanja
Destinasi	*Screen*, Printer
Aksi	IF Jumlah Bayar ≥ Total Belanja THEN Jumlah Uang Kembalian ≥ 0 ELSE “Pembayaran Kurang”
Kebutuhan	Tidak ada
Kondisi Awal	Jumlah Uang Kembalian ≥ Total Belanja AND Kertas NOT EMPTY
Kondisi Akhir	Keluar jumlah uang kembalian, Kertas struk berkurang
Pengaruh	Tidak ada

Untuk menghitung uang kembalian, maka syaratnya uang yang dibayar pembeli harus sama atau lebih dari total belanja. Kertas untuk mencetak struk juga harus dipastikan terisi atau tidak dalam keadaan kosong/habis. Kasir menginputkan jumlah uang pembayaran dari *keyboard* dan

total belanja sudah dihitung melalui fungsi F2. Jika uang pembayaran lebih besar dari total belanja, maka uang kembalian akan lebih dari nol, jika uang pembayaran sama dengan total belanja, maka uang kembalian adalah nol, sedangkan jika uang pembayaran kurang dari total belanja, maka akan menampilkan pesan bahwa uang pembayaran masih kurang. Semua proses fungsi ini akan dikirimkan ke layar dan juga ke *printer* untuk mencetak struk.



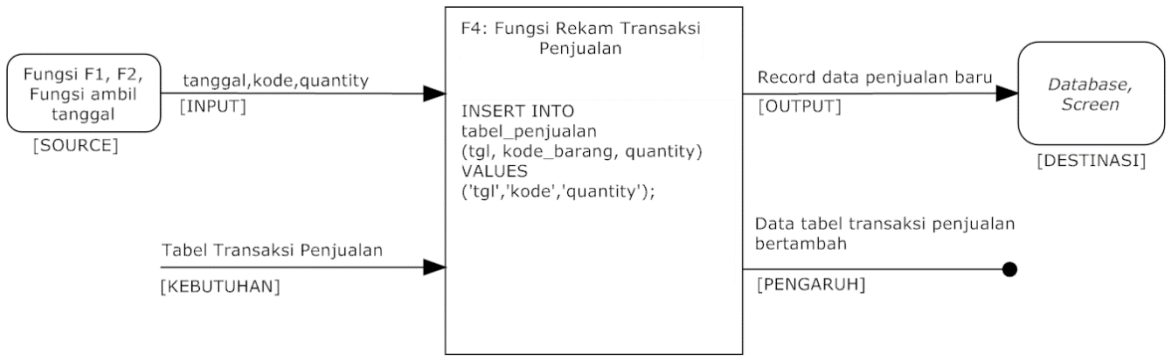
Gambar 3. Blok Diagram Fungsi F3 Hitung Uang Kembalian

Setelah semua proses di sisi kasir dijalankan, maka selanjutnya di sisi admin atau *backend* akan melakukan proses perekaman transaksi penjualan. Perekaman ini dibutuhkan karena sebagai bagian dari fungsional sistem di sisi *backend*. Tabel 4 memperlihatkan spesifikasi kebutuhan sistem untuk fungsi F4 yaitu rekam transaksi penjualan.

Tabel 4. SKS Fungsi Rekam Transaksi Penjualan

F4: Rekam Transaksi Penjualan/Backend	
Fungsi	Merekam data transaksi penjualan
Deskripsi	Setelah fungsi F1, F2, F3 dijalankan maka fungsi F4 ini bisa melakukan perekaman data transaksi penjualan ke tabel.
Input	Tanggal, Kode Barang, Quantity
Sumber	Tanggal diambil dari fungsi tersendiri, Kode Barang berasal dari fungsi F1, Quantity dari fungsi F2.
Output	Record data penjualan baru
Destinasi	*Database*, *Screen*
Aksi	Tidak ada
Kebutuhan	Tabel transaksi penjualan
Kondisi Awal	Tidak ada
Kondisi Akhir	Tidak ada
Pengaruh	Data tabel transaksi penjualan bertambah

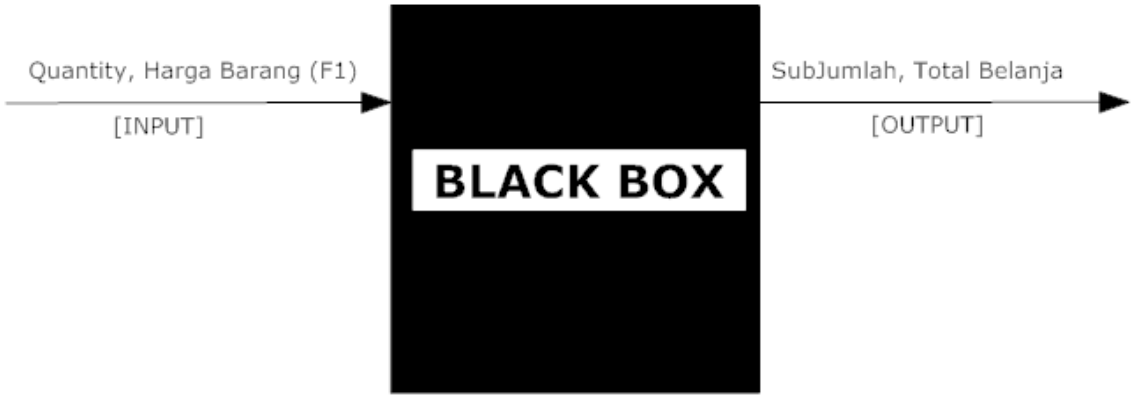
Input fungsi rekam transaksi penjualan ini adalah tanggal yang berasal dari fungsi ambil tanggal, kode barang yang berasal dari fungsi F1 dan *quantity* yang berasal dari fungsi F2. Dibutuhkan tabel transaksi penjualan agar bisa merekam data ke tabel tersebut. Fungsi rekam transaksi terdiri dari perintah SQL sederhana untuk menyisipkan satu record data baru (lihat gambar 4). Tujuan atau destinasi output fungsi F4 ini adalah ke layar dan ke *database*. Dampak yang dihasilkan dari fungsi ini adalah data tabel penjualan akan bertambah.



Gambar 4. Blok Diagram Fungsi F4 Rekam Transaksi Penjualan

Pengujian Black Box

Pengujian *black box* atau disebut juga pengujian fungsional adalah pengujian yang fokus pada input dan output untuk menentukan dan menilai apakah program telah melakukan apa yang seharusnya dilakukan sesuai dengan kebutuhan fungsional. Pada pengujian ini, fungsi atau program dianggap sebagai "kotak hitam" yang isi dan struktunya tidak diketahui oleh penguji (*tester*) seperti yang terlihat pada gambar 5.



Gambar 5. Ilustrasi Pengujian *Black Box*

Salah satu prinsip penting pengujian *black box* adalah tidak hanya mencoba input yang valid saja, tetapi melainkan semua kemungkinan input yang tersedia (Myers et al., 2004). Berdasarkan pada hasil spesifikasi kebutuhan sistem fungsi F1, F2, F3, F4 dan F5, maka didapatkan *test case* pengujian *black box* yang ditunjukkan pada tabel 5.

Semua kemungkinan input di setiap fungsi sudah teridentifikasi dan harus diujicoba satu per satu dengan output atau *exptected result* yang sudah definisikasn. Sehingga bila output sesuai dengan yang diharapkan maka *actual result* bisa dikosongkan. Sedangkan bila tidak sesuai maka *actual result* diisi dengan kondisi yang sesungguhnya. Biasanya kalau *actual result* diisi, artinya fungsi masih belum berjalan dengan benar/tepat atau dengan kata lain ada kesalahan yang terjadi. Hasil pengujian *black box* ini akan menjadi dasar bagi pemrogram untuk memperbaiki kode programnya.

Tabel 5. Format *Test Case* untuk Perencanaan Pengujian Black Box

Fungsi	Input	Expected Result (Output)	Actual Result
F1	Kode Barang (*barcode*)	Tampil Nama Barang, Harga Barang	Lulus
	Kode Barang (*keyboard*)	Tampil Nama Barang, Harga Barang	Lulus
F2	Quantity ≤ 0	"Keyboard tidak merespon"	Gagal
	Quantity > 0	Tampil hasil sub-jumlah	Lulus
		Tampil total belanja	Lulus
F3	Jumlah Bayar > Total Belanja	Tampil hasil kembalian lebih dari 0	Lulus

	Jumlah Bayar = Total Belanja	Tampil hasil kembalian 0	Lulus
	Jumlah Bayar < Total Belanja	Keyboard tidak merespon	Lulus
F4	Tanggal, Kode Barang, Quantity	Record di tabel penjualan bertambah	Lulus

SIMPULAN

Dengan menggunakan form spesifikasi kebutuhan sistem (SKS) memudahkan tim penguji piranti lunak (*software tester*) dalam membuat *test case* setiap fungsi. Bagi pengembang piranti lunak (*programmer*) dimudahkan dalam membuat setiap fungsi serta untuk pengujian tiap unit (*unit testing*). Berdasarkan semua kemudahan tersebut terjadi perbedaan yang nyata antara penggunaan SKS yang tidak tepat dan dengan penggunaan SKS yang tepat yaitu pada sisi pendefinisian spesifikasi kebutuhan fungsional yang jelas dan mengurangi ambiguitas.

DAFTAR PUSTAKA

IEEE Computer Society. (1998). IEEE Recommended Practice for Software Requirements Specifications. Std 830-1998. New York, New York: IEEE.

Myers, G. J., Badgett, T., Thomas, T. M., & Sandler, C. (2004). *The Art of Software Testing* (2nd ed.). New Jersey: John Wiley & Sons.

Preece, J., Rogers, Y., & Sharp, H. (2002). *Design Interaction: Beyond Human-Computer Interaction*. (G. R. Mutton & P. Crockett, Eds.) (I.). New York: John Wiley & Sons.

Pressman, R. S. (2010). *Software Engineering*. (F. aye M. Schilling, Ed.) (7th ed.). New York: McGraw Hill.

Sommerville, I. (2007). *Software Engineering* (8th ed.). Pearson Education Limited.

Williams, L. (2006). Testing Overview and Black-Box Testing Techniques. Retrieved from http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf