Pembangunan *Class Library* Pada Domain Aplikasi *Personal Finance*

Argi Sugiyarsa[1]

[1] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
Email : argisugiyarsa@*outlook*.com[1]

ABSTRAK

Pada dasarnya setiap individu membutuhkan kemampuan untuk mengelola sumber daya keuangan baik dengan cara manual atau dengan menggunakan aplikasi *mobile*. Kebutuhan pengguna dalam pengelolaan sumber daya keuangan menjadi pemicu pengembang perangkat lunak dalam membangun atau mengembangkan perangkat lunak *personal finance*. Berdasarkan hasil wawancara dengan beberapa *programmer* diperoleh fakta bahwa dalam membangun perangkat lunak dalam suatu domain selalu dimulai dari awal dikarenakan pengembang lebih fokus kepada proses implementasi daripada analisisnya. Solusi yang dapat diterapkan dari permasalahan di atas yaitu dengan membangun *class library*.

Class library adalah kumpulan dari kelas yang sudah ditulis sebelumnya yang dapat digunakan kembali oleh *programmer* dalam mengembangkan perangkat lunak. Dalam membangun sebuah *class library* diperlukan beberapa tahapan yang harus dilakukan, tahap pertama yang dilakukan adalah analisis domain untuk mendapatkan fungsional dasar dari aplikasi yang dianalisis, tahap selanjutnya adalah analisis *frozen spot* dan *hot spot* untuk menentukan persamaan fungsional dari aplikasi yang dianalisis. Tahap berikutnya adalah perancangan kelas yang didasarkan pada hot spot yang telah didapatkan, kelas yang terbentuk kemudian dikelompokan pada tahap perancangan *package*.

Class library yang sudah dirancang kemudian diimplementasikan pada kode program dan dilakukan pengujian. Pengujian yang dilakukan untuk menguji *class library* adalah dengan *unit testing*, *integration testing*, dan *acceptance testing*. Dari hasil pengujian yang telah dilakukan dapat disimpulkan bahwa pembangunan *class library* pada domain kasus *personal finance* dapat mempermudah *programmer* dalam membangun perangkat lunak *personal finance*.

Kata Kunci*: class library, personal finance.*

1. PENDAHULUAN

Pada dasarnya setiap individu membutuhkan pengetahuan dan kemampuan untuk mengelola sumber daya keuangan demi kesejahteraan hidupnya [1]. Pada saat ini pengelolaan sumber daya keuangan sudah menggunakan teknologi dalam penggunaannya seperti perangkat lunak mobile [2]. Menurut data *Mobile Marketing Statistics* 2015 diperoleh informasi bahwa jumlah pengguna perangkat lunak mobile secara global adalah 1900 juta. Setiap hari bermunculan perangkat lunak *personal finance* [3]. Dari hasil observasi pada situs penyedia perangkat lunak *mobile* yaitu *windows store* ditemukan lebih dari 1000 perangkat lunak yang bergerak pada domain *personal finance*, menurut analisis yang dilakukan IDC (*International Data Corporation*) menyatakan bahwa *market share windows phone* akan meningkat sebesar 10% pada tahun 2017 [4].

Membangun atau mengembangkan perangkat lunak pada suatu domain merupakan hal yang kompleks dan memerlukan proses yang panjang [5]. Berdasarkan hasil wawancara dengan beberapa programmer diperoleh informasi bahwa dalam pembangunan perangkat lunak pada suatu domain selalu dilakukan dari awal dikarenakan pengembang lebih fokus pada implementasi dari pada proses analisisnya. Dengan demikian penggunaan kembali struktur kode yang pernah ada sulit untuk dilakukan dalam proses pengembangan atau pembangunan perangkat lunak kedepannya. Berdasarkan permasalahan yang telah dijelaskan diatas maka akan dibangun class library pada domain kasus *personal finance*.

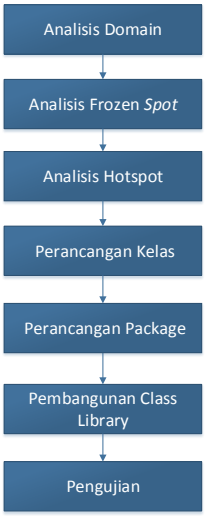
1.1 Class Library

Class library merupakan kumpulan kelas-kelas yang sudah ditulis sebelumnya yang dapat digunakan kembali oleh programmer dalam pembangunan perangkat lunak [2].

1.2 Metode Pembangunan Class lIbrary

Metode pembangunan *class library* tersusun oleh beberapa langkah diantaranya adalah analisis domain perangkat lunak, analisis *frozent spot*, analisis *hot spot*, perancangan kelas, perancangan

package, pembangunan *class library*, pengujian *class library* [3].



Gambar 1 Metode Pembanguan Class Library

1.3 Domain Kasus Personal Finance

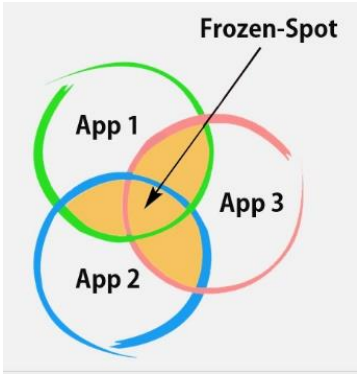
Personal finance management adalah ilmu yang mempelajari bagaimana individu atau keluarga mengelola sumber daya keuangannya [4]. Yang harus dilakukan oleh individu atau keluarga dalam menajemen keuangan pribadi adalah dengan mencatat aliran uang yang keluar masuk dari sumber daya keuangannya, proses mencatat aliran uang tersebut bisa dilakukan banyak cara seperti mencatat pada sebuah buku keuangan ataupun dengan mengingat aliran uang yang terjadi dalam periode waktu tertentu, akan tetapi pada era teknologi saat ini tidak relevan apabila tidak menggunakan komputer dalam manejemen keuangan pribadi [5].

1.4 Analisis Domain

Analisis domain mempresentasikan pendekatan sistematik untuk mengidentifikasi lingkup, fitur-fitur, dan titik-titik variasi pada suatu domain [6]. Ananlisis domain dalam pembangunan *class library* merupakan proses pengelompokan perangkat lunak yang dilihat dari kesamaan fungsionalitasnya.

1.5 Frozen Spot

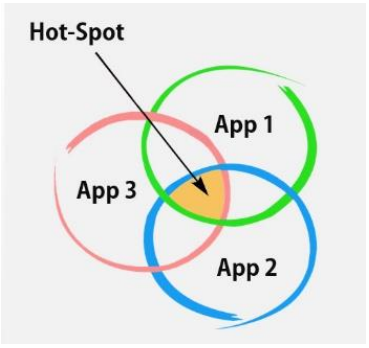
Frozen spot merupakan fungsionalitas yang identik ada pada perangkat lunak pada sutu domain tertentu [3]. *Frozen spot* bisa dikatakan sebagai bahan dasar suatu fungsional yang nantinya dapat diuraikan menajadi suatu *hot spot*, *frozen spot* dalam suatu domain perangkat lunak biasanya berupa bagian dari fungsional maupun keseluruhan fungsional yang ada pada beberapa perangkat lunak dengan domain yang sama.



Gambar 2 Frozen Spot

1.6 *Hot Spot*

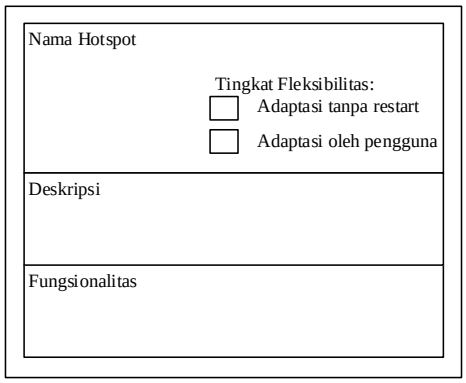
Pada dasarnya *hot spot* merupakan fungsional yang ada pada masing-masing perangkat lunak dalam suatu domain kasus yang sama. *Hot spot* didapatkan melalui penguraian dari analisis *frozen spot*, penguraian *hot spot* didapat melalui dengan dua cara yaitu metode *whitebox* dan metode *blackbox*, metode *whitebox* merupakan pendefinisian *hot spot* berdasarkan struktur kode yang sudah ada, dan metode *blackbox* merupakan pendefinisian *hot spot* melalui penguraian dari analisis *frozen spot* [7].



Gambar 3 Hot Spot

1.7 Kartu *Hot Spot*

Kartu *hot spot* digunakan untuk mengetahui tingkah laku dari suatu fungsionalitas yang akan diimplementasikan. Isi dari sebuah kartu *hot spot* antara lain nama *hot spot*, fleksibilitas, deskripsi, fungsionalitas.



Gambar 4 Kartu Hot Spot

1.8 Pengujian

Untuk melakukan pengujian *class library* dapat dilakukan beberapa cara pengujian seperti *unit testing, integration testing,* dan *acceptance testing*.

1. *Unit testing*

Unit Tesing adalah sebuah kode yang ditulis oleh pengembang yang digunakan untuk menguji bagian kecil atau area spesifik dari suatu fungsionalitas kode yang akan dites [7].

2. *Integration testing*

Integration testing merupakan pengujian setiap unit logis [8]. Tahapan integration testing berupa input modul-modul yang telah diuji pada tahap unit testing, diproses kedalam sub integration testing (interaction testing, UI testing, dll).

3. *Acceptance testing*

Acceptance testing merupakan pengujian formal yang berhubungan dengan kebutuhan pengguna, persyaratan, dan proses bisnis yang dilakukan untuk menentukan apakah suatu sistem memenuhi kriteria penerimaan atau tidak, dan pengguna atau badan lain yang berwenang sangat mungkin untuk menentukan untuk menerima sistem atau tidak [9].

1. ISI PENLITIAN

2.1 Analisis Domain Perangkat lunak

Analisis domain perangkat lunak yang dilakukan adalah dengan melakukan analisisi terhadap tiga perangkat lunak yang bergerak pada domain *personal finance.* Perangkat lunak *personal finance* dipilih berdasarkan *rating* dari situs penyedia perangkat lunak *windows store*. Detail perangkat lunak *personal finance* dapat dilihat pada Tabel 1.

Tabel 1 Analisis Domain Perangkat Lunak

Nama Aplikasi	Rating
***IMoney* V 1.2.0**	**4 *Star***
***myFinance* V 1.5.1.0**	**4 *Star***
***Toshl Finance* V 1.8.9.3**	**4 *Star***

2.2 Analisis Frozen Spot

Dari analisis domain perangkat lunak yang telah dilakukan kemudian dilakukan analisis *frozen spot* yaitu dengan mengelompokan dan mendefinisikan fungsionalitas yang ada pada setiap perangkat lunak yang di analisis. Hasil analisis *frozen spot* dapat dilihat pada Tabel 2.

Tabel 2 Analisis Frozen Spot

Fitur	Personal Finance			Ket
	App 1	App 2	App 3	
Membuat akun / dompet	✔	✔	✔	Setiap aplikasi memiliki fungsi menambah akun dengan isian parameter yang berbeda-beda.
Penyajian list jensi mata uang	✔	✔	✔	Setiap aplikasi memiliki fungsi penyajian jenis mata uang
Menambah Transaksi	✔	✔	✔	Setiap aplikasi memiliki fungsi penambahan transaksi
Penyajian list kategori	✔	✔	✔	Setiap aplikasi memiliki fungsi penyajian list categori
Menambah kategori transaksi				Setiap aplikasi memiliki fungsi

				menambah kategoru transaksi
Penyajian detail transaksi	✔	✔	✔	Setiap aplikasi memiliki Fungsi penyajian detail transaksi
Menambah jenis mata uang		✔	✔	Fungsi menambah jenis mata uang hanya ada pada aplikasi myFinance dan toshl finance
Menambah schedule	✔			Fungsi menambah schedule hanya ada pada aplikasi I-Money
Report	✔	✔	✔	Fungsi penyajian laporan transksi ada pada setiap aplikasi
Login			✔	Fungsi login hanya ada pada aplikasi toshl finance
Sign Up / Menamah user			✔	Fungsi menambah user hanya ada pada aplikasi toshl finance

2.3 Analisis Hot Spot

Setelah analisis *frozen spot* kemudian dilakukan analisis *hot spot*, kandidat *hot spot* didapatkan dari pendifinisian *frozen spot*. Hasil analisis *hot spot* dapat dilihat pada Tabel 3.

Tabel 3 Analisis Hot Spot

No	Hot Spot
1	Penambahan data transkasi
2	Penyajian data transaksi
3	Penyajian detail transaksi
4	Penyajian laporan transaksi
5	Penambahan kategori transaksi
6	Penambahan data akun
7	Tambah jenis mata uang
8	Penyajian jens mata uang
9	Penyajian laporan data transaksi
10	Login
11	Sign Up

1. Membuat Akun Pengguna

Hot spot membuat akun pengguna merupakan *hot spot* yang berasal dari fungsional menambahkan akun yang sudah didefinisikan dalam analisis frozen spot sebelumnya, fungsi menambah akun meliputi beberapa atribut antara lain nama akun, mata uang, dan jumlah uang.

2. Penyajian data transaksi

Hot spot penyajian data transaksi merupakan *hot spot* yang berasal dari fungsional peyajian data transaksi yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya, penyajian data transaksi yang ditampilkan memberikan berbagai informasi yang berkaitan dengan data transaksi.

3. Penyajian detail transaksi

Hot spot penyajian detail transaksi merupakan *hot spot* yang berasal dari fungsional melihat detail transaksi yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya, detail transaksi menyediakan informasi transaksi secara lengkap, *hot spot* detail transaksi memiliki beberapa atribut seperti tanggal transaksi, kategori transaksi, jenis transaksi, jumlah biaya, sisa saldo, dan deskripsi.

4. Penambahan data transaksi

Hot spot tambah transaksi merupakan *hot spot* yang berasal dari fungsional penambahan data transaksi yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya, transaksi yang dilakukan berupa transaksi pendapatan maupun pengeluaran, fungsi menambah transaksi memiliki beberapa atribut yaitu jumlah, akun, kategori transaksi, dan tipe transaksi.

5. Penambahan kategori transaksi

Hot spot tambah kategori transaksi merupakan *hot spot* yang berasal dari fungsional penambahan kategori transaksi yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya, fungsi menambah kategori transaksi meliputi beberapa atribut antara lain nama kategori.

6. Penyajian kategori transaksi

Hot spot penyajian kategori transaksi merupakan *hot spot* dari fungsional penyajian data kategori transaksi yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya, penyajian data kategori transaksi memberikan informasi kategori transaksi yang dapat dipilih pengguna ketika melakukan transaksi.

7. Penyajian laporan data transaksi

Hot spot Penyajian laporan merupakan *hot spot* yang berasal dari fungsional peyajian laporan data transaksi yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya, penyajian laporan data transaksi dapat memberikan berbagai informasi transaksi yang disajikan dalam bentuk diagram.

8. Penambahan jenis mata uang

Hot spot tambah jenis mata uang merupakan *hot spot* yang berasal dari fungsional penambahan jenis mata uang yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya, fungsi menambah jenis mata uang memiliki atribut nama mata uang dan deskripsinya.

9. Penyajian jenis mata uang

Hot spot penyajian jenis mata uang merupakan *hot spot* yang berasal dari fungsional menyajikan data jenis mata uang yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya.

10. Login

Hot spot login merupakan *hot spot* yang berasal dari fungsional *login* yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya. Fungsi login akan mengelola data pengguna perangkat lunak agar pengguna dapat masuk kedalam perangkat lunak sebagai pengguna yang sudah terdaftar.

11. Sign Up

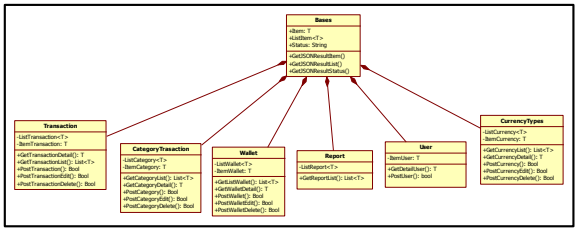
Hot spot sign up merupakan *hot spot* yang berasal dari fungsional penambahan user yang sudah didefinisikan dalam analisis *frozen spot* sebelumnya. Fungsi ini akan mengelola data pengguna agar pengguna dapat terdaftar sebagai pengguna perangkat lunak.

2.4 Perancangan Kelas

Pada bagian ini akan dilakukan perancangan kelas dari setiap *hot spot* yang sudah didefinisikan. Setiap *hot spot* akan dikelompokan berdasarkan data yang dikelolanya. Hasil perancangan kelas dapat dilihat pada Tabel 4 dan pada Gambar 5.

Tabel 4 Perancangan Kelas

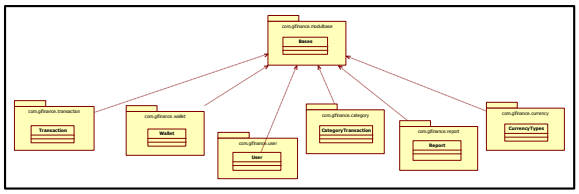
Hot Spot	*Nama Class*
Penambahan data transaksi	*Transaction*
Penyajian data transaksi	
Penyajian detail transaksi	
Penyajian laporan transaksi	
Penambahan kategori transaksi	*CategoryTransaction*
Penyajian kategori transaksi	
Tambah data akun / *Wallet*	*Wallet*
Tambah jenis mata uang	*CurrencyTypes*
Penyajian jenis mata uang	
Penyajian laporan data transaksi	*Report*
Login	*User*
Sign Up	
-	*Bases*



Gambar 5 Diagram kelas *personal finance class library*

2.5 Perancangan *Package*

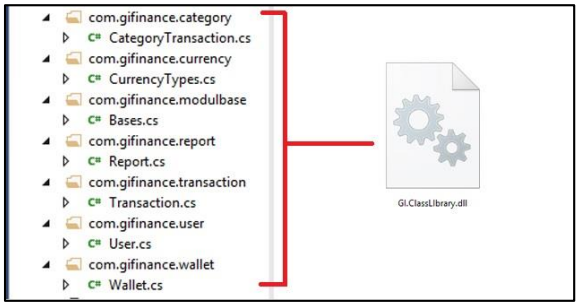
Perancangan *package* merupakan proses pengelompokan kelas-kelas yang berkaitan atau sejenis, hal ini dilakukan agar memudahkan pemahaman struktur kelas dan pemanggilannya. Perancangan *package* pada *class library personal finance* dapat dilihat pada Gambar 6.



Gambar 6 Perancangan package

2.6 Implementasi Class Library

Class library yang diimplementasikan berupa kelas-kelas yang sudah dirancang sebelumnya yang kemudian akan dikompilasi kedalam bentuk file *Dynamic Link Library* (dll).



Gambar 7 Implementasi Class Library

2.7 Pengujian Class Library

Pengujian class library terdiri dari beberapa tahapan yang disusun rencana dan skenario pengujiannya, kemudian hasil pengujian akan di evaluasi.

2.7.1 Rencana Pengujian Class Library

Rencana pengujian class libray pada domain aplikasi personal finance dilakukan dengan beberapa tahapan antara lain:

1. Pengujian *Unit Class Library*
2. Pengujian *Integration Class Library*
3. Pengujian *Acceptance Class Library*
4. Evaluasi Pengujian

2.7.2 Skenario Pengujian Class Library

Pada bagian skenario pengujian ini terbagi menjadi tiga jenis, yaitu skenario pengujian unit, skenario pengujian integration, dan skenario pengujian acceptance. Skenario pengujian class library dapat dilihat pada Tabel 5 dan Tabel 6.

Tabel 5 Skenario Unit Testing

Kelas Uji	Poin Pengujian	Jenis Pengujian
Transaction	Uji metode *GetList Transaction*	Unit Testing
	Uji metode *GetDetailTransaction*	Unit Testing
	Uji metode PostTransaction	Unit Testing
	Uji metode PostTransactionEdit	Unit Testing
	Uji metode PostTransactionDelete	Unit Testing
Wallet	Uji metode *GetList Wallet*	Unit Testing
	Uji metode *GetDetailWallet*	Unit Testing
	Uji metode Post*Wallet*	Unit Testing
	Uji metode *PostWalletEdit*	Unit Testing
	Uji metode *PostWalletDelete*	Unit Testing
CategoryTransaction	Uji metode *GetCategoryList*	Unit Testing
	Uji metode *GetCategoryDetail*	Unit Testing
	Uji metode *PostCategory*	Unit Testing
	Uji metode *PostCategoryEdit*	Unit Testing
	Uji metode *PostCategoryDelete*	Unit Testing
CurrencyTypes	Uji metode *GetListCurrency*	Unit Testing
	Uji metode *GetDetailCurrency*	Unit Testing
	Uji metode *PostCurrency*	Unit Testing
	Uji metode *PostCurrencyEdit*	Unit Testing
	Uji metode *PostCurrencyDelete*	Unit Testing
User	Uji metode	Unit Testing

	GetDetailUser	
	Uji metode *PostUser*	Unit Testing
Report	Uji metode *GetReportList*	Unit Testing

Tabel 6 Skenario Integration Testing

Item Uji	Jenis Pengujian
Detail Transaction	*Integration Testing*
Get Transaction	*Integration Testing*
Tambah data Transaction	*Integration Testing*
Edit data Transaction	*Integration Testing*
Delete data Transaction	*Integration Testing*
Detail Wallet	*Integration Testing*
Get Wallet	*Integration Testing*
Tambah data wallet	*Integration Testing*
Edit data wallet	*Integration Testing*
Delete data wallet	*Integration Testing*
Detail User	*Integration Testing*
Tambah data user	*Integration Testing*
Detail Currency	*Integration Testing*
Get Currency	*Integration Testing*
Tambah data Currency	*Integration Testing*
Edit data Currency	*Integration Testing*
Delete data Currency	*Integration Testing*
Detail Category	*Integration Testing*
Get Category	*Integration Testing*
Tambah data Category	*Integration Testing*
Edit data Category	*Integration Testing*
Delete data Category	*Integration Testing*
Penyajian Laporan	*Integration Testing*

Skenario pengujian *acceptance* yang akan dilakukan yaitu berupa skenario wawancara.

2. KESIMPULAN

Dari penelitian yang telah dilakukan maka dapat disimpulkan bahwa *class library personal finance* yang telah dibangun dapat mempermudah programmer dalam membangun perangkat lunak *personal finance*. *Class library personal finance* telah menyediakan fungsionalitas utama yang dibutuhkan dalam pembangunan perangkat lunak *personal finance*.

Adapun saran yang dapat diberikan yaitu dapat dilakukan penambahan penanganan *error* atau *exception* dan informasi niai tukar mata uang yang tidak dilakukan dalam penelitian ini.

3. DAFTAR PUSATAKA

[1] D. Nababan and I. Sadila, "Analisis Personal Financial Literacy dan Financial Behavior Mahasiswa Strata 1 Fakultas Ekonomi Iniversitas Sumatra Utara," *Jurnal Universitas Sumatra Utara*, 2013.

[2] G. Satriaji, Merencanakan Keuangan Pribadi dengan GNUCASH, Tulungagung: Galihsatria Publishing, 2012.

[3] S. Lynch, "Top 50 Personal Finance and Budget Planning Apps," Direct Capital, 13 May 2014. [Online]. Available: http://blog.directcapital.com/business-insights/finance-lending/top-50-personal-finance-and-budget-planning-apps/. [Accessed 11 June 2015].

[4] R. Edmonds, "IDC: Bright future for Windows Phone with 10 percent of the global smartphone market by 2017," windows central, 5 September 2013. [Online]. Available: http://www.windowscentral.com/idc-bright-future-windows-phone-10-per-globally-2017. [Accessed 11 June 2015].

[5] M. F. Rhomadhoni, "Rancang Bangun Class Library dan Integrasi Sistem pada Game Sosial Food Merchant Saga Pada Perangkat Android," *Teknik Informatika FTIF-ITS*, 2014.

[6] Microsoft Corporation, "Class Library," Microsoft Developer Network, [Online]. Available: https://msdn.microsoft.com/en-us/library/d11h6832%28v=vs.71%29.aspx. [Accessed 27 May 2015].

[7] M. E. Fayad, D. C. Schmidt and R. E. Johnson, Building Application Frameworks Object-Oriented Foundations of Framework Design, New York: Wiley Computer Publishing, 1999.

[8] L. J. Gitman, Principle of Finance 11th ed, New Jersey: Prentice Hall, 2004.

[9] A. Bakar, B. Setiawan, G. Syukriawati, L. M. Rahma and W. Wijaya, "Ananlisis Domain pada Website Toko Online," *Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya*, 2012.

[10] A. Hunt, D. Thomas and M. Hargett, Pragmatic Unit Testing In C# With NUnit Second Edition, United States of America: The Pragmatic Programmers, 2007.

[11] Microsoft Corporation, "Integration Testing," Microsoft Developer Network, [Online]. Available: https://msdn.microsoft.com/en-us/library/aa292128(v=vs.71).aspx. [Accessed 28 July 2015].

[12] Wikipedia, "Acceptance testing," Wikipedia, [Online]. Available:

https://en.wikipedia.org/wiki/Acceptance_testing.
[Accessed 29 July 2015].