ANALISIS PERBANDINGAN DAN IMPLEMENTASI METODE SCRUM TERHADAP METODE WATERFALL (STUDI KASUS CV 'X')

Fahmi Firman Anugerah Harsono Putra[1], Dana Sulistyo Kusumo[2], Sri Widowati[3]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

Tingkat efisiensi proses pembangunan perangkat lunak Waterfall untuk proyek kecil-menengah makin memburuk dalam dasawarsa ini. Konsep metodologi pembangunan perangkat lunak baru yaitu Scrum, salah satu keluarga agile dicoba untuk menyelesaikan masalah tersebut. Scrum membutuhkan metode agile lain seperti XP dan RUP sebagai komplemen supaya benar-benar dapat dipakai di lapangan. Scrum adalah istilah yang umum, untuk membedakan dengan Scrum versi lain maka penulis dalam TA ini menggantikannya dengan F-Scrum .
Hasil perbandingan Waterfall dengan Scrum (F-Scrum) dengan parameter ”Relative Effort” dan ”Time to Develop” (COCOMO) adalah bahwa Scrum lebih baik dibandingkan Waterfall untuk kasus tugas akhir ini.

Kata Kunci : Waterfall,Scrum,F-Scrum,COCOMO.

Abstract

Efficiency rate of Waterfall software development process for smallmedium project is getting worse in this decade. The new concept software methodology called Scrum, one of agile family method is tried to solved this problem. Scrum needs complementary other agile method like XP and RUP for really being applied in real world. Scrum is an ubiquitous term, then author of this final assignment refer to F-Scrum to differentiate Srum that is used here toward others.
The result of comparasion between Waterfall and Scrum (call F-Scrum) within “Relative Effort” and “Time to Develop” parameters (COCOMO) has found that Scrum was better than Waterfall for this final assignment scope only.

Keywords : Waterfall,Scrum,F-Scrum,COCOMO.

1 Pendahuluan

1.1 Latar belakang

Tingkat efisiensi pembangunan perangkat lunak mulai menunjukan penurunan yang sangat signifikan. Begitu pula yang dialami CV 'X', sebagai perusahaan pembangun perangkat lunak skala kecil-menengah [8] mengalami peningkatan biaya operasional yang meningkat secara bertahap terlebih untuk perusahaan non berbadan hukum faktor solvabilitas sangat sensitif. Beberapa analis internal perusahaan dan penulis sendiri berasumsi bahwa salah satu faktor pemicu permasalahan ialah metode pembangunan perangkat lunak *Waterfall* yaitu metodologi pembangunan perangkat lunak yang terurut dan sistematik, yang digunakan perusahaan sudah tidak cocok digunakan untuk situasi bisnis sekarang ini. Terlebih terdapat fakta materil yang mendukung asumsi tersebut.

Trend metode *Agile* yaitu metode pembangunan perangkat lunak yang ringan dengan mengedepankan program daripada dokumentasi, yang muncul satu dasawarsa lalu terbukti mampu membantu beberapa perusahaan meningkatkan efisiensinya. Salah satu turunan metode *Agile* yang banyak dipakai ialah *Scrum*. *Scrum* adalah metodologi pembangunan perangkat lunak agile berbasis sinergi antara kerja tim dan kebutuhan bisnis secara *iteratif* dan *incremental*. *Scrum* dalam praktik berperilaku seperti payung yang menaungi metode *agile* lain [1], sehingga beberapa *best practice* yaitu praktik-praktik di lapangan metode *agile* yang diperlukan dapat dipakai dalam lingkup *Scrum*.

Implementasi *Scrum* pada tugas akhir ini penulis mengambil beberapa *best practice Rational Unified Process* (*RUP)* yaitu metodologi *agile* yang menitikberatkan pada arsitektur perangkat lunak dan *Extreme programming* (*XP)* yaitu kumpulan *best practise agile* untuk mengatasi permasalahan dokumentasi desain dan tata cara pemrograman. Untuk membedakan *Scrum* yang terdapat dalam tugas akhir ini dengan *Scrum* versi lainnya maka *Scrum* untuk tugas akhir ini penulis namakan *F-Scrum* yaitu varian *Scrum* yang dilengkapi dengan dokumentasi perangkat lunak dan *best practices* . Untuk melihat tingkat efisiensi *F-Scrum* terhadap *Waterfall* maka digunakan pemodelan *COCOMO* yaitu model pengukuran ongkos pembuatan perangkat lunak yang terdiri dari *Relative Effort* yaitu parameter yang digunakan untuk mengukur ongkos pembuatan prangkat lunak dan *Time to Develop* yaitu parameter untuk mengukur perkiraan waktu yang diperlukan untuk membangun suatu perangkat lunak

1.2 Perumusan masalah

Berdasarkan uraian di atas, maka perumusan masalah yang dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana implementasi tata cara tahapan membangun perangkat lunak berbasis *Scrum*.
2. Bagaimana hasil akhir perbandingan pembuatan perangkat lunak berbasis metoda *Waterfall* dan *F-Scrum* berdasarkan parameter *Relative Effort* dan *Time to Develop* .

Untuk menjaga agar dalam penelitian tetap efektif, permasalahan tidak meluas, dan pembahasan tidak menyimpang dari tujuan semula serta menjadi mudah dipahami sesuai dengan tujuan penelitian yang hendak dilakukan, maka perlu dilakukannya pembatasan masalah sebagai berikut :

1. Spesifikasi kebutuhan perangkat lunak yang baru sama dengan perangkat lunak Aplikasi *Billing* yang telah ada sebelumnya, yang menitikberatkan pada fungsionalitas Registrasi Pelanggan Baru, *Maintenance* Data Pelanggan, Membayar Tagihan dan *Maintenance* Sistem,
2. Perangkat lunak yang dibuat memiliki karakteristik yang sama dengan aplikasi yang telah dibuat yaitu *standalone* yaitu untuk satu pengguna. Hanya konsep pembangunan perangkat lunak yang baru berorientasi objek sedangkan aplikasi terdahulu berbasis struktural,
3. Parameter pembanding utama yang berlaku untuk struktural dan berorientasi objek ialah *Relative Effort* dan *Time to Develop*,
4. Asumsi perbedaan karakteristik pribadi dan kemampuan pembangun aplikasi terdahulu(*Waterfall*) dengan pembangun aplikasi berbasis *Scrum* diabaikan.

Berdasarkan kerangka pemikiran yang telah dikemukakan pada penjelasan diatas maka dapat ditarik hipotesis penelitian :
Efisiensi pengerjaan aplikasi berbasis metode Scrum lebih baik dibandingkan aplikasi berbasis metode Waterfall, ditinjau dari parameter Relative Effort dan Time to Develop untuk perangkat lunak yang sama pada kasus TA ini.

1.3 Tujuan

Adapun tujuan yang hendak dicapai melalui penelitian yang dilakukan ini adalah sebagai berikut :

1. Mengimplementasikan konsep *Scrum* dalam pembangunan perangkat lunak pada praktek lapangan sebenarnya,
2. Membandingkan hasil akhir pembangunan perangkat lunak berbasis konsep *Waterfall* dan *Scrum* berdasarkan parameter *Relative Effort* dan *Time to Develop*,
3. Mengevaluasi hasil perbandingan proses pengerjaan perangkat lunak berbasis *Scrum* terhadap *Waterfall*.
4. Memberikan saran perbaikan atas konsep terapan pembangunan perangkat lunak berbasis *Scrum* khususnya untuk pengembang proyek skala kecil dan menengah.

.

1.4 Metodologi penyelesaian masalah

Sebagai realisasi untuk mencapai tujuan dan memecahkan masalah, dilakukan langkah-langkah sebagai berikut:

1. Studi literatur berhubungan dengan metode *Scrum* dan *Waterfall,*
2. Pengambilan beberapa *best practice RUP* dan *XP* dalam lingkup *Scrum*. Hasil modifikasi *Scrum* ini dinamakan *F-Scrum*,
3. Melakukan studi lapangan untuk membuat perencanaan jadwal pembangunan perangkat lunak,
4. Menyusun *User Stories* berdasarkan masukan dari *stakeholder*,
5. Menyusun *product backlog* yang me*rup*akan hasil translasi dari *User Stories*,

6. Memilih beberapa item *product backlog* untuk dikerjakan dalam waktu satu *Sprint*,
7. Menyusun *Use Case diagram* dan deskripsi *Use Case* terpilih untuk satu *Sprint*,
8. Memulai *Sprint*,
9. Membuat diagram kelas dan diagram *sequence* pada waktu *daily stand up* meeting untuk modul perangkat lunak yang telah selesai,
10. Melakukan *Refactoring* atas desain dan kode pemrograman,
11. Melakukan *Sprint Review* setelah satu *Sprint* selesai,
12. Ulangi *Sprint* sampai semua item product backlog selesai dikerjakan,
13. Membuat diagram komponen setelah perangkat lunak selesai,
14. Melakukan *Sprint Retspective* yang didalamnya berisi evaluasi pengerjaan perangkat lunak dengan *F-Scrum*. Selain itu juga dilakukan analisis perbandingan tingkat efisiensi pembangunan perangkat lunak antara *F-Scrum* dan *Waterfall* dengan melibatkan tim *Scrum* dan tim *Waterfall*.

5 Kesimpulan dan Saran

5.1 Kesimpulan

1. Metode pembangunan perangkat lunak *Scrum* dapat digabungkan dengan beberapa best practice *XP* dan *RUP* menjadi *F-Scrum* yang merupakan hasil implementasi *Scrum* dalam pengerjaan Tugas Akhir ini. .
2. Secara garis besar tata cara pembangunan perangkat lunak dalam *F-Scrum* ialah :
    a. Pembuatan *User Stories*
    b. Pembuatan *Product Backlog*
    c. Pemilihan *Product Backlog*
    d. Pembuatan Use Case Diagram dan deskripsi dari *product backlog* yang dipilih
    e. *Sprint*
        1) *Daily Stand Up meeting*(diagram kelas dan diagram *sequence*)
        2) *Refactoring*
        3) *Sprint* Review
        4) Ulangi *Sprint* sampai semua product backlog selesai diaplikasikan
    f. Pembuatan Diagram Komponen.
    g. *Sprint Retrospective*.
3. Dari hasil pengujian *Waterfall* dengan *Scrum* dalam TA ini dihasilkan informasi sebagai berikut :

Tabel 5-31 : Perbandingan *Waterfall* dan *F-Scrum*

	Waterfall	**F-Scrum**
Relative Effort	49 PM	38 PM
Time tto Develop	7.7 Bulan	6.8 Bulan

4. *Relative Effort Scrum* pada kasus pengerjaan Tugas Akhir ini secara bertahap jauh lebih kecil bila dibandingkan dengan *Waterfall*, secara implisit berarti rata-rata ongkos membangun perangkat lunak lebih murah bila dibandingkan dengan *Waterfall*.
5. *Time to Develop Scrum* pada kasus pengerjaan Tugas Akhir ini secara konstan sedikit lebih cepat bila dibandingkan dengan *Waterfall*, sehingga waktu yang diperlukan untuk membangun perangkat lunak dengan jumlah pembangun yang sama *Scrum* relatif lebih baik bila dibandingkan dengan *Waterfall*.

6. Hipotesis Penelitian pada kasus pengerjaan Tugas Akhir ini bahwa *Scrum* (*F-Scrum*) lebih baik dari *Waterfall* dari segi *Relative Effort* dan *Time to Develop* terbukti

5.2 Saran

1. Untuk lebih membuktikan hipotesis perbandingan *Scrum* dengan *Waterfall* maka harus juga diuji dengan beberapa proyek mulai dari skala kecil sampai besar.
2. Metode *F-Scrum* belumlah sempurna sehingga memerlukan studi lain yang lebih intens sehingga tidak hanya untuk proyek skala kecil saja bisa ditangani, namun juga untuk proyek yang jauh lebih kompleks.
3. Untuk lebih meningkatkan hasil *Relative Effort* dan *Time to Develop* untuk perusahaan disarankan untuk meningkatkan level CMM dan memiliki analisis resiko pembuatan perangkat lunak yang baik.

Daftar Pustaka

[1] Ken Schwaber, Mike Beedle, 2001, "Agile Software Development With *Scrum*", Prentice Hall.

[2] Kent Beck, 2000, "Extreme Programming E*XP*lained Embrace Change", Addison-Wesley.

[3] Martin Fowler, 2002, "Refactoring: Improving the Design of Existing Code", Addison-Wesley.

[4] Mike Cohn, 2004, "*User Stories* Applied: For Agile Software Development", Addison-Wesley.

[5] Per Kroll, Philippe Kruchten, 2003, "Rational Unified Process Made Easy: A Practitioner's Guide to the *RUP*", Addison-Wesley.

[6] Pete Deemer, Gabrielle Benefield, 2006, " *Scrum* Primer", milist yahoo *Scrum*Development, didownload pada Maret 2007.

[7] Roger S. Pressman, 2005, "Software Engineering : A Practitioner's Approach", MC Graw Hill.

[8] SRA, COST II, http://sra.itc.it/people/adolfo/spm07/Software%20Project%20Management, didownload pada 01 September 2007

[9] USC, COCOMO II Reference Manual, ftp://ftp.**usc**.edu/pub/soft_engineering/**COCOMOII,** Didownload pada 01 September 2007.

[10] USC, COCOMO II, materi kuliah COCOMO II oleh DR Ray Madachy via e-mail study group.

[11] Modul Kerja Lab Rekayasa Perangkat Lunak Object Oriented STT Telkom