Personal Extreme Programming – Proses Agile untuk Pengembang Otonom

Yani Dzhurov, Iva Krasteva, dan Sylvia Ilieva

Fakultas Matematika dan Informatika, Universitas Sofia, 5 James Bourchier Blvd, Sofia 1165, Bulgaria
yani.dzhurov@gmail.com, iva.krasteva@rila.bg, sylvia@acad.bg

Abstrak. Meskipun ukuran dan kompleksitas perangkat lunak yang dikembangkan cenderung meningkat, sebagian besar perangkat lunak tersebut masih dikembangkan oleh pengembang otonom. Studi penelitian saat ini mengusulkan modifikasi PSP yang bertujuan untuk meringankan proses pengembangan perangkat lunak dan membuatnya lebih mudah diikuti, sambil tetap mempertahankan prinsip dasar PSP. Metodologi baru tersebut diperluas dengan praktik pengembangan yang terbukti efisien dari Extreme Programming untuk mendukung perencanaan proyek dan pengendalian kualitas produk yang lebih baik. Makalah ini menyajikan hasil adopsi metodologi dan membandingkannya dengan pengembangan ad-hock.

Kata Kunci: optimasi proses perangkat lunak, Proses Perangkat Lunak Pribadi, Pemrograman Ekstrim, pengembangan perangkat lunak tangkas, pengembangan otonom

1 Pendahuluan

Pengembang otonom adalah teknisi perangkat lunak yang mengimplementasikan solusi perangkat lunak tanpa menjadi bagian dari sebuah tim. Teknisi perangkat lunak semacam itu juga disebut pengembang lepas atau kontraktor (perusahaan satu orang). Model pengembang otonom lahir sebagai hasil dari alih daya dan lepas-pakai TI yang dimulai pada awal tahun 90-an abad sebelumnya. Tujuan utama dari alih daya dan lepas-pakai adalah untuk mencapai kualitas produk yang lebih baik dengan biaya tenaga kerja yang lebih rendah. Layanan TI dapat dialihdayakan baik ke perusahaan perangkat lunak maupun ke pengembang otonom independen. Pengembang otonom independen sering kali berada dalam posisi yang lebih baik untuk menawarkan biaya tenaga kerja yang lebih rendah dengan mengurangi biaya operasional perusahaan, tetapi dalam banyak kasus tidak dapat menjamin kualitas produk yang baik dan kompetitif.

Pengembang otonom menghadapi tantangan besar dalam menangani proyek tepat waktu dan dengan kualitas produk yang tinggi serta meminimalkan kegagalan. Prasyarat untuk ini adalah kurangnya perencanaan dan kontrol kualitas. Pengembang otonom memerlukan proses pengembangan yang kaku untuk mengatur aktivitas harian mereka dengan lebih baik dan mengontrol kualitas proyek. Sebagian besar proses pengembangan perangkat lunak saat ini (baik yang tradisional maupun yang tangkas) terutama menargetkan pengembangan tim dan perlu disesuaikan saat diadopsi untuk pengembang otonom. Hanya Proses Perangkat Lunak Pribadi yang secara eksplisit ditentukan untuk digunakan oleh masing-masing insinyur. Namun, penerapan PSP dengan benar memerlukan pengetahuan yang sangat baik tentang spesifikasi proses yang cukup luas. Selain itu, PSP melibatkan banyak upaya untuk menyiapkan dan memelihara sejumlah dokumentasi dan data. Dengan demikian, penerapan PSP dalam pengembangan proyek nyata sangatlah sulit. Pengembang otonom tidak cenderung menginvestasikan sumber daya dan waktu dalam mempelajari dan menerapkan tugas-tugas berat.

proses karena ini akan menunda interval pengiriman yang akan menurunkan keunggulan kompetitif mereka di pasar.

Studi penelitian terkini mengusulkan modifikasi PSP yang bertujuan untuk meringankan proses pengembangan perangkat lunak dan membuatnya lebih mudah diikuti, sekaligus mempertahankan prinsip dasar PSP. Metodologi baru ini diperluas dengan praktik pengembangan yang terbukti efisien dari Extreme Programming untuk mendukung perencanaan proyek dan kontrol kualitas produk yang lebih baik. Tujuan utama dari metodologi yang disarankan, yang diberi nama Personal eXtreme Programming (PXP), adalah untuk meningkatkan kinerja dan kualitas teknisi otonom dengan mengotomatiskan aktivitas pengembang harian dan melakukan retrospeksi secara berkala.

Makalah ini terdiri dari 5 bagian. Bagian selanjutnya berisi ikhtisar tentang prinsip dan praktik PXP. Bagian 3 menyajikan fase-fase yang meliputi proses pengembangan. Bagaimana metodologi baru diimplementasikan oleh pengembang otonom dijelaskan di bagian 4. Bagian ini juga menyajikan analisis komparatif hasil dari penerapan pengembangan ad-hock dan PXP dalam dua fase dalam satu proyek yang sama. Bagian 5 menyimpulkan makalah ini.

2 Prinsip dan Praktik Pemrograman eXtreme Pribadi

Personal Extreme Programming (PXP) adalah proses pengembangan perangkat lunak yang dirancang untuk diterapkan oleh para insinyur perangkat lunak secara individual. PXP bertujuan untuk meringankan PSP dengan mengurangi jumlah skrip yang diikuti dan jumlah data yang harus diisi dalam formulir (untuk referensi lengkap tentang spesifikasi PSP, silakan lihat [1]). PXP mempertahankan prinsip-prinsip dasar PSP tetapi mengurangi jumlah upaya dokumentasi dan pemeliharaan. Selain itu, PXP memperkenalkan subset praktik pengembangan XP yang disesuaikan untuk dilakukan oleh pengembang otonom. Proses pengembangan PXP bersifat iteratif dan menerapkan praktiknya memungkinkan pengembang menjadi lebih fleksibel dan responsif terhadap perubahan. Metodologi yang disarankan sangat bergantung pada otomatisasi bagian utama dari pekerjaan pengembang harian untuk meningkatkan kinerja programmer dan memperpendek interval pengiriman dan waktu yang dihabiskan untuk dukungan sistem perangkat lunak.

Metodologi PXP didasarkan pada prinsip-prinsip berikut:

- PXP membutuhkan pendekatan yang disiplin - pengembang bertanggung jawab untuk mengikuti proses dan penerapan praktik PXP
- Pengembang harus mengukur, melacak, dan menganalisis pekerjaan harian mereka
- Pengembang harus belajar dari variasi kinerja mereka dan bertujuan untuk meningkatkan proses berdasarkan data proyek yang dikumpulkan
- PXP melibatkan pengujian berkelanjutan
- Perbaikan cacat harus dilakukan pada tahap pengembangan awal, ketika biaya itu lebih rendah
- Pengembang harus mencoba mengotomatiskan sebanyak mungkin aktivitas harian mereka bekerja

PXP disertai dengan 14 praktik pengembangan dan pengorganisasian. Metodologi ini mengandalkan praktik-praktiknya untuk memastikan kualitas produk yang baik dan perencanaan proyek yang akurat. Beberapa praktik PSP dipertahankan dan beberapa praktik lain dari XP diperkenalkan untuk menggantikan metode formal dan kompleks untuk perencanaan, desain sistem, dan verifikasinya.

Enam praktik PXP dilestarikan dari PSP seperti yang dijelaskan dalam [2]:

- Perekaman Waktu
- Jenis Cacat Standar
- Rekaman Cacat
- Pengukuran Ukuran
- Proposal Peningkatan Proses
- Ulasan Kode

Enam praktik PXP terbukti merupakan praktik pembangunan yang efektif dari Pemrograman Ekstrim ([3] dan [4]):

- Integrasi Berkelanjutan – PXP mencakup praktik pembuatan versi kontrol sumber, pembuatan otomatis, pelaksanaan pengujian otomatis, dan pengiriman cacat otomatis.
- Desain Sederhana •
  Rilisan kecil
- Refactoring •
  Pengembangan Berbasis Pengujian
- Solusi Lonjakan

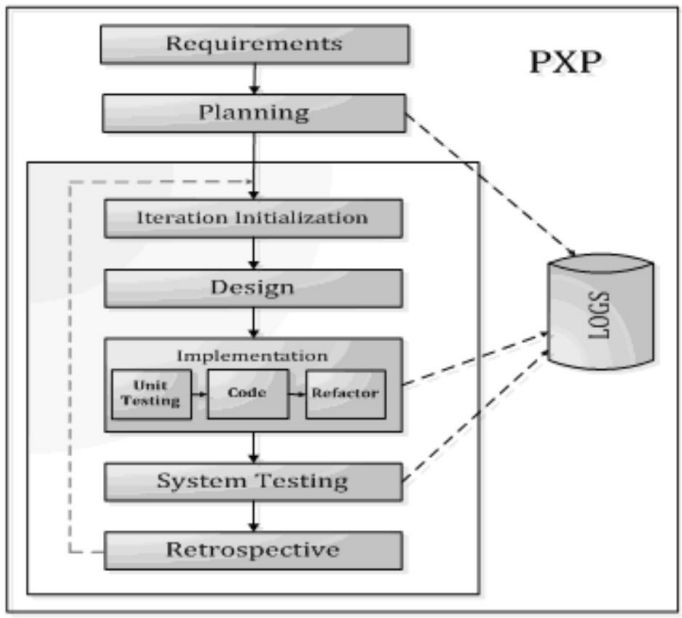
Salah satu praktik PXP, yaitu Coding Standard, hadir di PSP dan XP. PXP menyarankan cara baru untuk melakukan aktivitas perencanaan. Dalam PXP, perencanaan tugas terutama didasarkan pada laporan dari proyek sebelumnya. Untuk setiap persyaratan fungsional, serangkaian tugas teknis ditetapkan. Setiap tugas teknis diberi kategori – misalnya pembuatan pengujian unit, implementasi kelas bisnis, pembuatan SP, desain formulir UI, dll. Estimasi tugas didasarkan pada estimasi tugas dari kategori yang sama di proyek sebelumnya. Pengembang harus menggunakan data yang dikumpulkan sebelumnya dari proyek lain untuk memprediksi kinerja mereka – mereka tidak boleh mendapatkan waktu rata-rata untuk tugas dari kategori tertentu, tetapi mempertimbangkan tren nilai dalam laporan. Asumsinya adalah bahwa ketika pengembang melakukan tugas secara teratur di beberapa titik, itu menjadi rutinitas yang membutuhkan waktu lebih sedikit atau sama untuk dilakukan seperti terakhir kali tugas tersebut diimplementasikan.

3 Tahapan Pemrograman eXtreme Pribadi

Proses PXP bersifat iteratif dan terdiri dari beberapa iterasi dan siklus sebaris. Perencanaan persyaratan dan tugas biasanya dilakukan untuk keseluruhan proyek karena biasanya persyaratan ditentukan sebelumnya dan tetap stabil selama implementasi.

Jika terjadi perubahan persyaratan, perencanaan tugas dapat direvisi. Iterasi pengembangan dimulai dengan inisialisasi iterasi dan diakhiri dengan retrospeksi iterasi. Selama keseluruhan proses, pengembang menyimpan berkas log berisi informasi mengenai perencanaan tugas dan durasi aktual, saran perbaikan, jumlah cacat, dan detailnya. Gambar 1 memberikan gambaran umum proses PXP.

Selama fase persyaratan, dokumen berisi persyaratan fungsional dan non-fungsional untuk sistem dibuat. Fase ini bersifat opsional – dokumen persyaratan dapat dibuat dalam rapat antara klien dan karyawan lain dari organisasi tempat pengembang bekerja, atau dapat diperoleh dari insinyur perangkat lunak lain, jika ada perluasan produk yang sudah ada.



Gbr. 1. Tahapan proses PXP.

Pada tahap perencanaan, pengembang menyusun serangkaian tugas berdasarkan dokumen daftar persyaratan. Setiap tugas dapat terdiri dari tugas-tugas kecil yang dikategorikan. Setiap tugas kecil diestimasikan berdasarkan data perencanaan untuk tugas-tugas dengan tipe yang sama dari proyek-proyek sebelumnya (jika data yang dikumpulkan sebelumnya tidak ada, maka pengembang harus mencoba membuat asumsi terbaik tentang perkiraan waktu yang dibutuhkan untuk menyelesaikan tugas tersebut). Jumlah estimasi tugas anak adalah estimasi untuk tugas induk. Selama tahap ini, sebelum perencanaan tugas, keputusan desain utama dibuat – bahasa pemrograman apa yang akan digunakan, kerangka kerja pengembangan, model aplikasi, dll.

Inisialisasi Iterasi menunjukkan awal setiap iterasi. Iterasi dimulai dengan pemilihan tugas, yang akan menjadi fokus iterasi. Durasi iterasi dapat bervariasi dari 1 hingga 3 minggu, tergantung pada cakupan proyek. Setiap iterasi dapat menghasilkan kandidat rilis atau versi produk yang dirilis.

Selama fase desain, pengembang otonom memodelkan modul dan kelas sistem yang akan diimplementasikan dalam iterasi yang sedang berlangsung. Pengembang harus bertujuan untuk mendesain sistem agar hanya memenuhi persyaratan klien saat ini tanpa mencoba menebak apa yang akan dibutuhkan di masa mendatang. Metode desain dipilih oleh pengembang, tetapi disarankan untuk menggunakan alat sesederhana mungkin.

Tahap implementasi adalah tahap di mana pembuatan kode sebenarnya berlangsung. Pengembang mengimplementasikan semua objek yang ditetapkan dalam tahap desain sebelumnya dan melakukan pengujian untuk objek tersebut. Tahap ini terdiri dari tiga sub-tahap: pengujian unit, pembuatan kode, dan pemfaktoran ulang kode, yang dijalankan dalam urutan berikut:

Untuk keluar dari fase implementasi, kode harus dikompilasi tanpa kesalahan dan semua pengujian unit harus berhasil.

Semua fitur yang dikembangkan sejauh ini diuji selama fase pengujian sistem.

Pengembang harus memverifikasi apakah solusi yang diterapkan memenuhi persyaratan awal proyek. Semua cacat yang ditemukan dicatat dan diperbaiki.

Retrospektif menandai akhir dari iterasi proses. Analisis data yang dikumpulkan selama fase lainnya sedang dilakukan. Pengembang harus memverifikasi apakah perkiraan waktu tugas sama dengan waktu sebenarnya, menemukan alasan potensi penundaan untuk mencegah perkiraan yang terlalu rendah atau terlalu tinggi dalam proyek mendatang.
Pengembang menganalisis semua proposal perbaikan proses dan, jika diperlukan, ia dapat mengadaptasi praktik proses yang menyertainya. Perubahan yang mengatasi masalah dalam proses dan praktik harus dilakukan sedini mungkin untuk mencegah kegagalan proyek. Fase ini dapat diakhiri dengan rilis kandidat atau rilis produk. Fase 'Retrospektif' dapat memulai iterasi baru dengan beralih ke fase 'Inisialisasi Iterasi' atau untuk menandai akhir pengembangan proyek ketika semua persyaratan klien untuk sistem perangkat lunak terpenuhi, dan tidak ada lagi cacat yang belum terselesaikan.

4 Menerapkan Pemrograman eXtreme Pribadi

Dalam penelitian saat ini Microsoft Visual Studio System (VSTS) digunakan untuk implementasi Personal Extreme Programming. VSTS adalah platform alat yang produktif dan terintegrasi selama seluruh siklus hidup pengembangan perangkat lunak. Mereka meningkatkan komunikasi dan kolaborasi dalam proses pengembangan perangkat lunak, dan memungkinkan perluasan oleh tim yang menggunakannya. Meskipun VSTS terutama menargetkan tim pengembangan, pengembang otonom juga dapat memanfaatkan alat-alatnya yang canggih. VSTS dirancang untuk memenuhi kebutuhan insinyur perangkat lunak yang menggunakannya dengan templat proses pengembangan perangkat lunak. Templat proses terdiri dari serangkaian dokumen xml terstruktur yang menjelaskan berbagai aspek proses (spesifikasi terperinci dari templat proses tersedia di [5]). Untuk PXP, templat proses baru telah dibuat.

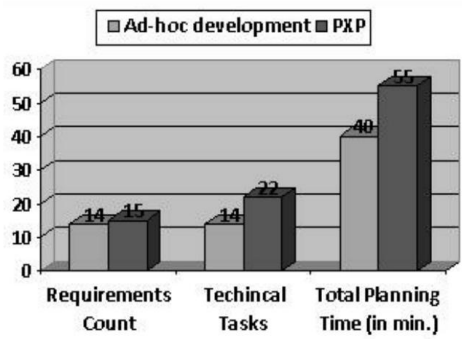
Templat proses berisi informasi tentang fase-fase proses PXP, item pekerjaan khusus untuk metodologi pengembangan ini (cacat, tugas, skenario, persyaratan kualitas layanan, proposal peningkatan proses), definisi laporan yang digunakan selama fase perencanaan, pengaturan dan izin sistem kontrol versi, dan terkait dengan portal web proyek untuk penyimpanan dokumen. [6] memberikan gambaran umum tentang alat bawaan VSTS yang mendukung semua praktik PXP yang menyertainya – alat untuk melakukan refaktor, peninjauan kode otomatis, pelacakan item pekerjaan, tampilan laporan, pembuatan jarak jauh otomatis dan integrasi berkelanjutan, implementasi dan pelaksanaan pengujian, pengukuran ukuran dan kualitas perangkat lunak, dan alat untuk desain sistem. Deskripsi terperinci tentang implementasi PXP dengan VSTS lihat [7].

Untuk mempelajari sejauh mana PXP membantu mengoptimalkan proses pengembangan, metodologi ini diadopsi oleh pengembang otonom. Sebuah proyek perangkat lunak dikembangkan dalam dua fase, yang masing-masing direncanakan berlangsung selama 2 minggu.
Tahap pertama dikembangkan dengan pengembangan ad-hoc, dan tahap kedua dengan PXP.

Selama kedua fase, proses dan metrik proyek berikut dilacak dan dianalisis:

- Upaya perencanaan – waktu yang dihabiskan pada fase perencanaan proses
- Efektivitas perencanaan – analisis rasio antara waktu yang direncanakan dan waktu aktual yang dihabiskan untuk tugas-tugas proyek.

- Rasio antara cacat yang ditemukan dengan jumlah cacat fungsional dan non-fungsional. persyaratan fungsional
- Cakupan Kode Pengujian Unit – persentase kode yang dicakup (dieksekusi) saat pengujian unit dijalankan. Metrik ini memungkinkan pelacakan bagian aplikasi mana yang dapat diuji secara otomatis.



Gbr. 2. Hasil dari fase perencanaan dalam pengembangan ad-hoc dan PXP.

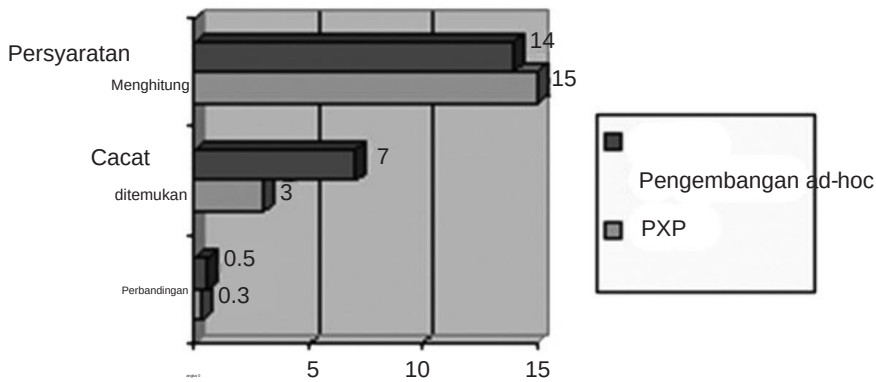
Gambar 2 memberikan informasi tentang upaya perencanaan di kedua fase. Fase perencanaan mencakup jumlah persyaratan yang hampir sama untuk setiap metode pengembangan (pengembangan ad-hoc - 14 persyaratan, dan PXP – 15).
Karena kurangnya data yang dikumpulkan sebelumnya untuk proyek PXP, perencanaan di kedua fase berlangsung dengan cara yang relatif sama. Perbedaan utamanya terletak pada jumlah tugas teknis yang ditetapkan – PXP menetapkan tugas yang lebih rinci dibandingkan dengan pengembangan ad-hoc. Fase I menetapkan satu tugas untuk setiap persyaratan - 14, sedangkan untuk fase II ditetapkan 22 tugas. Dalam PXP, waktu yang dihabiskan untuk perencanaan 37,5% lebih banyak daripada pengembangan ad-hoc, karena jumlah tugas yang akan diperkirakan lebih besar.

Tabel 1 memberikan gambaran umum efektivitas perencanaan di kedua fase proyek. Pengembangan ad-hoc tidak mencakup pelacakan waktu tugas yang direncanakan, jadi tidak mungkin untuk memberikan informasi terperinci tentangnya, tetapi total waktu tambahan yang dihabiskan untuk pengembangan – 16 jam yang merupakan 20% dari total waktu yang direncanakan. Dalam PXP pelacakan waktu adalah salah satu praktik utama yang memfasilitasi pengumpulan data proyek sehingga informasi yang lebih terperinci tersedia – 5 tugas selesai sebelum waktu yang direncanakan, 3 tugas selesai tepat waktu, dan 14 selesai setelah waktu yang direncanakan. Waktu pengembangan aktual yang dihabiskan di PXP adalah 67,5 jam ketika yang direncanakan adalah 58 jam. Hasilnya menunjukkan bahwa efektivitas perencanaan di kedua fase proyek kira-kira sama – alasannya adalah bahwa perencanaan itu sendiri dilakukan dengan cara yang sangat mirip. PXP memiliki efektivitas perencanaan yang sedikit lebih baik karena jumlah tugas yang ditetapkan lebih banyak – lebih mudah untuk memperkirakan serangkaian tugas yang lebih kecil dengan lebih baik dibandingkan dengan satu tugas besar (seperti halnya dalam pengembangan ad-hoc).

Tabel 1. Gambaran umum efektivitas perencanaan dalam pengembangan ad-hoc dan PXP.

	Pengembangan Ad-hoc PXP	
Jumlah tugas yang dinilai terlalu tinggi	-	5
Jumlah tugas yang diperkirakan dengan benar	-	3
Jumlah tugas yang diremehkan	-	14
Total Waktu Pengembangan yang Direncanakan (dalam jam)	80	58
Selisih waktu aktual dan rencana 16 jam (20%)		9,5 jam (16,3%)

Jumlah cacat yang ditemukan pada fase proyek kedua jauh lebih rendah dibandingkan dengan fase pertama (Gambar 3). PXP terutama berfokus pada pengembangan berbasis uji yang memungkinkan cacat dicegah selama fase implementasi proses. Pada fase I pengembangan proyek, hanya pengujian manual yang dilakukan dan sebagian cacat muncul selama pengembangan fitur baru.



Gbr. 3. Rasio antara cacat yang ditemukan dan persyaratan pada kedua fase proyek.

Metrik proyek terakhir yang dilacak adalah persentase kode yang dicakup oleh pengujian unit. Dalam pengembangan ad-hoc, pengujian unit tidak diimplementasikan sehingga kode dapat dianggap 0%. Dalam pengembangan berbasis pengujian PXP merupakan salah satu praktik utama yang dilakukan yang menyebabkan pengujian unit diimplementasikan untuk semua kelas yang termasuk dalam proyek. Dalam proyek kedua yang dikembangkan secara bertahap dengan PXP, cakupan kode pengujian unit mencapai 85,29%. Tingkat cakupan kode yang tinggi memungkinkan perluasan dan pemeliharaan sistem yang lebih mudah dengan lebih sedikit pengujian manual yang dilakukan.

5. Kesimpulan

Makalah ini menyajikan proses perangkat lunak baru yang ditujukan bagi pengembang otonom yang menjawab kebutuhan terkini dan masalah pekerjaan sehari-hari, meningkatkan kualitas sistem perangkat lunak yang dikembangkan, dan mempersingkat waktu implementasi. PXP mengecualikan skrip formal, formulir, dan metode verifikasi desain dari PSP, serta memperkenalkan beberapa praktik pengembangan XP. Implementasi proses baru ini didukung oleh platform Microsoft Visual Studio Team System yang memberikan peluang untuk mengotomatiskan sebagian besar aktivitasnya. Lebih jauh, analisis komparatif pengembangan yang mengikuti proses ad-hock dan PXP dilakukan untuk memvalidasi bagaimana PXP memengaruhi pekerjaan pengembang otonom. Perbandingan kumpulan artefak yang dihasilkan dari kedua fase tersebut dilakukan.

Metrik lain yang dipantau dari proses mengenai perencanaan proyek dan kualitas produk menyatakan bahwa perencanaan saat PXP diterapkan untuk pertama kalinya sama efektifnya dengan perencanaan dalam pengembangan ad-hoc. Alasannya adalah karena tidak ada data yang dikumpulkan sebelumnya mengenai fase perencanaan proyek lain yang dapat digunakan sebagai dasar untuk proyek saat ini. Dalam proyek pertama yang dikembangkan dengan PXP, waktu untuk perencanaan jauh lebih lama dibandingkan dengan pengembangan ad-hoc (sekitar 40% lebih lama), tetapi ini merupakan investasi awal yang akan segera membuahkan hasil.

dalam pengembangan proyek berikutnya. Tidak seperti perencanaan, dengan mempertimbangkan pelacakan cacat PXP akan memberikan hasil positif bahkan dalam pengembangan proyek pertama. Selama pengembangan modul dengan cakupan yang hampir sama, terjadi penurunan jumlah cacat yang ditemukan dalam fase PXP sekitar 50% dibandingkan dengan pengembangan ad-hoc. Fokus PXP pada pengembangan berbasis pengujian menghasilkan sekitar 85% cakupan kode logika bisnis melalui pengujian unit. Selain itu, UI web dan pengujian beban kinerja diimplementasikan dan dijalankan. Semua pengujian dapat dijalankan secara otomatis, yang bertujuan untuk memfasilitasi perluasan dan pemeliharaan sistem.

Sebagai pekerjaan di masa mendatang, validasi pendekatan harus dilakukan dengan mengimplementasikannya oleh pengembang otonom yang berpengalaman dalam PXP dan memiliki data riwayat dari implementasi proses sebelumnya. Dengan cara ini, pengembang dapat memperoleh seluruh potensi praktik perencanaan PXP. Lebih jauh, pengembang dapat mengadaptasi praktiknya dengan kebutuhannya berdasarkan tinjauan retrospektif dan saran untuk perbaikan yang dibuat dalam iterasi dan proyek sebelumnya.

Ucapan terima kasih. Karya ini sebagian didanai oleh Kementerian Pendidikan dan Sains Bulgaria, NSF.

Referensi

1. Software Engineering Institute, Carnegie Mellon, (Agustus 2005) “Proses Perangkat Lunak Pribadi – Tubuh Pengetahuan, v.1.0”, http://www.sei.cmu.edu/tsp/tools/bok.html
2. Software Engineering Institute, Carnegie Mellon, (2006) “Belajar Mandiri PSP” Bahan”, http://www.sei.cmu.edu/tsp/tools/student.html
3. Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, Juhani Warsta, (2002) “Metode Pengembangan Perangkat Lunak Agile – Review dan Analisis”
4. Kent Beck, (1999) “Penjelasan Pemrograman Ekstrim: Merangkul Perubahan”, Addison-Wesley Professional
5. Jean-Luc David, Mickey Gousset dan Erik Gunvaldson, (2007) “Professional Team Foundation Server”, Wrox Press
6. Richard Hundhausen, (2006) „Bekerja dengan Tim Microsoft Visual Studio 2005 Sistem”, Microsoft Press
7. Dzhurov Yani, (2008) „Menerapkan proses pengembangan perangkat lunak untuk pengembang perangkat lunak otonom dengan Sistem Tim Microsoft Visual Studio”, Tesis Magister, Universitas Sofia