GSLetterNeo Vol.78
2015年1月

創造的な仕事を取り戻すために（３）

土屋　正人
Masato Tsuchiya

前回までに、**ソフトウエア開発の「苦」は創造的な仕事に使える時間の多くが奪われること**で、これを脱するための施策を、Strategy（攻め）、Safety（守り）、Support（支え）、Stream（流れ）の四つのSの視点から捉えることを示し、「攻め」と「守り」を考察しました。今回は「支え」と「流れ」を考えてみます。

◆「支え（Support）」の視点

プロジェクトのタスクの状態と状況は毎日変化します。プロジェクト管理者、リーダ、メンバは役割毎に、それぞれ関心事が異なります。そのため、関心事に対応した進捗管理を行い、担当タスクの状態・状況を把握して、それらの課題やリスクを関係者で共有する必要があります（**タスクの状態・状況の管理**）。

リスクや課題の状況は毎日変化します。発生するQAや不具合は毎日記録・更新し、トラブルが想定される事案とその確度を予測して対策を準備する必要があります（**リスク・課題・QA・不具合の管理**）。

また、並行開発では、特定のリリースの構成の再現が求められます。構成要素のリビジョンがひとつでも違うと再現できず、多大な損失を招きます。そのため、リリースのリビジョンや変更、マージの管理を行う必要があります（**リビジョンの管理**）。

このような開発環境の構築には専門知識が必要で、コストがかかります。複数のツールを導入・連携させる場合、個々のツールのバージョンによっては不具合が出るリスクがあります。組織やプロジェクトに合わせて、無理のない導入計画（最小のコストで最大の効果）を立てる必要があります（**環境構築コストの削減**）。

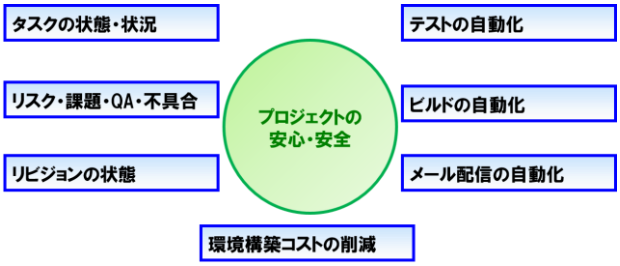

図１　「支え」の７つの施策

プロジェクトメンバ全員に確実に伝えたい情報がある場合、即効性がありコストがかからないのはメーリングリストです。成果物の登録など、特定のイベントが起きたときに関係者にメールが自動配信されるようにすることで、確実に事実を伝達することができます。メールを証跡として利用する場合や条件検索を行うためには、メーリングリストのアーカイブを作成する必要があります（**メール配信の自動化**）。

ビルドを毎日自動で実行することで、不具合を早期に検出できます。ビルド手順をスクリプトにすることで、ノウハウを共有することができますし、実行順序や使用するオプションなどのケアレスミスがなくなります。これらにより品質と効率が向上し、コストが削減できます。そのためには、ビルドの構成や時間などの設定が容易にできることが求められます（**ビルドの自動化**）。

テストを毎日自動で実行することで、不具合を早期に検出できます。特に再帰テストを自動化することで、デグレードを早期に検出できます。これは、夜間に実行することでリソースを効率的に使えます。これらにより品質と効率が向上し、コストが削減できます。そのためにはテストの構成や時間などの設定が容易にできることが求められます（**テストの自動化**）。

これらの施策が連携できると、開発者が安心してソースコードをチェックインできる環境が整います。

◆「流れ（Stream）」の視点

ここに挙げる施策は、リーンソフトウエア開発で示されたソフトウエア開発における 7 つのムダに対応しています。[1]ムダを認識し、どれだけの時間が割かれているかを把握することが、価値の最大化につながります。

仕掛中の作業は、完成するまで価値を生み出しません。価値のある成果物になるまでに、手戻りが発生するリスクがあります。手戻りにはムダなコストがかかります。仕掛中の作業がどれくらいあるか、何が進行の妨げになっているのかを把握し、手戻りリスクの対策を立ててコントロールする必要があります（**未完成の作業のムダ**）。

誰からも、どこからも参照されない成果物は、価値を生みません。開発プロセスで「必要な成果物」と定義されていても、すべてのプロジェクトで価値を生むとは限りません。価値を生まない成果物には、ムダなコストがかかります。参照されない成果物がないかを把握し、保守対象から外すなど、ムダなコストが発生するリスクを抑える必要があります（**余分なプロセスのムダ**）。

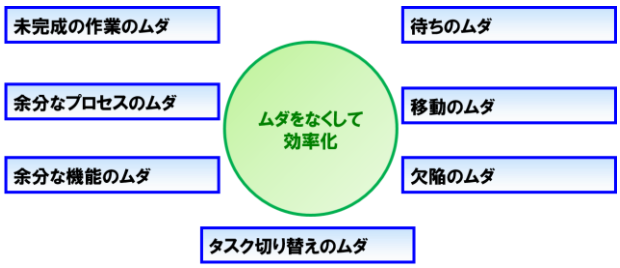

図 2　「流れ」の 7 つの施策

要件として定義した機能は、本当に使われるかどうかわかりません。使われない機能は価値を生まず、ムダなコストがかかります。顧客と要件を把握し、短いサイクルで優先順位を見直して、余分なものを作るリスクを抑える必要があります（**余分な機能のムダ**）。

メンバが同時に複数のチームに入っていると、作業の中断が多くなり、タスク切り替えも増えます。中断した作業から復帰するまでには時間がかかります。復帰するまでの時間は価値を生み出さず、ムダなコストがかかります。メンバのアサイン状況を把握し、ひとつのプロジェクトに集中できるように配員を調整して、タスク切り替えコストを抑える必要があります（**タスク切り替えのムダ**）。

欠陥コストは、除去するまでの時間に比例します。致命的な欠陥が短時間発見されないよりも、小さな欠陥が長期間発見されない方が、コストがかかります。テストと統合が頻繁に行える環境を用意して欠陥を早期に発見し、その状態を監視して、影響を最小限に抑えるよう、コントロールする必要があります（**欠陥のムダ**）。

ドキュメントに本当に必要な情報が書かれていないと、読む人は書いた人に何度も聞きにいくことになります。ドキュメントを書いた人と読む人双方が時間を取られます。引き継ぐための時間にはムダが入り込みやすく、コストがかかります。ノウハウや Tips などをプロジェクトで共有し、コミュニケーションを増やして、引継ぎコストを抑える必要があります（**移動（引継ぎ）のムダ**）。

仕様やアーキテクチャ、インタフェイスが決まらないと、これらを受けて作業する人に待ち時間が生じます。待ち時間は価値を生み出さず、ムダなコストがかかります。ボトルネックとそれによって生じるリスクを関係者で共有し、決断に必要な情報を特定して、リスク軽減策を立てる必要があります（**待ちのムダ**）。

◆ProjDepot という「器（うつわ）」

これまで挙げてきた施策を支援するツールとして、SRA の開発現場から生まれた ProjDepot があります。ProjDepot については、以下の URL をご参照ください。

http://www.sra.co.jp/projdepot/index.html。

1　メアリ＆トム・ポッペンディーク著「リーンソフトウエア開発」日経 BP 社

GSLetterNeo Vol.78
2015 年 1 月 20 日発行
発行者●株式会社 SRA 先端技術研究所
編集者●土屋正人

バックナンバを公開しています●http://www.sra.co.jp/gsletter
ご感想・お問い合わせはこちらへお願いします●gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋２－３２－８