ACM queue(queue.acm.org) 記事

思考フレームワークを「アクション可能なカーネル (actionable kernel)」として提示する。

bY Iver Jacobson, Pan-Wei NG, Paul E. Mcmahon, Ian Spence, and Svante Lidman

ソフトウェアエンジニアリングのエッセンス: SEMAT カーネル

The Essence of Software Engineering: The Semat Kernel

ソフトウェア開発に携わる人なら誰でも、それが複雑でリスクの高い仕事であること、そして関わる人々がよりよいソフトウェアにつながる新しいアイディアをいつでも探していることを知っている。幸運なことに、ソフトウェアエンジニアリングはまだその若年代にあり、毎年毎年ベストプラクティスの中にイノベーションと改善を見てとれる成長過程の専門分野である。それは例えば、リーンとアジャイルの考え方（lean and agile thinking）がソフトウェア開発チームにもたらした改善と恩恵を見てみれば分かる。ソフトウェア開発チームが成功するためには、動くソフトウェアシステムをすばやく提供すること、ステークホルダーを満足させることや、リスクに対処すること、さらには仕事の仕方（way of working）を改善することのバランスをうまくとらなくてはならない。

そのためには、現在の仕事の仕方と、採用しようとする新しいアイディアのギャップを橋渡しする効果的な思考フレームワーク（thinking framework）が必要である。この記事は、そのような思考フレームワークを「アクション可能なカーネル（actionable kernel）」の形で提示することで、様々なリスクのバランスをとり、仕事の仕方を改善しようとするチームを支援する。

このカーネルの構築、すなわち「ソフトウェアエンジニアリングのエッセンス（the essence of software engineering）」は、「ソフトウェアエンジニアリングの方法論と理論（Software Engineering Methods and Theory; SEMAT）」の行動宣言（call for action）（図1）に触発されたものであり、それに対する直接の回答でもある。そして、ソフトウェアエンジニアリングの再定義に向けた小さな一歩でもある。

SEMATは2009年9月に、人々のソフトウェア開発手法への関わり方を抜本的に変更する時期に来ていると感じた3人、Ivar Jacobson、Bertrand Meyer、Richard Soleyによって創設された [3,4,8] 。彼らは行動宣言を書き、いくつかの致命的な問題を特定し、なぜ行動が必要かを説明し、そして何が必要なのかを示唆した。その行動宣言を以下に示す。

現在のソフトウェアエンジニアリングのいくつかの分野では、未成熟なプラクティス(immature practices)に苦しめられている。具体的には、以下の問題を含む。

* 言葉の流行が、エンジニアリングの一分野というよりファッション業界のようである。
* 堅固で広く受け入れられるような理論的基礎を欠いている。
* 非常に多くの方法論(methods)とその派生であふれ、それらの違いはほとんど理解されず作為的に強調されている。
* 信頼できる実験的評価(experimental evaluation)と妥当性確認(validation)を欠いている。
* 産業界の実践(industry practice)と学界の研究(academic research)の乖離。

このように行動宣言は、ソフトウェア産業が流行とファッションに陥りやすいという前提を置いており、ある人々には「新しいアイディアへの抵抗」と映ったようである。しかし、それは真実からかけ離れている。この記事および間もなく刊行される書籍（The Essence of Software Engineering — Applying the SEMAT Kernel）[6]で見てとれるように、SEMATの支持者は新しいアイディアに敏感である。

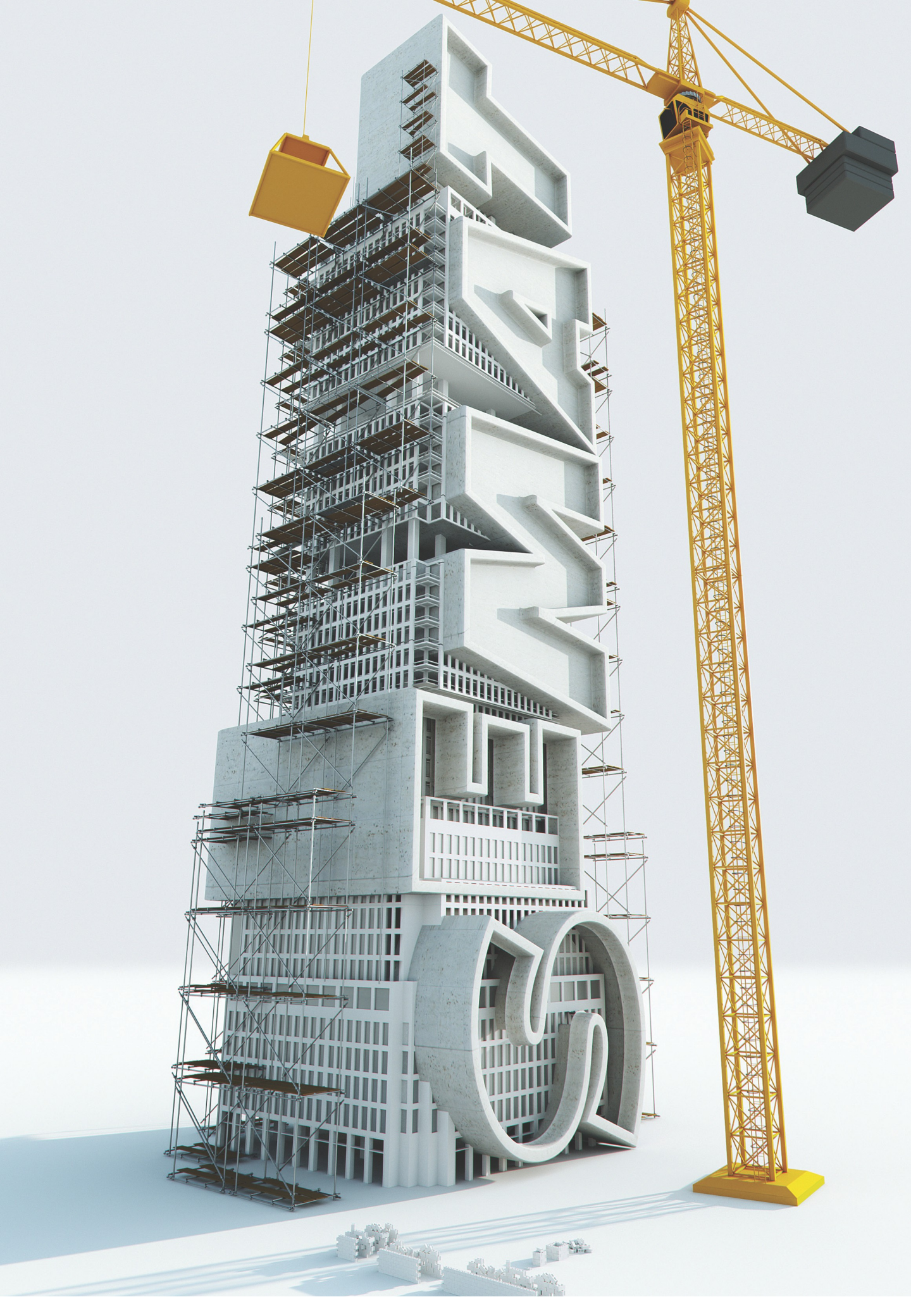


図1. 我々が仕事で扱うこと

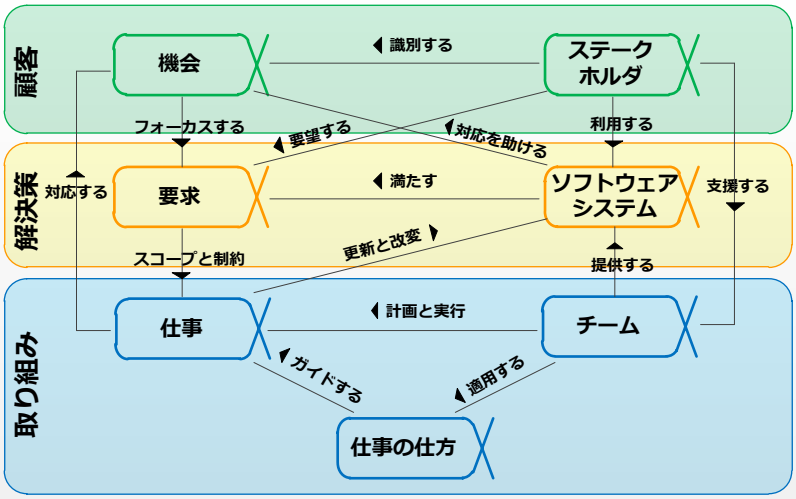
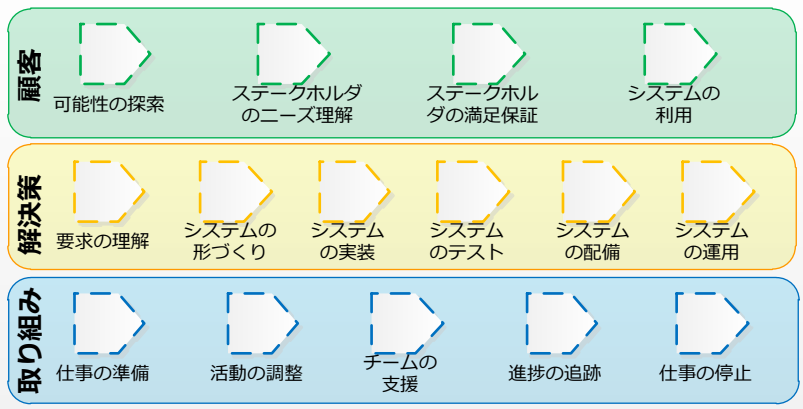


図2. 我々が行うこと



SEMATの支持者が反対しているのは「新しいアイディア」ではない。ただソリューションが流行っているから、あるいは、単なる政治的な理由や同僚からの圧力によって、不適切なソリューションを採用する人々の非リーン（non-lean）、非アジャイル（non-agile）な行動である。

SEMATは、堅固な理論および検証された原則とベストプラクティスに基づいて、ソフトウェアエンジニアリングを再建するプロセスを支援する。
SEMATは以下の特徴を備えている。
* 広く合意された要素からなり、特定用途に拡張可能なカーネルを含む。
* 技術の問題と人の問題の両方を扱う。
* 産業界、学界、研究者、そして、ユーザによって支えられる。
* 要求と技術の変化に応じて追随できるような拡張性を備える。

SEMATの行動宣言は支持を広く集め、署名者や支持者が増え続けている（http://www.semat.org）。2010年2月には、SEMATの創設者は行動宣言をビジョン声明（vision statement）として発展させた [5]。

このビジョンに沿って、SEMATは2つのゴールに焦点を合わせた： 広く合意された要素からなるカーネルを見つけること、そして、堅固な理論的基礎を定義することである。
これらの2つの作業は、かなりの範囲で独立している。カーネルやその要素を見つけることは、多数の既存手法の知識を備えた経験のあるソフトウェアエンジニアによる実践的な活動となる。対して理論的基礎の定義はアカデミックな研究活動であり、成功といえる結果に到達するには何年もかかるだろう。

共通基盤の「力」

SEMATの第一歩は、ソフトウェア開発における共通基盤(common ground)を捉えることであった。この共通基盤は、すべてのソフトウェア開発活動において普遍的な必須要素のカーネル(核; kernel)、および、手法やプラクティスを記述するためのシンプルな言語として示される。
カーネルは、SEMAT のOMG(Object Management Group)に対する提案の中で初めて公開された。図1と図2に示すように、カーネルはソフトウェアシステムを開発するときに「我々が常に仕事で扱うこと)things we always work with)」と「我々が常に行うこと(things we always do)」に関する厳選された少数の概念からなる。
「我々が常に持つべきスキル(skills we always need to have)」を定義する活動も進行中である。しかし、これはカーネルの将来のバージョンまで待たなければならない。

カーネルは単なる概念モデルではなく、以下を提供する。
・チームにおいて、開発の進捗や取り組みの健全性(health of endeavors)を推論し判断するための思考フレームワーク
* ソフトウェア開発の手法とプラクティスについて議論、改善、比較および共有するための共通基盤
* チームが、別々に定義され由来の異なるプラクティスを集めてチーム独自の仕事の仕方を組み立てて、継続的に改善するためのフレームワーク
* 開発されたソフトウェアや、用いられた開発方法論の品質を評価するためのプラクティス非依存な測定方法を定義する基礎(foundation)
* 最も重要なこととしてチームがどこにいて、次に何をすべきで、どこを改善すべきかを理解することの支援方法

ビッグ・アイディア

カーネルを、ソフトウェアエンジニアリングの単なる概念モデル以上とするものは何か？何が新しいのか？それは、基礎を成す原則(図3)であり、原則が次の3つの特徴をカーネルにもたらしている:
アクション可能なこと(actionable)、拡張可能なこと(extensible)、および、実践的なこと(practical)である。

カーネルはアクション可能である。

カーネルの優れた特徴として、「我々が常に仕事で扱うこと」を管理しているという点がある。これらは「ワークプロダクト(work product)」(例えば文書)としてではなく、「アルファ(alpha; 主要素)」という名前で呼ばれている。アルファとはソフトウェアエンジニアリングの取り組みにあたり本質的な要素であり、ソフトウェアエンジニアリングの進展や健全性の評価に関わっている。図1に示すように SEMATでは7つのアルファを識別している: 機会(opportunity)、ステークホルダ(stakeholders)、要求(requirements)、ソフトウェアシステム(software system)、仕事(work)、仕事の仕方(way of work)、チーム(team)。

アルファはシンプルに進展や健全性を表す状態集合として特徴付けられる。例えばソフトウェアシステムは選択されたアーキテクチャにおける次の状態間を遷移する: 論証完了(demonstrable)、使用可能(usable)、準備完了(ready)、運用(operational)、退役(retired)。各状態には当該状態に至る際に満足すべき基準を指定したチェックリストがある。それらの状態によってカーネルはアクション可能なものとなり、ソフトウェア開発チームの振る舞いをガイドできるようになる。

カーネルは、ソフトウェア開発を線形のプロセスではなく協調する要素群のネットワークとして表す。この要素群をバランスをとって維持することで、チームが効果的かつ効率的に開発を進めて、無駄を省き、すばらしいソフトウェアを開発することができる。カーネルのアルファは、実際に適用するプラクティスや従う指針からは独立しており、ソフトウェア開発の取り組みを駆動し進めるフレームワーク全体を与える。

カーネルに加えられるプラクティスが増えてくれば、カーネルの既存アルファの進展を推進または抑制を表現するために、新たなアルファが加えられるだろう。例えば「要求」アルファは、全体で1つのものとして扱われるのではなく、「要求項目」毎に1つずつ進展しうる。個々の要求項目の進展が、要求アルファの進展や健全性を推進または抑制する。要求項目の種類としては、フィーチャ(feature)やユーザストーリー(user story)、ユースケース(usecase)など様々が考えられるが、いずれもアルファとして表され、状態を持ち追跡することができる。それらの細かなアルファをより粗いカーネル要素へと関連付けることで、全体として取り組みの健全性を追跡することが可能となる。これにより、個々の項目群の下位レベルの追跡に対してバランスが与えられ、チームが仕事の仕方を理解し最適化することが可能となる。

カーネルは拡張可能である。

カーネルの他の優れた特徴として、異なるプロジェクト（例えば新規開発、レガシーの拡張、社内開発、オフショア開発、ソフトウェアプロダクトラインなど）を扱えるように拡張できるという点がある。求める方法論を構築するために、ユーザストーリーやユースケース、コンポーネントベース開発、アーキテクチャ、ペアプログラミング、朝会、自己組織型チームといったプラクティスを追加できる。例えば、社内開発や委託開発、セーフティクリティカル組込みシステム開発や事務処理レポートシステム開発のそれぞれに対して、異なる手法を組み立てられる。
ここでキーとなるアイディアは、プラクティスを分離したということである。産業界で長年にわたり「プラクティス」という言葉は広く用いられているが、カーネルはプラクティスを扱い共有するための特別なアプローチを取る。プラクティスは互いに異なり分離されたモジュール単位として表され、チームがその利用を選択できる。この点が従来のアプローチとは異なる。従来のアプローチでは、ソフトウェア開発を互いに区別しがたいプラクティス群の「スープ」として扱っており、チームがある方法論から他へと移るにあたり、良いことも悪いことも投げ捨てるように仕向けてしまう。

カーネルは実践的である。

おそらくカーネルの最重要な特徴は、実践におけるカーネルの用いられ方にある。従来のソフトウェア開発方法論に対するアプローチは、プロセスエンジニアや品質エンジニアの支援に焦点をあてる傾向にあった。対してカーネルは、ソフトウェアプロフェッショナルの仕事遂行の支援に焦点をあてた実践的で「触れられる(tangible)」思考フレームワークである。例えば、カーネルはカードとして実際に手にとって用いることができる（図4参照）[7] [10]。カードは、チームメンバにとって日々のタスクを進めるにあたっての簡潔な備忘録や合図となる。実践的なチェックリストや合図があることで、概念的な議論ではなく、カーネルはチームが日々用いるものとなる。これは、従来のアプローチとは根本的に異なっている。従来のアプローチではしばしば手法の利用よりも方法論の記述が過度に強調され、それを実際に参照するのはチームに加わったばかりの人たちだけという状況になりがちである。カードは、チームメンバにとっての備忘録となる簡潔な記述を与える。チームメンバはカーネルを、ポケットに小さなカード一組として持ち、簡単に引き出して、開発の現状や仕事の割り当

図3. 基礎を成す原則

図4. カーネルはカードとして実際に手に取って用いることができる

て、メンバ間の協調作業について議論できる。チームはまた、カードを参照することで改善すべき領域を議論できる。このように、カーネルはチームがすべき事柄の長大な記述というよりも、チームが日々行っている事柄の根幹を形成する。

カーネルの利用

カーネルは、ソフトウェアプロフェッショナルの日々の活動の中で様々に用いられる。例えば以下が挙げられる。

* イテレーション（またはスプリント）の実行において。
* アイディアから製品までの開発全体の実行において。.
* 大きな組織や複雑なソフトウェア開発への取り組みへの拡大において。

カーネルは最初に、「イテレーション計画」において適用できる。この適用を、チームがカーネルを用いてできることの例として説明する。他の適用は、十分に書籍『The Essence of Software Engineering - Applying the SEMAT Kernel』[6]でカバーされている。

この例ではある企業が、形式化されたプロセスをほとんど持ち合わせていない状況を想定する。これまで同企業は、経験豊かなチームにおけるスキルを持ち創造的な個々人に頼ってきたが、現在、大きくなり新たな従業員を多くを雇いつつある。新たな従業員のほとんどは、大卒の新人であり、プログラミング言語などの技術的によいスキルを持つが、ステークホルダと協働し要求について同意を取り付けるといったソフトウェア開発上の様々な側面に不慣れである。

同企業にはアイディアや写真、コメントをユーザが共有し閲覧できるモバイルソーシャルネットワークアプリケーションを開発するためのチームがあるとする。チームは当初、カーネルに詳しいスミス（チームリーダ）とトムによって立ち上がった。その後、その仕事について初めてで、カーネルを知らないディックとハリエットという2名の開発者がチームに加わった。チームリーダのスミスにとって「成功」とは、機能やスケジュール、品質以上のものを意味する。同チームは開発をイテラティブ（iterative; 反復的）に進めた。ここで、イテレーションの計画を以下のように考えることができる。

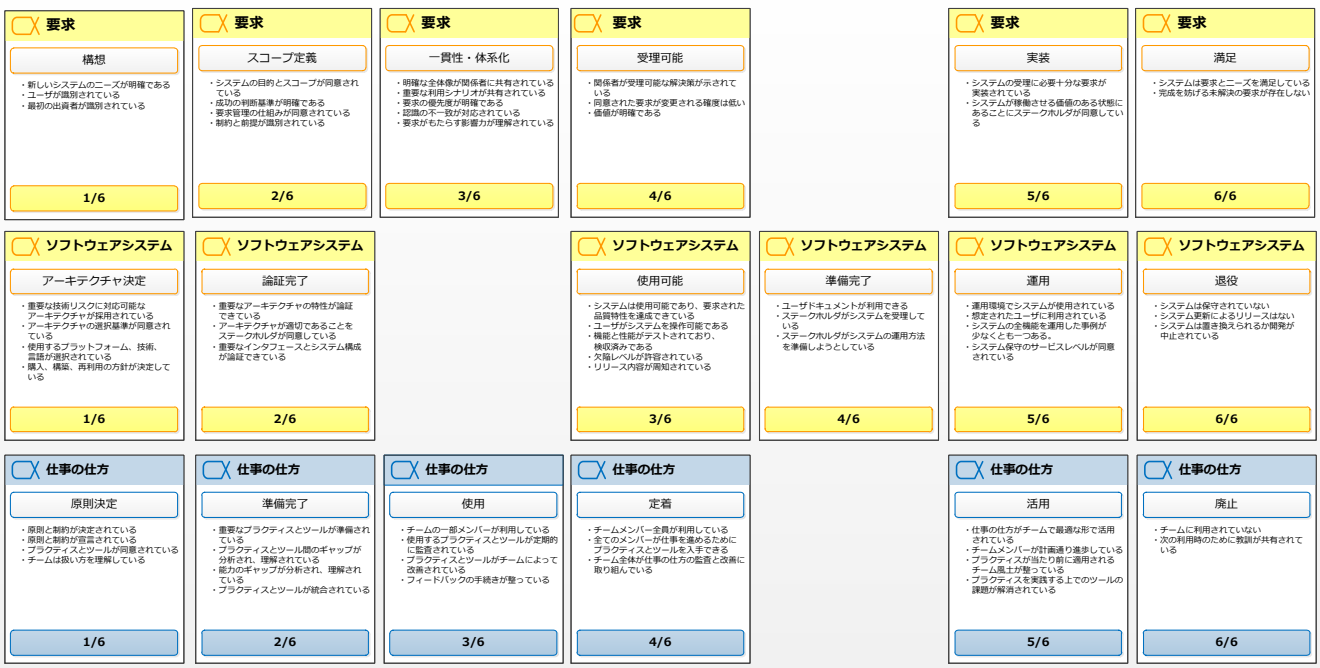
1. 現在地の特定: 取り組みの現状を分析する。
2. 行き先の決定: 次のイテレーションにおいて重視する事柄と目標を決定する。
3. 行き方の決定: 目標達成のためにチームがすべきタスクについて同意する。

カーネルによるチームの現在地の特定

スミスや彼のチームが開発を始めて6週目であるとする。彼らは既にステークホルダに対して初期のシステムデモを示し、ステークホルダは喜んで価値あるフィードバックを与えてくれている。しかし、システムはまだユーザが使える状態にはない。ここで、カーネルを様々な方法で活用できる。もしアルファの状態を示すカードを用いている場合は、以下を一通り進められる。

1. 各アルファのカードを、テーブル上で横一列に、左端に最初の状態、右端が最終状態となるように並べる。
2. 各状態を一通り読み進めて、チームにおいて当該状態に達しているかどうかを問いかける。
3. もしある状態に達している場合は、

図5. チームは現在地の特定にアルファを使用する



当該状態のカードを左側に寄せる。これを、チームが未到達の状態のカードに至るまで続ける。
4. 未到達の状態のカードおよび未検討の状態のカードを右側に移動させる。図5において、スミスのチームが達した状態のカードは左側に、達していない状態のカードは右側に位置している。簡単のため、図5ではカーネルの3つのアルファのみを示している。

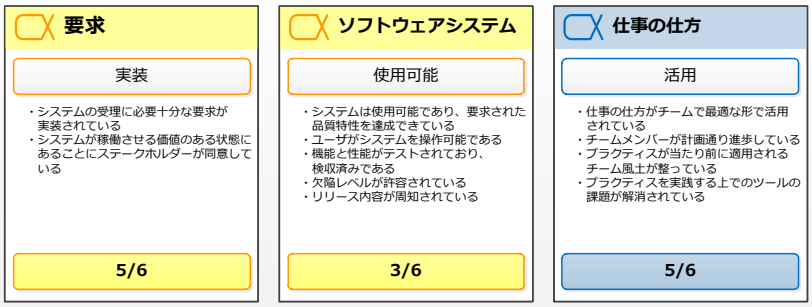
カーネルによる行き先の決定

現在のアルファの状態にチームが同意したら、以降の計画をガイドするために、メンバは次の望ましい目標状態を議論する。チームは、次のイテレーションの目標を打ち立てるために、図6に示すように、すぐ隣のアルファの状態を用いることに同意したとしよう。アルファの状態名は、当該状態に至るために必要な事柄を理解するためのヒントを与えてくれる。さらにチームメンバは、アルファの状態のチェックリストを読んで理解することでより多くを見出すことができる。各アルファを一つずつ確認することで、チームは各状態を達成するために必要な事柄をよく知るようになる。このようにして、チームはカーネルのアルファを学ぶと同時に、開発の現在の状態および次の目標となる状態を決定する。

カーネルによる行き方の決定

スミスと彼のチームは次の目標状態を確認し、優先順位付けの必要性について合意した。そこで優先順位として、まず第一に仕事の仕方が「活用」されていることとし、第二にソフトウェアシステム「使用可能可能」であることとし、最後に、要求が「実装」されていることとすることとした。その理由は簡単で、仕事の仕方がうまく活用されていなければ、ソフトウェアシステムを使用可能とすることが妨げられてしまうだろう。さらに、チームは要求満足のために不足している要求事項を実装することの優先度について同意した。
スミスと彼のチームは続いて、図6下の表に示すように、それらの状態に達するために必要な事柄を議論した。目標となるアルファの状態を確認する中で、スミスは次のイテレーションにおける目標とタスクを決定できた。

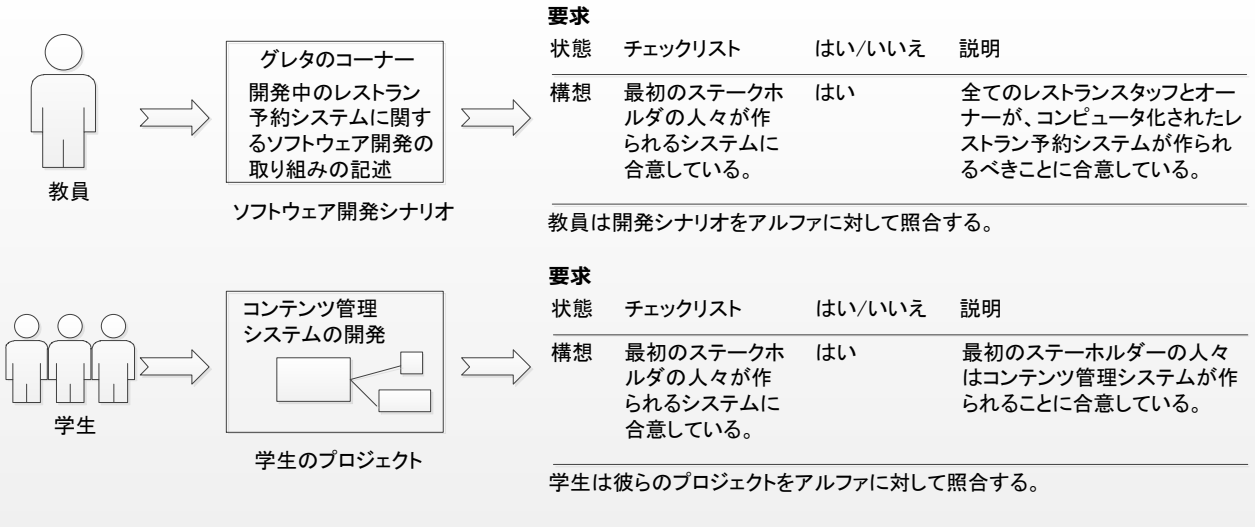
図6. 次の目標状態



チームがその状態に達するために必要な事柄

目標状態	チームにおける達成計画
仕事の仕方 活用 ・仕事の仕方がチームで最適な形で活用されている ・チームメンバーが計画通り進歩している ・プラクティスが当たり前に適用されるチーム風土が整っている ・プラクティスを実践する上でのツールの課題が解消されている 5/6	ディックとハリエットは、自動テストの適用が困難なことに同意した。彼らには進めるための支援が必要であった。そこでトムが、幾らか時間を費やして彼らに教えてくれることになった: イテレーションバックログに、ディックとハリエットに対して自動テストのトレーニングを施すというトムのタスクが追加された。 ・タスク1. 自動テストのトレーニングの実施
ソフトウェアシステム 使用可能 ・システムは使用可能であり、要求された品質特性を達成できている ・ユーザがシステムを操作可能である ・機能と性能がテストされており、検収済みである ・欠陥レベルが許容されている ・リリース内容が周知されている 3/6	この状態はつまり、ソフトウェアシステムが十分な品質と機能を持ってユーザに有効な形で示されるということであった。これまでのところ、スミスのチームは開発環境下でテストを進めつつある。ここで、未準備であった受け入れテスト環境を準備して、同環境下でテストを実施する必要があった。これは、以下のタスクをもたらすこととなった: ・タスク2. 受け入れテスト環境を準備する。スミスのチームは、システムにおいて既に論証可能となっているすべての要求事項を完了させる必要があった。これは、各要求事項が受け入れ環境下で完全にテストされなければならないことを意味する。 ・タスク3. 要求項目A「オンラインおよびオフラインに閲覧する」を完了する。 ・タスク4. 要求項目B「コメントを投稿する（オンラインおよびオフライン）」を完了する。 ・タスク5. 要求項目C「アルバムを閲覧する」を完了する。
要求 実装 ・システムの受理に必要十分な要求が実装されている ・システムが稼働させる価値のある状態にあることにステークホルダーが同意している 5/6	この状態はつまり、開発したシステムについてステークホルダが満足するようにステークホルダと協力する必要があるということであった。今回のストーリーにおいてはスミスが、顧客の代表であるアンジェラと協力して、追加実装される必要のある追加要求項目を決定する必要があった。これは、以下のタスクをもたらすこととなった: ・タスク6. システムを運用可能な形にするためにアンジェラと話して、イテレーションに収まる追加要求事項について合意する。

図7. SEMATカーネルの教育方法



イテレーション計画におけるカーネルの支援

よい計画は包括的でなければならない。包括的とは、全ての本質的な事項を含み、チーム全体をカバーしていることを意味する。また計画は、具体的でなければならない。つまり、チームにとってアクショナブルな必要がある。さらにチームは計画に対する進展を監視する方法を持たなければならない。カーネルはそれらの達成にあたり以下を支援する。

包括的: カーネルのアルファは、ソフトウェア開発の異なる次元をまたがった備忘録として機能する。従って、全ての次元をバランスよく扱う計画を策定しやすくなる。

具体的: 各アルファの状態のチェックリストは、イテレーションにおいてすべき事柄のヒントを与える。同チェックリストにより、達成した事柄を明確として、本来達成すべきと意図した事柄との比較を通じて現在の進展を特定しやすくなる。

現実世界におけるカーネル

ここで紹介されたアイデアは多くの皆さんにとって新しいものであるにもかかわらず、すでに産学両方の現実世界において適用され成功を収めている。すべての事例で Ivar Jacobson Internationalが開発したカーネルとプラクティスが使用されている。カーネルアイデアの早期導入は以下を含む。

* 再保険に関する世界的リーディング企業であるMunichReでは、ソフトウェアとアプリケーション開発のすべての領域をカバーするための協働モデル(collaboration models)を整理している。調査、基準、保守、サポートの4つの協働モデルがあり、それぞれ共通カーネル上で12のプラクティスセットから構築されている。
* Fujitsu Servicesでは、アジャイルとウォータフォールの両方の仕事の仕方を想定した「Aptツールキット(the Apt Toolkit)(訳注：Aptは「Appropriate; 適切」の意味)を初期バージョンのカーネルで構築している。
* 日本のある大手家電メーカでは、チームによる新プラクティスの適用やオフショア開発ベンダーのマネジメントを支援するために、初期バージョンのカーネル上でソフトウェアプロセスを定義している。
* KPNではイテラティブな開発(iterative development; 反復的開発) への移行の一環として13の計画に渡り300以上のプロジェクトで採用されている。さらにカーネルは結果を重視する新たな品質保証プロセスの基礎を提供し、そのプロセスは用いる方法論やプラクティスの違いに関わらずあらゆるプロジェクトに適用することができた。
* 英国政府のある省庁では、カーネルベースのアジャイルツールセットが導入されている。ツールセットの導入によりプラクティス非依存の形で、規律ある機敏さ(disciplined agility)と、プロジェクトの進展や健全性の追跡を実現している。

カーネルはすでにスウェーデン王立工科大学(KTH)で、1−2年目のソフトウェアエンジニアリングコースに利用されている。学生は1年目にプロジェクトを進めた後で、Anders Sjogrenの指導の下、プロジェクトの結果をSEMATのアルファと照らし合わせた。学生はアルファに精通し評価する機会を得て、さらに、プロジェクトの進展と健全性を考察できた。学生は2年目にMira Kajko-Mattssonにより、開発方法論に従ってプロジェクトを進めるにあたり、SEMATカーネルを使用するよう指示された。

図7に示す様にKajko-Mattssonは、ソフトウェア開発シナリオを作成し、アルファ毎にその状態と状態毎のチェックリスト項目を評価した。そして学生は、プロジェクトの運営と評価にあたり同じことをするよう指示された。

これらのコースを通した体験を通じて有益な学びがもたらされた。例えば、カーネルによって、ソフトウェアエンジニアリングのすべての本質的な側面がプロジェクトで確実に考慮されるようになる。プロジェクトの結果をカーネルのアルファに照らし合わせることで、学生は彼らの開発方法論の良い面と悪い面を容易に識別できた。またカーネルは、教育活動を最小限に抑えて、将来のソフトウェアエンジニアリングの取り組みに対する備えを学生に与えた。

カーネルの全てのアルファに従うことで、学生はソフトウェアエンジニアリングの取り組みの全範囲を習得でき、将来プロフェッショナルとしての必要とされるものを確かめることができた。

カーネルはアジャイルその他とどのように関係しているのか？

カーネルは、スクラム、カンバン、リスク駆動イテラティブ、ウォータフォール、ユースケース駆動開発、受け入れテスト駆動開発、継続的インテグレーション、テスト駆動開発といった人気の高いマネジメントや技術上のプラクティスと共に利用できる。またカーネルは、革新的なソフトウェア製品の新開発に加えて、既存のソフトウェア製品の強化や保守に取り組んでいるチームを支援する。さらには、単独から1000人強まで、あらゆるチームサイズのソフトウェアエンジニア計画を支援する。

例えば、カーネルはアジャイル宣言における「価値」を支持している。カーネルは特定のプラクティスに依存せず、チェックリストと結果に焦点を当てることで、プロセスやツールよりも、個人や個人間の対話に価値をおいている。また、プロフェッショナルなソフトウェア開発チームの必要性に焦点を当てることで、開発方法論よりも、仕事の仕方およびチームの責務遂行に価値をおいている。カーネルは、アジャイルなどの既存の開発方法論とは競合しない。むしろカーネルは、チームが選択する方法論にとらわれることはない。チームがすでに特定の開発方法論を使用している時でさえ、カーネルは有用である。Robert Martinが「The Essense of Software Engineering」の序文で指摘しているように、プロジェクトは（仮にアジャイルなものであっても）用いる方法論によらず時にうまく行かなくなるものであり、そうなった時にチームはさらに学ぶ必要がある。そのような時、カーネルは真価を発揮する。カーネルは、チームがプロジェクトを順調な状態へと戻したり、方法論を拡張したり、仕事の仕方に関する重大なギャップに対処するときに、チームがとるべき行動をガイドできる。カーネルはソフトウェアのプロフェッショナルのニーズに焦点を当て、「方法論の定義を記述すること」よりも「方法論を利用すること」に価値をおいている（従来は前者が通常であった）。

カーネルは、最新のベストプラクティスをサポートしているわけではない。膨大な量のソフトウェアがすでに開発され、保守される必要があることは認識済みである。それらのソフトウェアは何十年も稼働し、効率的な方法で維持される必要がある。これは、仕事の仕方がソフトウェアそのものと一緒に進化しなければならないことを意味し、新しいプラクティスはすでに使用されているプラクティスを補完する形で導入できる必要がある。カーネルは、長年使われた方法論を、融通の利かないウォータフローアプローチからより現代的なアジャイル、さらにはその先の方法論へと進化的な方法で移行させる仕組みを提供する。具体的には、チームのソフトウェア提供力を維持し向上させながら、長年使われた方法論をプラクティス毎に変更できる。

カーネルはあなたをどのように支援するか？

カーネルの利用により、経験豊富なソフトウェア開発者や意欲的な開発者、そして彼らが働くチームに多くの利益がもらされる。例えばカーネルは、ソフトウェア開発の取り組みの進展や健全性の診断、現在取り組んでいるプラクティスの評価、そして仕事の仕方の改善を支援する。

また、コミュニケーションの改善や、個人の容易なチーム間の移動、さらには新しいアイデアの採用を支援する。さらにカーネルはチーム、サプライヤ、および開発組織間の相互運用の改善を通じて、産業界全体を支援するだろう。

カーネルは、プラクティス非依存に方法論を定義する基盤を提供することで、カーネルは方法論の定義やプラクティスの共有の仕方を根本的に変革させる力を持っている。例えば、チームが由来の異なるプラクティスを混在および調和させて仕事の仕方を構築し改善することが可能となる。これにより、カーネルは産業界が直面している以下の重要な2つの方法論上の問題に取り組んでいる。

* チームは、もはや自分たちの方法論に固執することはない。チームは状況に応じてプラクティスを追加および削除して、絶え間なく仕事の仕方を改善できる。
* 方法論者は、もはや完全な方法論を記述するために時間を浪費する必要はない。方法論者は、新しいアイディアを簡潔かつ再利用可能な方法で容易に記述できる。

最後に、カーネルは学界(academia)にとっても有益である。カーネルは、初期の教育課程やより後の専門課程の一部として、ソフトウェアエンジニアリングの基礎的コースの構築の基礎を与え、そしてそのようなコースは特定プラクティスに関する追加コースにより補完できる。さらに、カーネルが共通の参照モデルとして機能し、さらなる研究と実験を可能にする点も同様に重要である。

Related articles on queue.acm.org
There's no Such Thing as a Free (Software) Lunch
Jay Michaelson http://queue.acm.org/detail.cfm?id=1005066
Purpose-Built Languages
Mike Shapiro
http://queue.acm.org/detail.cfm?id=1508217
Open Source to the Core
Jordan Hubbard
http://queue.acm.org/detail.cfm?id=1005064

References
1. Azoff, m. Apt methods and Tools, Fujitsu. Ovum Technology Report. Reference Code O100032-002 (Jan. 2011).
2. Fujitsu, Ivar Jacobson International AB, model driven Solutions. Essence—kernel and language for software engineering. Initial submission, 2012, version 1.0.
3. Jacobson, I. and meyer, B. methods need theory. Dr. Dobb's Journal,(2009).
4. Jacobson, I., meyer, B. and Soley, R. Call for action: The SEmAT initiative. *Dr. Dobb's Journal,* (2009).
5. Jacobson, I., meyer, B. and Soley, R. The SEmAT vision statement, (2009).
6. Jacobson, I., Pan-Wei Ng, mcmahon, P., Spence, I. and Lidman S. *The Essence of Software Engineering—Applying the SEMAT Kernel.* Addison- Wesley. (Forthcoming in Jan. 2013 but available in a prepublication version on safaribooksonline.com.)
7. Jacobson, I., Pan-Wei Ng and Spence, I. Enough of process—let's do practices. *Journal of Object Technology 6,* 6 (2007), 41-67.
8. Jacobson, I. and Spence, I. Why we need a theory for software engineering. *Dr. Dobb's Journal,* (2009).
9. Object management group (Omg). RFP: A foundation for the Agile creation and enactment of software engineering methods, (2012).
10. Pan-Wei Ng and magee, m. Lightweight application lifecycle management using state cards. *Agile Journal* (Oct. 2010)

Ivar Jacobson, the chairman of Ivar Jacobson International, is a father of components and component architecture, use cases, the Unified modeling Language, and the Rational Unified Process. He has contributed to modern business modeling and aspect-oriented software development.

Pan-Wei ng coaches large-scale systems development involving many millions of lines of code and hundreds of people per release, helping them transition to a lean and agile way of working, not forgetting to improve their code and architecture and to test through use cases and aspects. He is the coauthor, with Ivar Jacobson, of Aspect- oriented Software Development with Use Cases.

Paul McMahon is an independent consultant focusing on coaching project managers, team leaders, and software professionals in the practical use of lean and agile techniques in constrained environments. He has been a leader in the SEmAT initiative since its inception.

Ian Spence is CTO at Ivar Jacobson International and the team leader for the development of the SEmAT kernel. He has introduced hundreds of projects to iterative and agile practices as well as led numerous successful large- scale transformation projects working with development organizations of up to 5,000 people.

Svante Lidman has extensive experience building high- performance enterprise software-development teams. He has held positions at Hansoft AB, Ivar Jacobson International, microsoft, Rational Software, Objectory, among others. Since mid-2010 Lidman was the leading change agent in the largest lean/agile transition ever done in Scandinavia.
practice