Vol.2010-SE-170 No.4
2010/11/11

プロジェクトの実現可能性を検討するための負荷容量参照モデルの提案

艸薙匠[†,††]　齋藤彰儀[††]　落水浩一郎[††]

本研究では, WBS をベースとした従来型のプロジェクト計画立案とその実現可能性を検討する方法に関わる課題を整理し，それに基づいて，プロジェクトの負荷構造と組織の容量構造を定義し，負荷容量参照モデルを提案する，さらに参照モデルの事例から割り当ての自動化と実現可能性の検証を行う方法を検討する．提案する参照モデルは，プロジェクト計画の初期段階で計画の効率的検証を支援することが期待される．

Conceptual framework for a project design

Takumi Kusanagi[†,††], Akinori Saito[††], Koichiro Ochimizu[††]

This paper discusses some problems about a project planning and a verification method for the acceptance of a project in a software development organization. We propose the conceptual framework for a project design that defines an effort structure of a project and the capacity of an organization. The proposed framework supports us to verify a project plan efficiently at a early stage.

1. はじめに

まず従来のプロジェクト計画に関する問題点を整理する．

従来型のプロジェクト計画の作成において，まず WBS を作る（ワークパッケージまでブレイクダウンする）ところから始める．この際以下のような指針が一般的である．(1) ワークパッケージの抜け漏れをなくす．(2) 全てのワークパッケージを定義する．(3) 作業が具体的に理解できるまで詳細化する．(4) 担当を明確にする．(5) 進捗が定義できる．次に WBS ワークパッケージ毎に工数を見積もる．ボトムアップに合計して全体の工数を見積もる．さらに全体規模から算出するトップダウンの見積もり工数との整合性を図る．最後にスケジューリングをする（ガントチャート作成）．具体的には WBS ワークパッケージの見積もり工数と人数からガントチャートを書く．この時ワークパッケージの依存関係を考慮してプロジェクトの計画を立案する．

上記の一般的なプロジェクト計画作成方法において一番大きく，かつ現場的な問題は，「経営的視点から，組織として，このプロジェクトが達成できるのかどうか不明である」ということに尽きる．

その原因として以下の 4 つが考えられる．

- 「ある作業をできる人が何人いるのか？」を考慮する必要がある．それを加算すると，組織に必要な人数となるが，「現状の組織のメンバーで足りるのか？」という判断は難しい．通常，スキルとの関係もあり，キーマンが足りないことが一般的であり，その割り当ての難しい作業がボトルネックとなってしまう．
- 製品のアーキテクチャーの依存関係が WBS に反映されにくい. 下回りを先に作っていかないとミドルウェアが作れない，GUI の設計ができないとどんなミドルウェアを使うかも決まらない，というようなアーキテクチャーに対する依存関係を WBS に反映しないといけない．しかしこれは個別のアーキテクチャーの事情によるため，SEPG や PMO が標準化の一環で作成する標準 WBS には出にくい. プロジェクトリーダーが計画を策定する時に考慮しなければならないが，製品アーキテクチャーを計画当初から想定していないと，その依存関係まで WBS に反映することは非常に難しい.
- その製品アーキテクチャーの依存関係と関連して，要員のコミュニケーション依存関係も WBS に反映されにくい．規模が大きい開発の場合，アーキテクチャーに従って各機能チームが構成されるとすると，そのアーキテクチャ

† 株式会社 東芝
TOSHIBA
†† 北陸先端科学技術大学院大学　情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology(JAIST)

一関係によって，コミュニケーション関係も決まる部分が多い．例えば機能間のインターフェースを決める場合に，どうアーキテクチャーを切り分けるかによって，コミュニケーションオーバーヘッドは相当違ってくる．また一般的に行われる外部委託先の窓口の一本化は，設計情報の伝言ゲームを発生させることも少なくない．このようなコミュニケーションオーバーヘッドはほとんど WBS には反映されてこなかった．そのため見えない負荷オーバーの原因となっている．上述のキーマンが外部委託先の窓口となると，さらにボトルネック化しやすくなる．

- 複数プロジェクトの負荷の整合性が分からない．組織には，通常いくつかの進行中の既存プロジェクトが存在している．しかし，現在の組織に，次のプロジェクトを実行できる容量が残っているのか分からない場合も少なくない．トップダウン見積もりから算出される総人数の比較だけならば判定可能であるが，実際にガントチャートレベルになると，割り当てができなくなる場合もある．その理由は，特定の作業に対するスキルが特定の人だけとなってしまう場合が多いからである．

そのために，現場では次のような現象が発生する．

- 組織全体としては継続的に高負荷である．組織として，どのくらいまでが健全な負荷であるのかが分からないため，それ以上に負荷を積まれる可能性が高い．
- 作業が一部の人に偏りやすいが，その理由が可視化できていない．
- その高負荷を解決するための方法が，外部委託の活用などに限られている．組織内ではリソースを調整しようとするが，特に既存プロジェクトが存在する場合は非常に複雑である．最終的には政治的な人依存の調整となる．

従来組織内の調整は，ガントチャートをプロジェクトの数だけ複数組み合せることによって，特に理論的な問題はなく，制約条件に基づいた調整によって解決できるという考え方が一般的であった．しかし複数プロジェクトをまたがったリソースとタスクの依存関係が非常に複雑になるため，実際には計画時に十分調整が行えていない．

特に複数プロジェクトの負荷の整合性が分かりにくい点に大きな問題がある．組込み系製品の開発現場では，当初は単一のプロジェクト体制で組込みソフトウェアを開発するが，製品がビジネス的に成功すると，マーケティングとして複数の派生機種の開発を求められる．そのため保守体制も考慮したプロダクトライン開発体制に移行することが必要となるが，この時，機能毎にチームを構成し，複数の派生機種を開発するというマトリクス型開発組織へ移行する場合が多い．一つの派生機種に複数の機種チームが参加し，一つの機種チームは複数の派生機種の開発を担っているという構造となる．この時，プロダクトライン上のソフトウェアアーキテクチャーと，組織内チーム体制，機種別プロジェクト体制の整合性が分かりづらく，組織全体のリソースの過不足が見えにくい．ソフトウェアアーキテクチャー上，デバイス周りなど，チームのスキルはある程度固定する必要があり，可視化しづらい複数プロジェクト間のリソース調整が難しく，あるチームに負荷が集中しやすい（現実には，プロジェクトから離れて，前もって開発を進めているチームも存在する）．複数プロジェクトでの全体負荷に関しては，従来プログラムマネジメントという観点から言及されているが，十分解決策が無く，現場の SPI（Software Process Improvement）活動または，PMO（Project Management Office）活動では，手探りが続けられている．

本研究では，上記のように，WBS をベースとした従来型のプロジェクト計画立案と実現可能性を検討するにあたっての問題の整理に基づいて，プロジェクトの負荷構造と組織の容量構造の整合性を考慮に入れたプロジェクト計画の新しいモデル（負荷容量参照モデル）を提案する．さらに割り当ての自動化と実現可能性を検証する方法を検討する．提案する参照モデルは，プロジェクト計画の初期段階で計画の効率的検証を支援することが期待される．

2. 負荷容量参照モデル

本研究では，このような問題を解決するためのベースとなる負荷容量参照モデルを提案する．

2.1 全体ストラクチャー

全体は，アーキテクチャー型，WBS 型，WSB インスタンス，プロジェクトチーム，組織の 5 つの構成要素からなる．以下のように定義する．

- アーキテクチャー型とは，作るべきシステムのアーキテクチャーのタイプである．
- WBS 型とは，作業の集合の基本的なタイプであり，ひとつは新規開発のための WBS，もうひとつは派生開発のための WBS である．
- WBS インスタンスとは，実際にプロジェクトで行う作業全体である．
- プロジェクトチームとは，作業主体であり，具体的な個人の集合である．
- 組織とは，個人の集合であり，機能チームという部分集合を持っている．各プロジェクトを遂行するためのリソースがあり，具体的には組織（プロパー）と外部委託先がある．

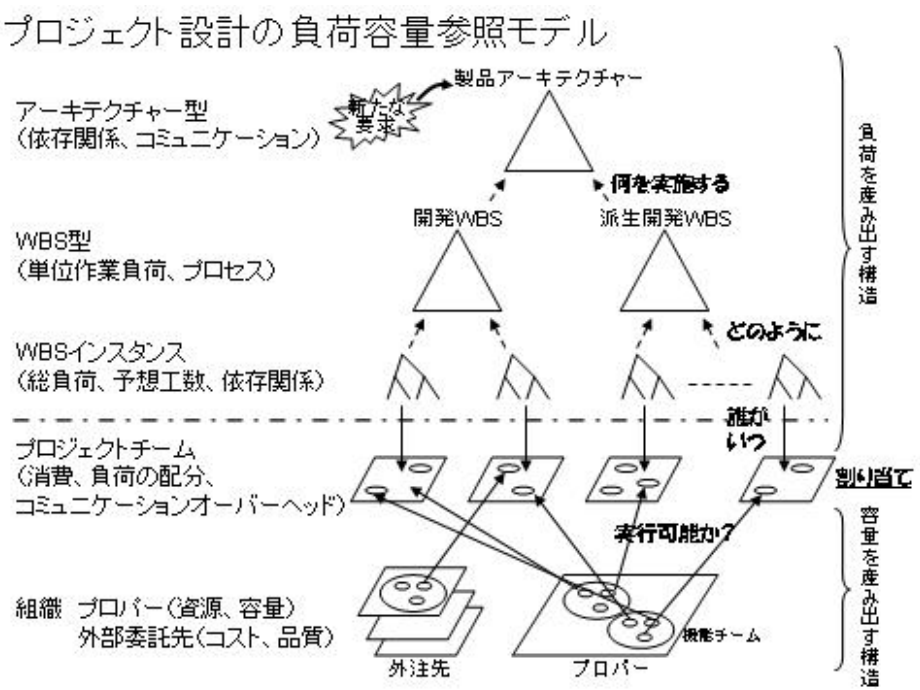

図 1　プロジェクト設計の負荷容量参照モデル

2.2 構成要素間の関係

それぞれの構成要素間の関係は，以下のように定義する．

- アーキテクチャー型と WBS 型の関係は「何を実施するか？」である．システムに対する新たな要求と，OS，ドライバ，GUI などアーキテクチャーの全体構造が，どちらの WBS 型を選ぶべきかを決める．
- WBS 型と WBS インスタンスの関係は「具体的にはどのように実施するか？」である．WBS 型から，プロジェクトに適するようにテーラリングした具体的な作業の集合が WBS インスタンスとなる．
- WBS インスタンスとプロジェクトチームの関係は「具体的に誰がいつするか？」である．WBS インスタンスが各プロジェクトに割り当てられ，作業一つ一つに対して，「プロジェクトの誰が，いつまでに行うか？」を明確にする．「割り当て」の具体的な結果が，例えばガントチャートとなる．
- 組織とプロジェクトチームの関係は「実行可能か？」である．組織はプロジェクトに人員を供給するという関係となる．「その人員供給の結果，プロジェクトが実行可能なリソースを確保できたか？」がこの関係のポイントである．

2.3 構成要素の属性

それぞれの構成要素の属性は，以下のようになる．

- アーキテクチャー型における重要な属性は「依存関係」である．プロジェクト計画の立案の結果，依存関係は，担当部門間（担当者間）のコミュニケーションオーバーヘッドを発生させる．すなわちアーキテクチャーの要素間には依存関係があり，その依存関係がプロジェクト内のコミュニケーション構造を規定する．
- WBS 型は「単位作業負荷，プロセス」という属性を持つ．WBS 型には，標準作業が埋め込まれ，個々の単位作業には，見積もりのベースとなる単位作業負荷が割り当てられている．また単位作業の一つ以上の集合をプロセスと呼ぶことにすると，WBS 型にはプロセスが埋め込まれている．一般的に，WBS 型にはアーキテクチャー型にあったアーキテクチャーによるコミュニケーションの依存関係は見えなくなっている．WBS 型は一般に SEPG や PMO，品質保証部門などスタッフが作ることが多く，標準化されている．
- WBS インスタンスは，「総負荷，予想工数，依存関係」という属性を持つ．一つの具体的なプロジェクトには，一つの具体的な WSB インスタンスがあり，プロジェクト全体の総負荷が決められる．単位作業毎に負荷の予想工数が見積もられる．ここでは，負荷の予想工数を総和し，かつコミュニケーションオーバーヘッドを足すと，総負荷になると定義する．しかし WBS インスタンスでは，コミュニケーションオーバーヘッドは見えにくい．単位作業間には依存関係がある．
- 組織は，2 種類に分かれる．プロパーは「資源，容量」という属性を持つ．資源は能力によって 2 種類に分けられ，エキスパートと一般的な技術者である．そのため，組織の容量とはエキスパートの能力と一般的な技術者の能力を足し合わせたものと言える．組織の容量は日本の場合，中期的には一定である場合が多く，急激に変化しない．そのため外部の資源の活用がポイントとなる．一方その外部の資源である外部委託先の属性は「コストと品質」である．納期を遵守させることは絶対であり，その上で外部委託先選択基準はコストである．そして納品された成果物の品質が求められる．
- プロジェクトチームは「資源の消費，負荷の配分」という属性を持つ．プロジェクトは，WBS インスタンスに割り当てた人的資源（リソース）の工数を消費し，作業が行われる．人的資源の中の時間という資源は一定であるため，その負荷を人的資源間で分散させる必要がある．そのためプロジェクトでは，負荷の配分が必要となる．負荷の配分のベースとなるものが，組織である．組織の容量の大き

さが負荷の配分を規定する．過負荷という状態や人数が多い場合のコミュニケーションオーバーヘッドなどが，組織から影響を受けて，ここで表れる．コミュニケーションオーバーヘッドは，WBS インスタンスからは出てきにくい点で注意が必要である．アーキテクチャー型と WBS 型の関係は「何を実施するか？」である．システムに対する新たな要求と，OS，ドライバ，GUI などアーキテクチャーの全体構造が，どのような WBS のタイプを選ぶべきかを決める．

全体の構造として，図 1 の上半分が負荷を生み出すもとになる構造（アーキテクチャー型－WBS 型－WBS インスタンス），下半分が容量を生み出すもとになる構造（プロパー，外部委託先）となる．その中間であるプロジェクトチームには，負荷構造と容量構造からの両方の要求を合わせて満たす必要がある．プロジェクトチームはバランスを取るという「割り当て」を現実的かつ具体的に計画し，実践する．これを一つの組織に，複数のプロジェクトで適用すると，組織容量側の実現可能性を検証することにつながる．

3. 参照モデルに基づくプロジェクト計画実現法に関する基本考察

ここでは，参照モデルの活用事例として，WBS から導出される負荷構造モデルと，組織のリソース制約から導出される容量構造モデルを示し，その二つをバインディングすることにより，妥当なプロジェクト計画立案のもととなる負荷容量図を導出する手段について考察する．

3.1 負荷構造モデル:WBS インスタンスグラフ

負荷構造は，参照モデルの上部を利用して，製品のアーキテクチャーと組織が持っている WBS 型をベースに，WBS インスタンスから作られる．

例えば図 2 の G1 において，プロジェクト計画書作成という作業は，体制図作成，プロジェクト概要作成，概略スケジュール作成の前に，各種報告会検討，ステークホルダー一覧という作業があるという構造を持つ．一番左のノードの作業を行うために，それ以外のノードの作業を行う必要があるという関係をグラフ構造で表現している．これを WBS インスタンスグラフと呼ぶ．同様に要件定義作成など WBS インスタンスで明確になった作業をグラフで表現すると，図 2 の G2，G3，G4，G5 のようになる．

作業を示すノードの属性として，「標準作業工数」と「必要スキル」を考える．ここでは必要スキルとして A，B，C を割り当てる．また標準作業工数は，ここでは全て，1 単位時間とする．これらを図 2 にあてはめると，図 3 のようになり，これを負荷グラフと呼ぶ．

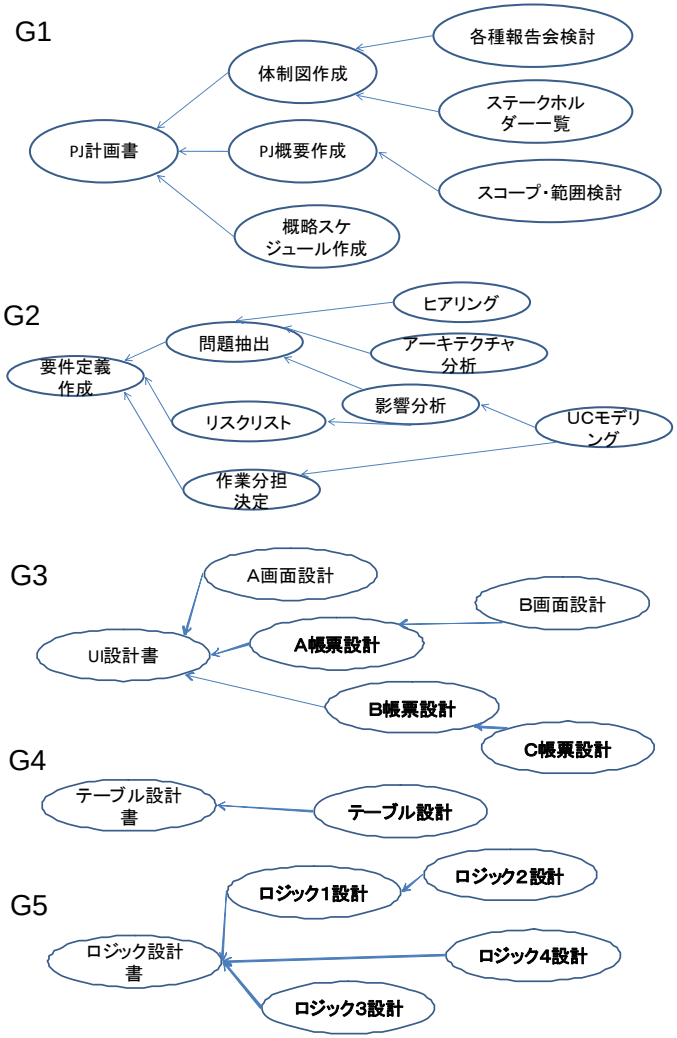

図 2 WBS インスタンスグラフ

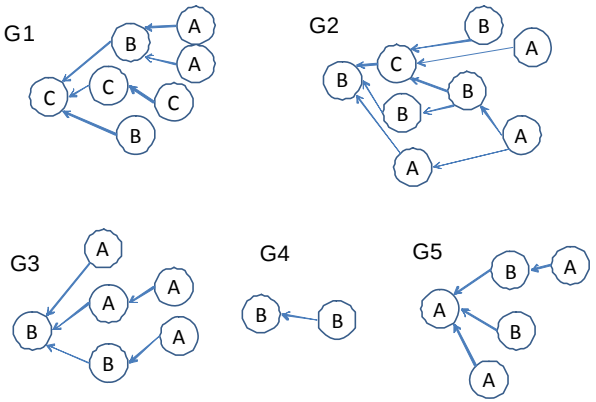

図 3　負荷グラフ

3.2　容量構造モデル:機能チームグラフ

組織の容量とは，「組織が何をどのくらいできるか？」ということを示す．参照モデルの下部にあたる組織容量は，組織を機能チームという単位に分けて考える．ここでは図 4 のように，機能チーム X と機能チーム Y という二つの機能チームがあり，チームリーダー（TL），サブチームリーダー（STL），一般開発者（担当）のメンバーが，ピラミッド構造をしていると仮定する．これを容量グラフと呼ぶ．

各メンバーに，負荷グラフの必要スキルに対応して，保持スキルを定義する．ここでは TL はスキル A，B，C を持ち，STL は A と B，一般開発者は A のみを持つとする．すると，図 4 の機能チーム X 全体では，メンバーに対する様々な作業割り当ての組合せは可能であるが，作業割り当てに制約が発生する．例えば A の作業は全てのメンバーに割り当てることが可能であり，作業を並行して進めることができる．一方 C の作業は TL1 にのみ割り当てることができるため，TL1 が C の作業をしている間，他の C の作業は引き受けられないこととなる．

このように，容量グラフにより，組織が「どのくらいの負荷を受け入れられるか」をモデル化できる．

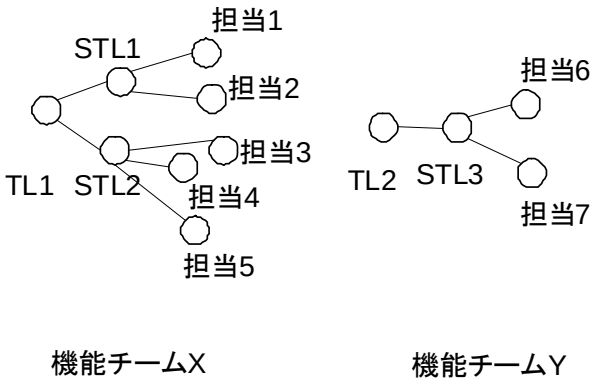

図 4　機能チーム毎の容量グラフ

3.3　負荷容量図の定義と機能チームへの割り当て

新しく発生した負荷を表現する負荷グラフと現在の組織の容量を示す容量グラフから，プロジェクト計画を立案し，組織の容量に与える影響を把握する手段を検討する．負荷グラフに対して，作業期間（通常は日単位で定義する）とリソース（人員）を割り当てることによって，作業スケジュール（例えばガントチャート）を生成することができる．この時スキル条件を満たすメンバーを割り当てることとなるが，それは組織の機能チーム毎の容量グラフで表現されたメンバーを，負荷を持った作業に割り当てることを意味する．これにより，参照モデルにおける負荷を生み出す構造と容量を生み出す構造のバインディングが達成されたことになる．

例えば，図 5 のようなリソース制約下での時間割り当てを行った場合，X チームに G1 と G2 を割り当て，Y チームに G3，G4，G5 を割り当てたとし，各作業の標準作業工数を 1 単位時間と仮定すると，X チームの工期は 4 単位時間，Y チームの工期は 6 単位時間となる．次に図 6 のように，必要スキル B と C の作業をリソース（メンバー）に割り当てる．これにより作業スケジュールが完了する．

ここで重要なことは，同時に組織側の容量を知る手段が存在することである．すなわち図 7 に示す負荷容量図は，チームメンバーの現在の負荷と残存容量を示している．

Vol.2010-SE-170 No.4
2010/11/11

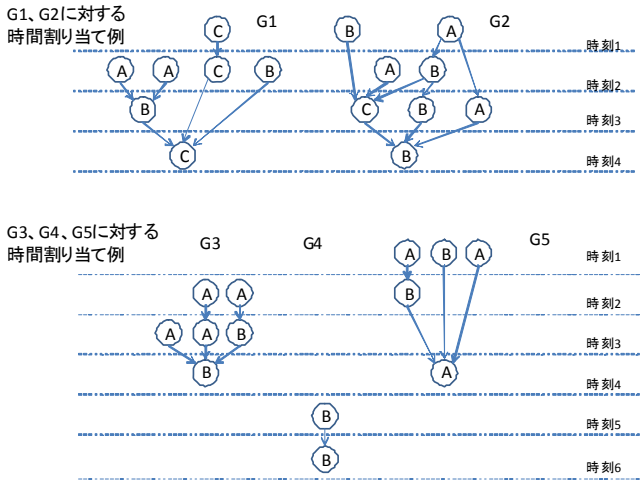

図 5　リソース制約下での時間割り当て例

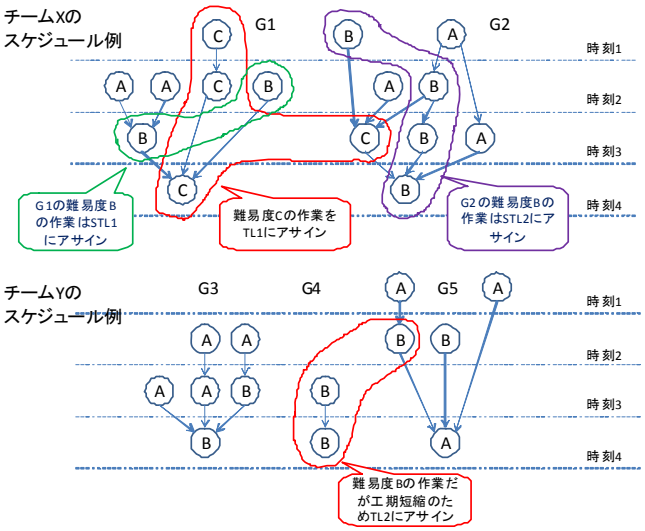

図 6　リソース制約下での作業割り当て例

図 7 の負荷容量図では，各ノードを工期分の領域に分割する．この例では図 6 に基づき，四工期に分割する．右から工期 1，工期 2，工期 3，工期 4 を示す．また負荷容量図には窓口業務（コミュニケーションオーバー業務）も記載する．これによりコミュニケーション遅延が発生する可能性があることも分かる．さらに TL1 と STL2 の間のコミュニケーション頻度が高いことも分かる．負荷容量図は負荷を割り当てられた組織の状況を可視化している．

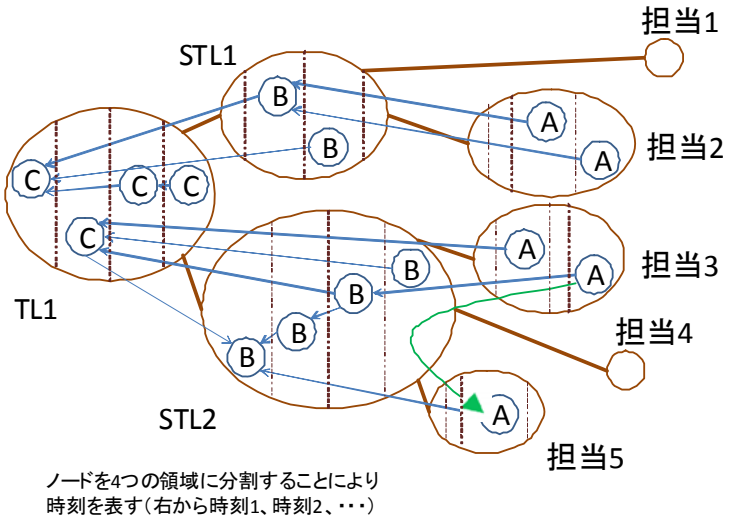

図 7　チーム X の負荷容量図

負荷容量図の形式化の効果は，以下の通りである．(1) 負荷構造モデルと容量構造モデルの割り当てのせめぎ合いから，「負荷を作るプロジェクトが，容量を持った組織で実行できるか？」を検証することができる．(2) かつ過重な過負荷を防止するために割り当てられた組織の容量を可視化することができる．(3) プロジェクトの実行可能性を機能チームへの割り当てから検討することができる．(4) さらに，このプロセスを詳細化し，ツール等により自動化することができれば，計画段階でプロジェクトの実行可能性を簡便に検証することができる．(5) また一般に組織は，一つ以上のソフトウェア開発プロジェクトを実施している状態で，次のプロジェクトの開始を検討する場合が多い．ある特定のチームの一つの負荷容量図の上に，もう一つの負荷容量

図を重ねる形で，プロジェクトの実現可能性を検討することとなる．この時重ね合わせた負荷容量図上で，メンバー一人一人の具体的な負荷予測状況を簡便に見ることができれば，リソース施策上効果的な判断を下すことができる．一つの組織が複数のプロジェクトを担当する際に組織の容量を考慮した，プロジェクト計画の実現可能性の検討が容易になる．

現在，上記の機能群を実現する土台となるアルゴリズムを整数計画法に基づいて開発中であり，機会をあらためて報告する．

4. 負荷容量参照モデルの有効性に係わる実務面からの考察

4.1 WBS による負荷と容量の考え方の違いと方法論の不足

図 1 の参照モデルをベースに考察すると，従来型のプロジェクト計画の問題点が整理できる．通常プロジェクトの割り当ては，今までこの参照モデルの上から順に流れているという方法論になっている．組織からプロジェクトへ人員を割り当てるという考え方は今まで十分なされていない．実際には日本の雇用関係ではプロジェクト毎の資源の見積もりよりも前に，「先に人がいる」という状況であり，この違いが割り当て方法に大きな違いを産む．欧米型のプロジェクト管理は，不足している場合，委託先に発注すれば良い．一方，日本の経営では「いる人を使いこなさなければいけない」という発想であり，そのため，図 1 の参照モデルの下からの理論，「容量を産み出す構造」の議論が必要である．その組織の長は，組織の容量を意識して，仕事を慎重に引き受ける必要がある．アーキテクチャー依存関係やエキスパートのスキル，コミュニケーションオーバーヘッドなどにより，スケジュールや容量の計算は，組織として単純な足し算ではない．プロジェクト視点での負荷の検討は従来から十分行われてきたが，組織長視点での容量（負荷を受ける大きさ）という点では，今まで十分検討されておらず，プロジェクトの実現可能性の検証を支援する妥当な方法論がまだ不十分な状況である．

4.2 組織のおける機能チームへの配慮不足

上述のように，プロジェクト計画時には，従来組織側の容量という視点が欠如していた．そのため機能チームは過負荷となる可能性が大きかった．組織は機能チームで構成されている．機能チームはアーキテクチャーから生まれる．技術ノウハウは一般にこの機能チームに蓄積される．そのため組織容量は個人ではなく，機能チームに依存する．エキスパートが組織容量を大きくする可能性があるからである．機能チームの運営は，プロジェクトの実現可能性にとって非常に重要である．組織長及び機能チームリーダーの視点では，「あるプロジェクトが負荷構造を引き受けた場合，機能チーム内の負荷がどのようになるのか？」が一番知りたいこととなる．3.3 で述べたバインディングの結果による機能チームへの負荷の割り当てが，プロジェクトの初期段階で簡便に可視化できれば，エキスパート育成への配慮など，従来よりも組織運営を効果的に進めることができるようになる．

4.3 複数プロジェクトでの割り当てにおける組織視点の欠如

さらに複数プロジェクトの割り当ての場合，一つ一つのプロジェクトの割り当てとは別に，組織としてのプロジェクトの実現可能性が一段と分かりにくくなってしまう．複数のプロジェクトのため，組織容量からの制約条件が複雑に絡み合い，機能チームレベルの実現可能性が分からなくなる．WBS インスタンスが同時並行的に構成されると，全体の負荷（組織からプロジェクトへ割り当てている時に，誰がどこで何をしているのか？どのくらいの仕事をしているのか？）が，特にエキスパート人材の負荷分散が見えなくなる点に大きな問題がある．つまり複数機種展開している場合，組織から見た機能チームリーダーや組織長が全体の負荷状況を把握しづらく，混乱が発生してしまう．そこに新たなプロジェクトを立ち上げる場合，計画の初期段階での適切な実現可能性判断の手法が組織側にはなく，過負荷の一因となってしまっている．

この問題は，さらにプロダクトライン実現のためのマトリクス組織では一層顕著となる．新しい開発が始まる度に，本来ならば既に複数プロジェクトに参加している機能チームの中で，その新しい負荷を吸収できるか，初期段階で簡便に判断できる必要がある．3.3 で述べたバインディングを，複数回重ねることによって，マトリクス組織での機能チームの複数開発の実現可能性を概略的に検討することができる可能性がある．

4.4 コミュニケーションオーバーヘッドのモデル化不足

負荷オーバーの原因の一つとして，見積りが困難な（または，見積りにカウントされていない）コミュニケーションオーバーヘッドがある．コミュニケーションの負荷は，計画時には考慮抜けしている．現場での現象を見ると，人員が不足しているために，委託先に業務を発注するが，そのためにかえって負荷が増える場合も少なくない．また委託先管理者を設置することもあるが，それによって，管理はしやするなる一方，設計情報が伝言ゲームになってしまうことも多く，コミュニケーションロスとなってしまう．このようなコミュニケーションオーバーヘッドがどこから来るのか，今後検討する必要がある．

コミュニケーションオーバーヘッドは，二つに分類できる[2]．伝達オーバーヘッドとネゴシエーションオーバーヘッドである．伝達オーバーヘッドは作業成果物の受け渡しや工程間での作業移行などで発生する説明のオーバーヘッドである．これは WBS 上にも表すことができる．一方ネゴシエーションオーバーヘッドは，調整や摺り合わせのためのオーバーヘッドであり，これはインターフェースなどの界面を決めるため

Vol.2010-SE-170 No.4
2010/11/11

の作業となり，アーキテクチャーの構造とその分担と関係が深い．アーキテクチャーが複雑でも，一人でやるならば，特にネゴシエーションオーバーヘッドは発生しにくい．逆に単純なアーキテクチャーでも，分担が多くなれば，ネゴシエーションオーバーヘッドは無視できないほど大きくなる．このモデル化は非常に重要であるが，どのように参照モデルの中に盛り込んでいくか，今後の課題である．

5. おわりに

本研究では，製品開発の負荷と組織の容量という考え方から，プロジェクトの計画を考える場合の負荷容量参照モデルを提案した．そしてその参照モデルをベースに，新しく発生した負荷を表現する負荷グラフと現在の組織の容量を示す容量グラフから，プロジェクト計画を立案し，組織の容量に与える影響を把握する手段を検討した．またその応用として，プロダクトラインで複数機種を開発する場合に，どのような課題があるかを明らかにし，それがこの参照モデルの中でどのように位置付づけられるかを考察した．今後は，負荷と容量の割り当てアルゴリズムの開発やコミュニケーションオーバーヘッドのモデル化を行い，その上で，複数機種を開発する場合に，ある容量を持った組織全体の負荷の可視化，個々の機種開発のスケジュール調整の手法を考案していきたい．

参考文献

1) 齋藤彰儀, 落水浩一郎: 要求追跡ネットワークを用いたプロセス改善支援およびプロジェクトスケジューリング法，信学技報， SS2009-49，2010
2) 門脇千恵，落水浩一郎，“非同期分散型会議の事象駆動型討議プロセスによるモデル化と調整支援への応用”, 情報処理学会ソフトウェア工学研究会, ソフトウェア工学 96-25, pp.193-200, 1994.
3) 池上俊也: WBS の作り方，日経システムス，日経 BP 社（2010）
4) Gregory T.Haugan: 実務で役立つ WBS 入門，翔泳社（2005）
5) S・M・デイビス，P・R・ローレンス，津田達男，梅津祐良(訳),”マトリックス経営-柔構造組織の設計と運用”，ダイヤモンド社，東京，1980.
6) ウォーカー・ロイス，日本ラショナルソフトウェア（株）監訳,”ソフトウェアプロジェクト管理”，ピアソン・エデュケーション，東京，2001.
7) Project Management Institute, "A Guide to the Project Management Body of Knowledge", Project Management Institute, Pennsylvania, 2008.