修 士 論 文

A Project Scheduling Method and Tool based on Load-Capacity Model

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

DOUNGTHAPHET PONPRUD

2011 年 9 月

修 士 論 文

A Project Scheduling Method and Tool based on Load-Capacity Model

指導教官　落水浩一郎教授

審査委員主査　落水浩一郎教授
審査委員　鈴木正人 准教授
審査委員　青木利晃 准教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

0910208 DOUNGTHAPHET PONPRUD

： 2011 年 8 月

概要

複数のソフトウェア開発プロジェクトがすでに実行されているとき、新しいプロジェクトの実現可能性を検討するには、製品開発の作業負荷だけでなく、組織の残容量を考慮する必要がある。この考え方に基づき、艸薙が負荷容量参照モデルを提案した。また、齋藤がこのモデルにおける、スケジューリング問題を整数計画問題として定式化し、負荷容量参照モデルに基づくプロジェクトスケジューリング法を提案した。しかし、これらの手法を活用するためには、手法を支援するツールが必要である。本論文では、負荷容量参照モデルに基づくプロジェクトスケジューリング法を支援するツール、 LCB(Load-Capacity Balancer) の機能定義、設計結果を示し、その一実装例について述べる。開発したツールは、ユーザが入力した負荷グラフ・容量グラフ・目的関数に関するデータを使って、負荷グラフと容量グラフを作成し表示する。さらに、ソルバとして GLPK を利用して、作業の開始時刻とリソース割り当てを行い、負荷容量図とガントチャートを作成し表示する。

目次

図目次

表目次

第 1 章 はじめに

近年の組込みソフトウェア開発プロジェクトは，長期の開発期間にもかかわらず，適切な市場投入時期に合わせて計画される。そのため，複数のプロジェクトが並行して実施される。この状況下にて，新規プロジェクトの計画作成を行う時，実行可能なスケジュール案を得るには，リソース（開発要員）の現状の負荷状況や，組織全体の残容量を把握しておく必要がある。

この考え方に基づき，艸薙が，プロジェクトの計画立案時にてプロジェクトの実現可能性を検討するためのモデルである負荷容量参照モデルを提案した[1],[2]。このモデルでは，プロジェクト計画立案時に，作業スケジュール（ガントチャート等）だけでなく，負荷容量図（作業負荷が割り当てられた組織の状況等を可視化した図）を用意する。これにより，リソースに関する検討及び，組織の残容量に対する負荷の状況を把握することができ，新規プロジェクトの実現可能性を検討することができる。

また、齋藤が負荷容量参照モデルにおける、負荷容量参照モデルに基づくプロジェクトスケジューリング法[3]を提案した。負荷容量参照モデルに基づくプロジェクトスケジューリング法においては、作業負荷を表す負荷グラフと，組織容量を表す容量グラフを入力として、スケジューリング問題を整数計画問題として記述して、計算する。この手法の出力は，作業スケジュールと負荷容量図である。しかし、負荷容量参照モデルに基づくプロジェクトスケジューリング法を使うために、整数計画問題に関する数学的な専門知識が必要である。つまり、普通の人が使えるために、この手法を支援するツールが必要である。

本論文では、負荷容量参照モデルに基づくプロジェクトスケジューリングツールについて提案する。本ツールの入力は三つあり、負荷グラフのデータ、容量グラフのデータ、目的関数である。本ツールの負荷グラフのデータは WBS （Work Breakdown Structure）のデータベースからデータを照会して、入力できる。また、容量グラフのデータも組織のデータベースからデータを照会して、入力できる。出力は負荷容量図とガントチャートである。

本論文は 6 つの章よりなり、その構成は以下のとおりである。第 1 章、「はじめに」、では研究の背景・関連研究・本研究の目的について述べる。第 2 章では、本研究のツールの機能仕様を定義する。第 3 章では, ツールの設計結果について述べる。第 4 章では, ツールの

実現方法を述べる。第 5 章では, 開発したツールの利用方法を説明する。最後に第 6 章において, 本研究のまとめと今後の課題について述べる。

1.1 研究の目的

本研究の目的は普通の人（プロジェクトマネージャー）が負荷容量参照モデルに基づくプロジェクトスケジューリング法を使えるために、負荷容量参照モデルに基づくプロジェクト計画作成支援ツール、Load-Capacity Balancer (LCB)を開発することである。そのために、ツールは以下の機能を具備する必要がある。

- 負荷グラフ・容量グラフ・目的関数の入力ユーザインタフェース
- 負荷グラフ・容量グラフ・負荷容量図・ガントチャートを自動的に作成する

1.2 現在のプロジェクトスケジューリングソフトウェア

現在世の中で色々なプロジェクトスケジューリングソフトウェアがある。例えば、最もよく利用されているのは「Microsoft Project」である。

1.2.1 Microsoft Project

Microsoft Project（マイクロソフト プロジェクト） [3] は、マイクロソフトが Windows 向けに販売しているプロジェクト管理ソフトウェアである。ガントチャートやクリティカルパス法を利用して計画の見積もりや予算の状況などを把握し、一つのプロジェクト状況を一元的に管理する事ができるソフトウェアである。Microsoft Project はこれらのスケジュールや予算、人材の状況をプロジェクト管理ソフトで詳細に分析し、統括的な行程具合を管理･指揮する事ができるようになる。

Microsoft Office Project は機能や用途によっていくつかのエディションが用意されている。プロジェクトマネージャが個人で仕事を進めるためのスタンドアロン版は「Standard」エディションと呼ばれる。組織でプロジェクト管理を行うためには「Server」エディションでサーバを構築し、「Professional」エディションから接続して利用する。Professional は機能的には Standard とほぼ同様である。「Web Access」エディションを利用すれば Web ブラウザ

からサーバを利用することもできる。プロジェクトのポートフォリオ分析に特化した「Portfolio Server」エディションがあるが、日本語版では用意されていない。

最新版の Microsoft Project 2010 は、全世界で 2000 万人が利用する世界標準のプロジェクト管理ツールである。Project 2010 を利用する事で、正確な計画の作成、リアルタイムな進捗管理、プロジェクトメンバーの負荷の調整、プロジェクトの遅延時の効果的なリカバリーが可能になる。また Project Server 2010 を使用すると、全社の全てのプロジェクトを統合管理やプロジェクト・ポートフォリオ管理が実現致する。小規模なチームのワークマネジメントから、企業横断プロジェクトのエンタプライズ・プロジェクト管理まで全てのユーザのニーズを実現するプロジェクト管理が Microsoft Project 2010 である。

Microsoft Project の特徴 (Advantage) は、以下のようである。

- 使いやすい
  多数のマネージャーが Microsoft Project を利用する理由は使いやすさにある。短期間で利用方法を勉強、また、分かりやすい出力を提供する。
- 他の Microsoft の製品と一緒に使える
  例えば、他の Microsoft 製品 MS Word・MS Excel・MS Outlook などである。MS Project が他の Microsoft 製品を統合する機能を MS Project は持っている。
- MS project は「Desktop Application」である
  「Desktop Application」の特徴はインターネットがなくても使えるということである。

Microsoft Project には色々な特徴があるが、欠点もある。例えば、以下が挙げられる。

- 組織構造の可視化が難しい
- プロジェクトの実現可能性が検討しにくい
- ガントチャートは自動で作れない
- プロジェクトの規模が大きい場合、入力にコストがかかる。

これ等の欠点は Microsoft Project だけでなく、多数のプロジェクトスケジューリングソフトウェアが共通に持つ欠点である。これらの欠点を解決するために負荷容量参照モデルが考察された。

1.2.2 Red Mine

Redmine [11] は web ベースのプロジェクト管理ソフトウェアである。Redmined では、プロジェクトのタスク管理、進捗管理、情報共有が行える。また、Subversion や Git などのバージョン管理システムとの連携機能も備えており、ソフトウェア開発や web サイト制作などの IT プロジェクトで特に威力を発揮する。

Redmine はプログラミング言語 Ruby 本体の開発を始めとして、さまざまな企業やプロジェクトでの導入が報告されている。また、Redmine はオープンソースで GNU General Public License v2 (GPL)で配布されており、その設計は Trac に影響を受けている。Trac はオープンソースで web ベースのソフトウェアであり、プロジェクト管理とバグ追跡のためのツールである。

Redmine の特徴はおもに次の通りである。

- 複数プロジェクトを管理することが可能である。
- ガントチャートやカレンダ表示が可能
- 課題（Issue）にひも付いたコメントや、関連する課題、ファイル、ディスカッション、Wiki などの情報を蓄積できる。これにより、それぞれの課題に関する対応の経過や関連資料の把握、行われた議論の内容を必要に応じて入手することができる。蓄積した内容の検索機能も持っているため、情報を投入すればするほど有益なナレッジベースとしての価値も上がっていくことも想定される。
- バージョン管理システムとの連携が可能である。Web 画面上からリポジトリを閲覧できるほか、コミット時のコメント内容から課題とソースコードを関連付けたり、自動的に課題の状態（ステータスや進捗%）を変更するような設定が可能である。ワークフローと組み合わせて使うことにより、修正と承認の制限をかけるような使い方も可能である。

Redmine にはこのような特徴があるが、プロジェクトスケジューリングソフトウェアが共通に持つ欠点がある。例えば、組織構造の可視化が難しいことやプロジェクトの実現可能性が検討しにくいことやガントチャートが自動で作れないことなどである。これらの欠点を解決するために負荷容量参照モデルが考察された。

1.3 負荷容量参照モデル

新規プロジェクトの実現可能性を検討するためのモデルとして，負荷容量参照モデルが提案された[1], [2]．製品開発の負荷を，組織の残容量を考慮しつつ，開発時間軸に割り付けるという考え方である．

1.3.1 全体構造

負荷容量参照モデルは大きく 5 つの構成要素（アーキテクチャ型，WBS 型，WBS インスタンス，プロジェクトチーム，組織）からなる（表 1 と図 1.1）。「アーキテクチャ型－WBS 型（Work Break Down Structure）－ WBS インスタンス」が負荷を生み出すもとになる構造（負荷構造），「プロパー，外部委託先」が容量を生み出すもとになる構造（容量構造）となる。検討対象となる新規プロジェクトチームは，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。

構成要素	属性	分類
アーキテクチャ型	依存関係，コミュニケーション	負荷を生み出す構造
WBS 型	単位 WP 負荷，プロセス	
WBS インスタンス	負荷，標準予想工数，必要スキル，先行制約	
プロジェクトチーム	資源の消費，負荷の配分，コミュニケーションオーバーヘッド	
プロパー組織	資源，容量，保持スキル	容量を生み出す構造
外部委託先	コスト，品質，保持スキル	

表 1.1 負荷容量参照モデル

- アーキテクチャ型

アーキテクチャ型とは，作るべきシステムのアーキテクチャの構造的特徴を表したものである。例としては，クライアントサーバ型等がある。属性は，依存関係とコミュニケーションである。一般に，システムのアーキテクチャに従って機能チームが構成される。アーキテクチャ構成要素間には同時に考慮すべきものがあり，これは，リソース間のコミュニケーションを発生する。

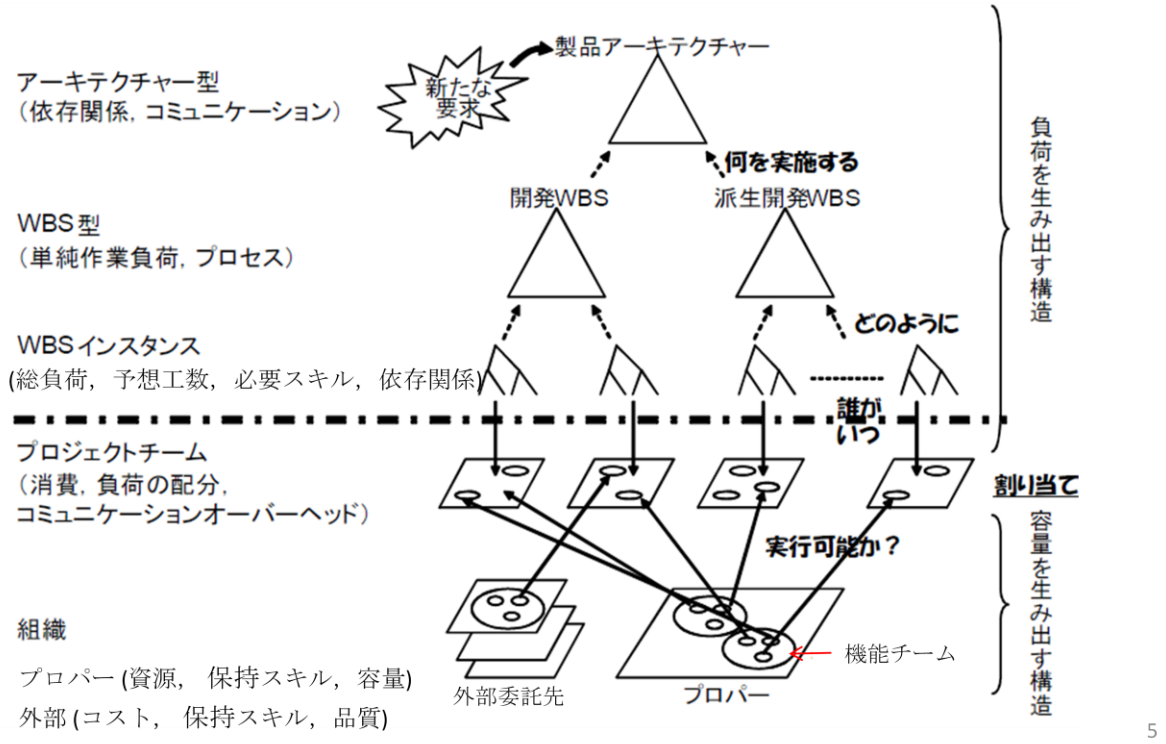


図 1. 1 負荷容量参照モデルの概念

- WBS 型

WBS 型とは，基本的な作業（ワークパッケージ，以下 WP）の集合を表したものであり，新規開発のための WBS 型と派生開発のための WBS 型がある。これは，各組織のプロセス改善グループ等が作成している場合が多い。属性は，WP (Work Package) ごとの単位負荷（単位 WP 負荷）とプロセスである。個々の WP には，見積もりの基準となる単位 WP 負荷を持つ。また，プロセスとはひとつ以上の WP の集合であり，WBS 型はプロセスを部分集合として持つ。

- WBS インスタンス

WBS インスタンスとは，WBS 型から導かれた WP の集合であり，実際にプロジェクトで行う WP 全体と対応している。属性は，総負荷，標準予想工数，必要スキル，先行制約である。負荷容量参照モデルでは，各 WP の標準予想工数の総和にコミュニケーションオーバーヘッドを足したものを総負荷と呼ぶ。ただし，WBS インスタンスにはコミュニケーションオーバーヘッドは表れにくい。標準予想工数は単位 WP 負荷より導かれる。WP 間には，

依存関係やプロセスから導かれた先行制約が存在する。スキルについては，1.3.2 にて説明する。

- プロジェクトチーム

プロジェクトチームとは，作業主体であり，開発に携わる人々の集合である。属性は，資源の消費，負荷の配分とコミュニケーションオーバーヘッドである。納期やスキル等の関係により，負荷（WP）をリソースに配分（割り当て）して開発を行う。これは，リソース（容量）を消費していることになる。

- 組織

組織とは，個人の集合であり，プロパー組織と外部委託先からなり，それぞれ機能チームという部分集合を持つ。組織は，プロジェクトを遂行するためのリソースとして，各プロジェクトに機能チームを提供している。プロパー組織の属性は，資源，容量と保持スキルである。外部委託先の属性は，コスト，品質と保持スキルである。負荷容量参照モデルでは，保持スキルの数により資源を分類する。複数の保持スキルを持つリソースを高資源と考える。

1.3.2 スキル

各 WP に属性として与えられているスキルのことを必要スキルと呼び，各リソースが持つスキルのことを保持スキルと呼ぶ。本稿でのスキルとは，能力とは別の概念であり，種類に近い概念である。スキルの例：「〇〇システムの開発経験がある」，「××ソフト設計のピアレビューができる」,「JAVA でプログラムが書ける」等。能力の例：ピアレビューを「A さんは 3 日でできる」，「B さんは 1 日でできる」等。負荷容量参照モデルでは，必要スキルと保持スキルが一致する場合，WP をリソースに割り当てることが可能となる。能力は割り当てに影響しない。

1.3.3. 負荷容量参照モデルに基づく基本概念の定義と仮定

本章では，スケジューリングを行う上で必要な概念の定義と仮定について，具体例（ミドルウェアの派生開発[1]）を用いながら述べる。

本論文でのスケジューリングとは，各 WP に開始時刻とリソースを割り当てることと定義する。時間の単位は，半日や 1 時間といった，ある一定の時間間隔（タイムスロット）とする。リソースはプロジェクト要員（人）とする。各 WP に開始時刻を割り当てることを，「時刻割り当て」と呼ぶ。また，各 WP にリソースを割り当てることを，「リソース割り当

て」と呼ぶ。各 WP には一人のリソースを割り当てる。リソースには同時刻に 2 つ以上の WP を割り当てない。WP 数は，それを実行するリソース数より多いものとする。

WBS インスタンスのことを本稿では負荷グラフと呼ぶ。負荷グラフは，負荷容量参照モデルの負荷構造より導かれ，有向グラフで表される。各頂点は WP を表し，有向辺は WP 間の先行制約を表す。各 WP には，標準予想工数（標準予想タイムスロット数）と必スキルが属性として与えられているとする。既存プロジェクトが実施中に新規プロジェクト計画を作成する際は，新規プロジェクトの負荷グラフと既存プロジェクトの残りの負荷グラフをひとつの負荷グラフとして考える。G を負荷グラフとする。

- 頂点集合 G(V) = {vi ; i =1, 2, · · · , n} は，WP の集合を表す。
- 辺集合 G(E) ={(vi, vj) ; i, j = 1, 2, · · · , n} は，WP 間の先行制約を表。
- 各 WP の標準予想工数は，SEH = {hi ; i =1, 2, · · · , n} と表し、
- スキルは，= { θ s, ; s =1, 2, · · · , skl}, (skl <= n) と表す。

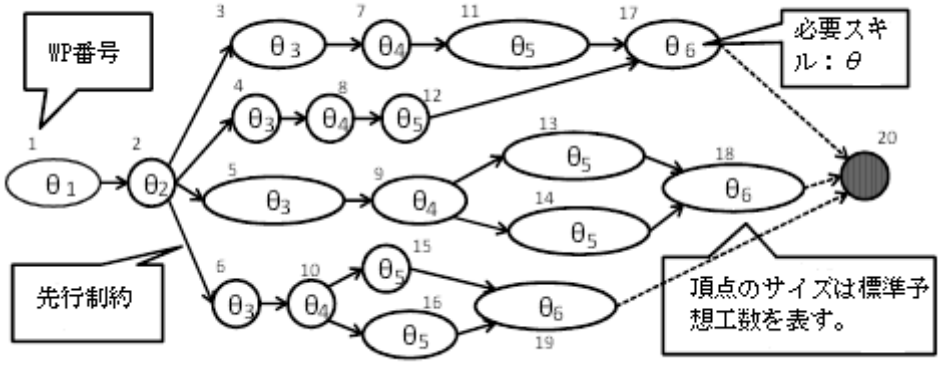


図 1. 2 負荷グラフ G1

派生開発 WBS 型の例（図 2）より得られた負荷グラフの例を図 3 に示す。この例では，タイムスロットを 1 週間とし，WPv2, v5, v6, v8, v12 の標準予想工数を 1 週間，WPv1, v3, v7, v13, v14 を 2 週間，その他を 3 週間としている。また，必要スキルとして θ 1 ～ θ6 が与えられている。これは，図 2 における WBS ワークパッケージ名に対応している。v20 は定式化する上で必要なダミーノードである。

負荷グラフが，負荷容量参照モデルの負荷構造より導かれるのに対し，容量グラフは負荷容量参照モデルの容量構造より導かれる。これは組織構造・体制を表したものである。頂点は具体的なリソースや役割を表し，辺は組織上の関係を表す。容量グラフは H で表すことにする。容量グラフの各頂点は，保持スキルを 1 つ以上持つ。

残容量については，各時刻 l ごとに考慮した，以下に示す残容量関数を用いて表す。時刻 l における保持スキル θ s のリソース数を Nl(θ s) と表す。同様に，時刻 l における割り当て可能なリソース数を Nl と表す。また，時刻 l におけるリソース r の残容量は，WP が割り当てられていないとき 1，そうでないとき 0 をとる関数 f(r, l) ∈ {0, 1} を用いて表す。

	ネットワークミドルウェア						DB ミドルウェア					
	要求仕様差分まとめ θ_1	計画立案 θ_2	差分設計 θ_3	ピアレビュー θ_4	実装・単体テスト θ_5	組合せ/統合テスト θ_6	要求仕様差分まとめ θ_7	計画立案 θ_8	差分設計 θ_9	ピアレビュー θ_{10}	実装・単体テスト θ_{11}	組合せ/統合テスト θ_{12}
MTL1	○	○	○	○	○	○				○		
MSTL1	○	○	○	○	○	○						
担当 M1-M2			○	○	○	○						
MSTL2							○	○	○	○	○	○
担当 M3-M5									○	○	○	○

表 1. 2 ミドルウェアチームのスキル表

図 4 に例（ミドルウェアチーム）を示す。このチームは，表 2 に示す保持スキル表を持っている。時刻 l のとき，誰にも WP が割り当てられていなければ $N^l(\theta 1) = 2, N^l = 8$, f(MTL1, l) = 1 等となる。

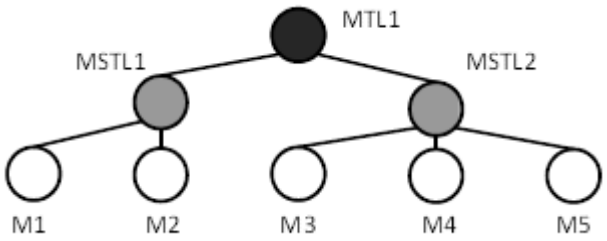


図 1. 3 容量グラフ H_m

従来のスケジュール作成の代表的な技法に，CPM，CCPM 等がある。これらは，先行制約及び各 WP の見積り結果にてスケジュールネットワーク分析を行い，次にリソースを割り当て，その後調整（再分析）という手順となっている[7]。

負荷容量参照モデルでは，製品開発の負荷だけでなく，組織の残容量を考慮する。つまり，リソース制約を常に考慮しながらスケジューリングする。

実現可能性検討のため，同一の入力に対し様々な出力結果が要求される。これは，リソース数と開発期間のトレードオフポイントを列挙することになる。ただし，リソースを先に割り当てると，与えられた開発期間内で時刻割り当てができない可能性がある。従って時刻割り当てを先に行う。

1.4 負荷容量参照モデルに基づくプロジェクトスケジューリング法

負荷容量参照モデルに基づくプロジェクトスケジューリング法 [4][5] は，入力を，負荷グラフとして表現される作業負荷と容量グラフとして表現される組織容量（残容量）とし，出力を，作業スケジュールと更新された負荷容量図とする問題である（図 1）。

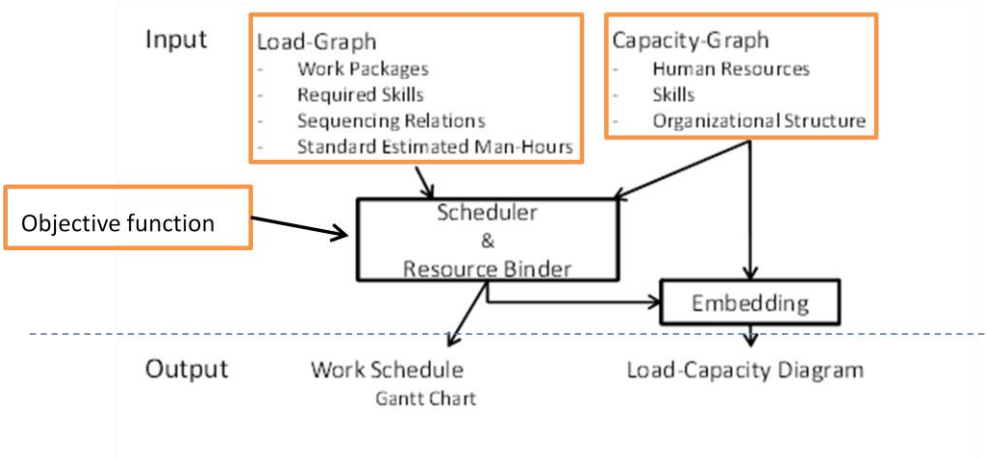


図 1.4 負荷容量参照モデルに基づくプロジェクトスケジューリング法の全体構造

この問題は，リソース制約下での作業の時刻割り当て（作業の開始時刻を決定すること）と具体的なリソース割り当て（作業を実行するリソースを決定すること）からなる。一方，負荷容量図は，得られた作業スケジュールと容量グラフからグラフ埋め込み[6] を適用して作成する。

リソース制約下での（複数）プロジェクトスケジューリング問題に対する発見的解法は，従来から数多く研究されている。これらは条件に特化しているため高速ではあるが柔軟性にやや欠け，複数の計画案を作成しようとする際にコストが発生しがちである。

このスケジューリング問題を整数計画問題として定式化する。整数計画法は，柔軟性の高い理論的体系であり，本稿で扱う，残容量を考慮した複数のスケジュール案を作成する問題に適している。整数計画法の持つ特徴を以下にまとめる。

- プロジェクトの様々な制約式を整数計画問題の制約式として記述可能
- 目的関数，制約式を変えることにより，複数の作業スケジュール案を容易に生成可能
- 与えられた制約を満たす最適解を出力

- ソルバや計算機の性能向上により，従来より高速に解が得られる

その他の特徴として，要求工学での実績もある。これらは各要求に対し，収益や要求間の依存関係，ステークホルダー，リソース制約等を考慮し，収益を最大化する計画を立案する方法である。また，ハードウェア分野での実績もある[13]。

一方，大規模な入力事象に対して，整数計画問題を解くことは，計算量的には困難であると考えられる。齋藤　彰儀の稿では，文献[1], [9] を元にした例題にて計算機実験を行い，現実レベルの問題においては十分実用可能であることを示す。

1.4.1 負荷容量図

負荷容量図とは，WP が割り当てられた組織の状況を可視化した図である。負荷容量図を作成することにより，作業負荷だけでなく，コミュニケーション伝達業務等の作業以外の負荷状況も可視化可能となる。以下に挙げる評価関数とともに検討することにより，リソース割り当てを再考することや，次期プロジェクトをどう吸収するか，プロジェクト管理上の重点部分，あるいは人材育成といった様々な視点で検討することができる。

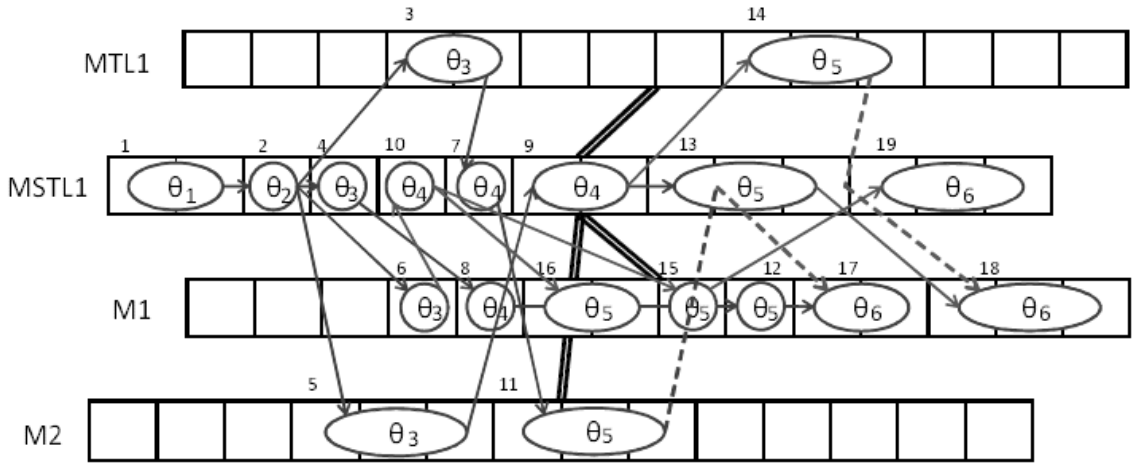


図 1. 5 負荷容量図

負荷グラフ G，容量グラフ H に対し，G から H へのグラフ埋め込み⟨φ, ρ⟩ とは，G の頂点集合 V (G) から H の頂点集合 V (H) への写像 φ : V (G) → V (H)と，G の辺集合 E(G) から H のバス集合 P(H) への写像 ρ : E(G) → P(H) の 2 つの写像の対として定義される。ただし，G の辺 e

= (u, v) に対し，ρ (e) は φ(u) と φ(v) を両端点とするバスとする。また，評価関数は以下となる。

- コミュニケーション遅延：G の辺が埋め込まれる H のバスの辺数をコミュニケーション遅延という
- 頂点負荷：H のある頂点 ui に埋め込まれる G の頂点数を ui の頂点負荷という
- 窓口度：G の辺が埋め込まれる H のバスの中で，H の頂点 ui を内点（端点ではない頂点）として含むバスの数を ui の窓口度という
- コミュニケーション頻度：G の辺が埋め込まれる H のバスの中で，H の辺 fj を含むバスの数を fj のコミュニケーション頻度という

1.4.2 計算機実験

負荷容量参照モデルに基づくプロジェクトスケジューリング法法の実行時間を評価するために，いくつかの負荷グラフと容量グラフ，および負荷グラフと開発期間の組に対して計算機実験を行った。齋藤はソルバとして GLPK(GNU Linear Programming Kit) ver.4.45 [8] を用いた。

GLPK (GNU Linear Programming Kit) は線形計画問題を解くためのルーチンを集めたライブラリーである。これらのルーチンはシンプレックス・アルゴリズム、分枝限定法アルゴリズム、主双対内点法アルゴリズムやその他多くのアルゴリズムを実装している。

第 2 章　Load Capacity Balancer の設計と実現

2.1 ツールの設計方針

負荷容量参照モデルに基づくプロジェクトスケジューリング法が提案されたが、この手法を使うために以下の問題がある。

1) 方法論を理解する必要がある

負荷容量参照モデルに基づくプロジェクトスケジューリング法を理解するためには、負荷容量参照モデルや整数計画法を勉強しなければならない。

2) データ入力が難しい

図 1.4 により、負荷容量参照モデルに基づくプロジェクトスケジューリング法の入力は大きく分けて 3 つでがあり、負荷グラフ、容量グラフ、目的関数である。これらのデータは様々なデータセットを保持しているため、入力は負担となる。

- 負荷グラフのデータ
  負荷グラフのデータは WP・先行制約・工数・必要スキルえある。WP は数十個入力する必要がある。
- 容量グラフのデータ
  容量グラフのデータはリソース・組織の構造・保持スキルのセットである。しかし、一つのチームには 5 から 10 のリソース（人）により、構成されている。
- 目的関数のデータ
  目的関数の入力は一番困難な作業である。その原因は、プロジェクトや組織の方針を表現する要があるからである。

3) 出力データからガントチャートと負荷容量図を作成することが難しい

三つの入力データとソルバ（GLPK）からの出力から、ガントチャートと負荷容量図を作成することは難しい。

2.1.1 設計方針

上記問題を解決するために、以下のような設計方針を設定した。

1. 入力しやすく、分かりやすいユーザインタフェースを持つ

(1) 入力しやすい

データ入力が容易であるユーザインタフェースを作るためには、できるだけユーザの仕事を最小限にとどめることが必要である。そのために、以下のような方針を設定した。

- 2 種類データベースを準備する

  ひとつは組織構造に関するデータベースである。このデータベースはプロパー・外部委託先・機能チーム・社員・スキルのデータを保持する。これによりユーザは、容量グラフのデータをこのデータベースから照会できる。そして、目的関数は照会した容量グラフのデータを用いて、入力する。

  もうひとつは負荷構造に関するデータベースは、新規 WBS 型・派生開発 WBS 型・WBS インスタンス・WP のデータを保持する。これによりユーザが負荷グラフのデータをこのデータベースから照会できる。

  データベースを準備することにより、ユーザの入力を軽減させる。

- 入力しやすい方法を用いる。

  もし、ユーザが入力しなければならない部分があれば、入力しやすい方法を用いる必要がある。例えば、表やドロップダウンコンボボックスやラジオボタンなどを利用する。

(2) ツールの出力はできるだけ、絵や図や表などで表す。

2. 自動に負荷グラフ・容量グラフ・ガントチャート・負荷容量図を作成する。

LCB がユーザから入力したデータから、自動で負荷グラフと容量グラフを作成する。そして、ユーザから入力したデータとソルバの出力を用いて、自動でガントチャートと負荷容量図を作成する。

2.2 LCB の機能要求仕様

設計方針（2.1.1）を基に、負荷容量参照モデルに基づくプロジェクトスケジューリング法における、LCB の必要な機能は以下の通りである。そして、図 2.2 のようなユースケースモデルを設計する。ソルバとして GLPK を用いる。

- LCB が組織のデータベースと WBS のデータベースを持つ。
- GUI で負荷グラフを入力する機能
- 入力した負荷グラフデータを編集機能
- 入力した負荷グラフデータを保存する機能
- 保存した負荷グラフのデータを開く機能
- 負荷グラフを追加する機能
- 負荷グラフを作成機能
- GUI で容量グラフを入力する機能
- 入力した容量グラフデータを保存
- 保存した容量グラフのデータを開く機能
- 容量グラフを作成機能
- GUI で目的関数を入力する機能
- 入力した目的関数編集機能
- 入力した目的関数を保存する機能
- 保存した目的関数のデータを開く機能
- GLPK 入力データの加工機能
- GLPK 出力の加工機能
- ガントチャート作成機能
- 負荷容量図作成機能

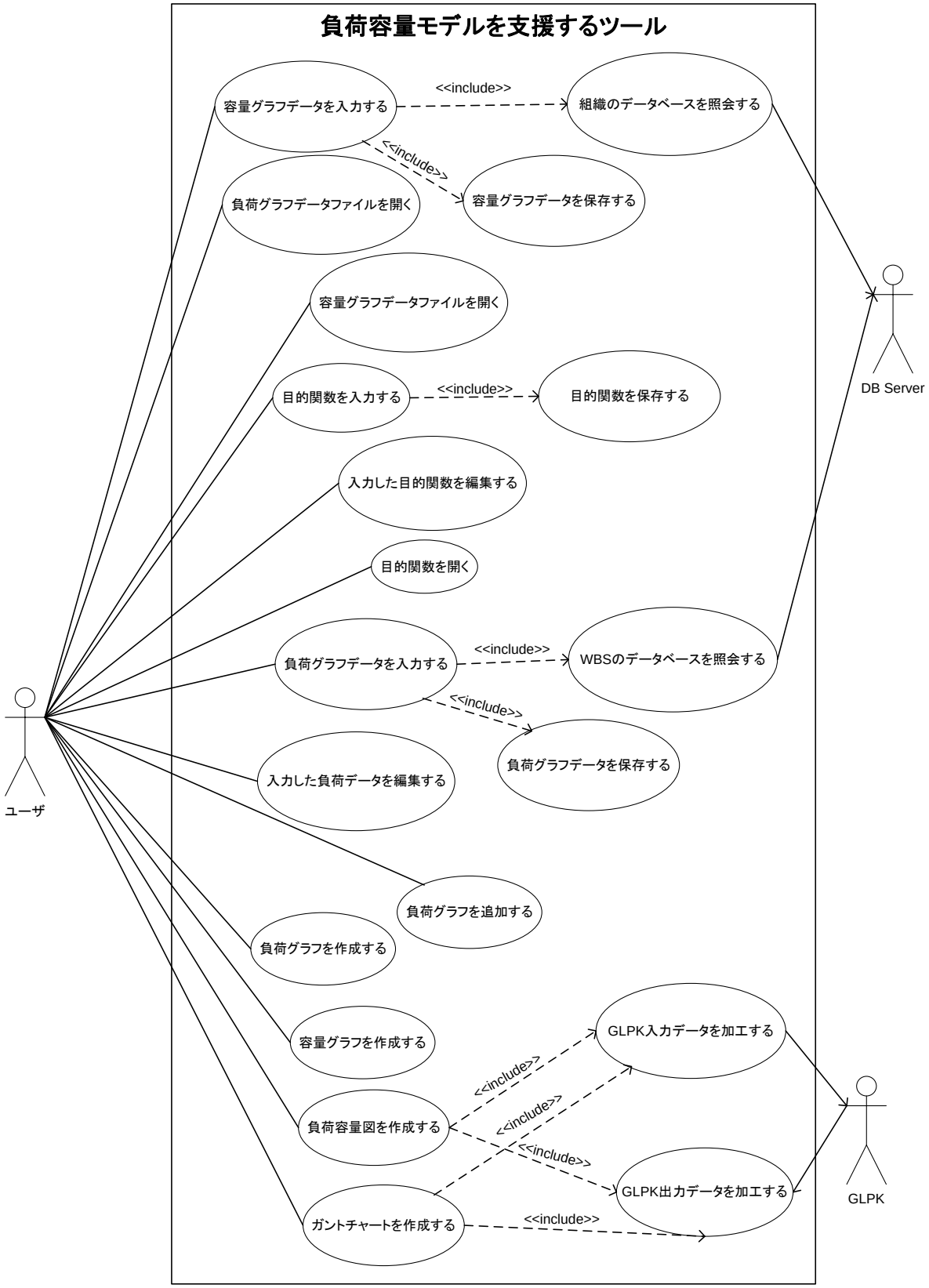


図 2. 1 LCB のユースケースモデル

2.3 LCB の各部の機能仕様定義

LCB の各部の機能仕様を各ユースケースのユースケース記述として述べる。

機能定義：LCB を起動すると、メイン画面が表れる。

2.3.1　負荷グラフデータの入力機能

負荷グラフを作成するためのデータは 4 つあり、WP・先行制約・工数・必要スキルである。従って、GUI で負荷グラフを入力する機能はユーザから入力を貰って、WBS データベースから必要なデータを照会して、4 つの必要なデータアレーを作る機能である。

GUI で負荷グラフを入力する機能は 2 つのユースケース、「負荷グラフデータを入力する」・「WBS のデータベースをクエリする」ユースケースで実現する。設計方針（2.1.1）を基に、負荷グラフの入力機能が GUI、負荷グラフ入力画面で行うことが必要である。

設計方針と LCB の機能要求仕様により、2 つのユースケースの記述を表 2.1 と 2.2 に示す。

ユースケース名	負荷グラフデータを入力する
ユースケース ID	UC01
アクター	ユーザ
概要	このユースケースはユーザによる負荷グラフデータの入力についての説明である。
事前条件	メイン画面が表示されている。
基本フロー	1. ユーザがメイン画面の「負荷データを入力」ボタンを押す。 2. ツールは負荷グラフ入力画面を表示する。 3. ユーザは WBS 型の種類を選択する。（新規 WBS か派生開発 WBS か一つ選択する。） 4. ツールはデータベースサーバーから選択した WBS 型の種類により、全ての WBS 型の名前を照会する。 5. ツールは、照会した WBS 型の名前を WBS 型のドロップダウンコンボボックスに表す。 6. ユーザは WBS 型のドロップダウンコンボボックスから一つの WBS 型を選択する。 7. ツールは選択した WBS 型により、データベースサーバーから全ての WBS インスタンスの名前を照会する。 8. ツールは、照会した WBS インスタンスの名前をドロップダウンコンボボックスで表す。 9. ユーザは WBS インスタンスのドロップダウンコンボボックスから一つの WBS インスタンスを選択する。 10. ユーザは容量入力画面の「入力」ボタンを押す。

	11. ツールは *UC02　WBS のデータをクエリーする*で WP 名・必要スキル・工数・先行制約のデータを照会する。 12. ツールは照会した WP 名・必要スキル・工数・先行制約を表す。 13. ユーザが照会したデータを編集しないままで、「OK」ボタンを押すと、ツールは WP 名・必要スキル・工数・先行制約のアレーを作成する。 14. ツールは、保存を確認するメッセージボックスを表示する。 15. ユーザは「はい」を選択する。 16. ユーザは *UC04 負荷グラフデータを保存する*で入力したデータを保存する。 17. ツールは負荷グラフ入力画面を閉じて、メイン画面を表示する。
代替フロー	A1.　照会したデータを編集する もし 13 の段階で、ユーザが照会したデータを編集して、「OK」ボタンを押すと、ツールは編集したデータを用いて、WP 名・必要スキル・工数・先行制約のアレーを作成し、負荷グラフ入力画面を閉じる。 A2 もし 15 の段階で、ユーザが「いいえ」を選択すると、ツールは入力したデータを保存しないで、負荷グラフ入力画面を閉じ、メイン画面を表示する。
事後条件	入力したデータから、WP 名・必要スキル・工数・先行制約のアレーが作成される、負荷グラフ入力画面が閉じて、メイン画面が表示される。

表 2. 1 負荷グラフデータを入力するユースケース記述

ユースケース名	WBS のデータを照会する
ユースケース ID	UC02
アクター	ユーザ
概要	このユースケースはツールによる WP 名・必要スキル・工数・先行制約の照会についての説明である。
事前条件	ユーザが組織の WBS インスタンスを選択して、負荷グラフ入力画面の「入力」ボタンを押した。
基本フロー	1. ツールは、選択した WBS インスタンスの全ての WP 名を照会する。 2. 各 WP 名により、その WP の必要スキル・工数のデータ・先行制約を照会する。
代替フロー	なし
事後条件	ツールは WP 名・必要スキル・工数のデータ・先行制約のデータを照会した。

表 2. 2 WBS のデータを照会するユースケース記述

2.3.2　負荷グラフデータの編集機能

入力した負荷グラフデータの編集は「入力した負荷データを編集する」ユースケースで実現する。この機能は負荷グラフ入力画面で行う。設計方針と LCB の機能要求仕様により、「入力した負荷データを編集する」のユースケース記述を表 2.3 に示す。

ユースケース名	入力した負荷データを編集する
ユースケース ID	UC03
アクター	ユーザ
概要	このユースケースは入力した負荷データを編集する機能について説明する。
事前条件	• メイン画面が表示されている。 • ユーザが負荷グラフのデータを入力した。
基本フロー	1. ユーザはメイン画面の負荷グラフの「Edit」ボタンを押す。 2. ツールは、入力した負荷グラフ入力画面を表示する。 3. ユーザは入力したデータを編集する。 4. ユーザは「OK」ボタンを押す。 5. ツールは編集したデータにより、WP 名・必要スキル・工数・先行制約のアレーを作成する。 6. ツールは、負荷グラフ入力画面を閉じ、メイン画面を表示する。
事後条件	データが編集された。負荷グラフ入力画面を閉じて、メイン画面が表示される。

表 2.3 入力した負荷データを編集するユースケース記述

2.3.3　負荷グラフデータの保存機能

入力した負荷グラフデータを保存する機能は「負荷グラフデータを保存する」ユースケースで実現する。設計方針（2.1.1）を基に、この機能を実現するために、負荷グラフ入力画面には「ファイル」メニューバーが必要となる。さらに、メニューバーでは「保存」タブが必要となる。保存するファイルは以下のような構造を持つ。

行番号	保存する内容
1	Load Graph
2	WBS 型種類
3	WBS 型名
4	WBS インスタンス名
5	WP のアレー
6	必要スキルのアレー
7	工数のアレー
8	先行制約のアレー

表 2.4 負荷データを保存するファイルの型式

設計方針と LCB の機能要求仕様により、「負荷グラフデータを保存する」のユースケース記述を表 2.5 に示す。

ユースケース名	負荷グラフデータを保存する
ユースケース ID	UC04
アクター	ユーザ
概要	このユースケースはユーザの入力した負荷データを保存する機能についての説明である。
事前条件	• 負荷グラフ入力画面が表示されている。 • ツールが WP 名・必要スキル・工数のデータ・先行制約のデータを照会した。
基本フロー	1. ユーザは「ファイル」メニューバーの「保存」タブを押す。 2. ツールは保存画面を表示する。 3. ユーザは保存するファイル名を入力する。 4. ユーザは保存画面の「保存」ボタンを押す。 5. ツールは表 2.4 のような型式のファイルを作成し、保存画面を閉じる。
代替フロー	なし
事後条件	表 2.4 のような型式のファイルが作成された。

表 2.5 負荷グラフデータを保存するユースケース記述

2.3.4 負荷グラフの保存データを開く機能

保存した負荷グラフのデータを開く機能は「負荷グラフデータファイルを開く」ユースケースで実現する。設計方針（2.1.1）を基に、この機能を実現ために、負荷グラフ入力画面の「ファイル」メニューバーの「開く」タブが必要である。開く結果は、ツールがユーザが選択したデータファイルのデータを画面に表示する。設計方針と LCB の機能要求仕様により、「負荷グラフデータファイルを開く」のユースケース記述を表 2.6 に示す。

ユースケース名	負荷グラフデータを開くする
ユースケース ID	UC05
アクター	ユーザ
概要	このユースケースはユーザの入力した負荷データを開くする機能についての説明である。
事前条件	• 負荷グラフ入力画面が表示されている。 • 負荷データファイル、表 2.4 のような型式のファイルがある。
基本フロー	1. ユーザは「ファイル」メニューバーの「開く」タブを押す。 2. ツールは開く画面が表す。 3. ユーザは負荷データファイルを選択する。 4. ツールはツールは、ユーザが選択したデータファイルのデータを

	画面に表示する。
事後条件	負荷データファイルから作成した負荷グラフ入力画面が表示される。

表 2.6 負荷グラフデータファイルを開くユースケース記述

2.3.5 負荷グラフの追加機能

ユーザが一つ以上の負荷グラフを入力することも可能となる。この機能は「負荷グラフを追加する」ユースケースで実現する。この機能を行うために、メイン画面の「負荷グラフを追加」ボタンが必要である。「負荷グラフを追加」ボタンを押すと、新たな負荷グラフの入力・表示場所がメイン画面に表れる。

設計方針と LCB の機能要求仕様により、「負荷グラフを追加する」のユースケース記述を表 2.7 に示す。

ユースケース名	負荷グラフを追加する
ユースケース ID	UC06
アクター	ユーザ
概要	このユースケースは負荷グラフを追加する処理についての説明である。
事前条件	• メイン画面が表示されている。
基本フロー	1. ユーザは「負荷グラフを追加」ボタンを押す。 2. ツールはメイン画面に新たな負荷グラフの入力・表示場所を表示する。
代替フロー	-
事後条件	メイン画面では新たな負荷グラフの入力場所が表示される。

表 2.7 負荷グラフを追加するユースケース記述

2.3.6 負荷グラフの作成機能

負荷グラフの作成機能は「負荷グラフを作成する」ユースケースで実現する。設計方針（2.1.1）を基に、この機能は自動的に図 1.2 のような負荷グラフを作成する。そして、この機能を実現するために、出力画面、負荷グラフ画面が必要となる。

設計方針と LCB の機能要求仕様により、「負荷グラフを作成する」のユースケース記述を表 2.8 に示す。

ユースケース名	負荷グラフを作成する
ユースケース ID	UC07
アクター	ユーザ
概要	このユースケースは負荷グラフを作成して、表示する処理についての説明である。
事前条件	• メイン画面が表示されている。 • 負荷グラフデータが入力されている。
基本フロー	1. ユーザはメイン画面の「負荷グラフを表示」ボタンを押す。 2. ツールは入力された負荷グラフのデータから、負荷グラフを作成する。 3. ツールは負荷グラフ画面に、作成した負荷グラフを表示する。
代替フロー	-
事後条件	負荷グラフが負荷グラフ画面で表示される。

表 2. 8 負荷グラフを作成するユースケース記述

2.3.7　容量グラフの入力機能

容量のデータ、リソース・組織の依存関係（Supervisor）・保持スキル、は組織のデータベースから照会できる。照会したデータから、容量グラフのデータアレーを作成する。

GUI で容量グラフの入力機能は 2 つのユースケース、「容量グラフデータを入力する」と「組織のデータベースをクエリする」ユースケースで実現する。これらつのユースケースの記述が表 2.1 と 2.2 に示す。容量グラフの入力機能には、容量入力画面が必要となる。しかし、もしリソース数が多い場合、ユーザの入力負担が増す。従って、機能チームデータを保持するデータベースが必要となる。照会方針は照会方針が表 2.9 で述べる。

ステップ	行動
1.	チームリーダのデータを照会する。
2.	サブリーダーの人数により、新しい開発期間中で負荷が少ないサブリーダーを照会する。
3.	照会したサブリーダーの全ての部下のデータを照会する。

表 2. 9 機能チームデータを照会する照会方針

ユースケース名	容量グラフデータを入力する
ユースケース ID	UC08
アクター	ユーザ
概要	このユースケースはユーザによる容量グラフデータの入力について

	の説明である。
事前条件	メイン画面が表示されている。
基本フロー	1. ユーザはメイン画面の「容量データを入力」ボタンを押す。 2. ツールは、容量グラフ入力画面を表示する 3. ツールがデータベースサーバーから全ての組織の名前を照会する。 4. ツールは、照会した組織の名前を組織のドロップダウンコンボボックスに表示する。 5. ユーザは組織のドロップダウンコンボボックスからひとつの組織を選択する。 6. ツールはデータベースサーバーから全ての組織の機能チームの種類を照会する。 7. ツールは、照会したチームの種類をチームの種類のドロップダウンコンボボックスに表示する。 8. ユーザはチームの種類のドロップダウンコンボボックスからひとつの組織を選択する。 9. ツールはデータベースサーバーから、選択した機能チームの種類により、機能チームの名前を照会する。 10. ツールは、照会した機能チームの名前を機能チームのドロップダウンコンボボックスで表示する。 11. ユーザは機能チームのドロップダウンコンボボックスから一つの機能チームを選択する。 12. ユーザはサブリーダーの人数を入力する。 13. ユーザは容量グラフ入力画面の「OK」ボタンを押す。 14. ツールは *UC09 組織のデータをクエリーする*で機能チームのデータを照会する。 15. ツールはが照会したデータから名前・Supervisor・保持スキルのアレーを作成する。 16. ツールは保存を確認するメッセージボックスを表示する 17. ユーザは「はい」を選択する。 18. ユーザは *UC10 負荷グラフデータを保存する*入力したデータを保存する。 19. 容量グラフ入力画面が閉じ、メイン画面を表示する。。
代替フロー	A1 もし 17 の段階で、ユーザが「いいえ」を選択すると、ユースは、入力したデータを保存しないままで、容量グラフ入力画面が閉じ、メイン画面を表示する。
事後条件	名前・Supervisor・保持スキルのアレーが作成される、容量グラフ入力画面が閉じて、メイン画面表示される。

表 2. 10 容量グラフデータを入力するユースケース記述

ユースケース名	組織のデータを照会する
ユースケース ID	UC09

アクター	ユーザ
概要	このユースケースは機能チームデータ照会についての説明である。
事前条件	ユーザが組織の名前・機能チームの種類・機能チームを選択して、容量グラフ入力画面の「OK」ボタンを押した。
基本フロー	1. ツールはユーザが選択した機能チームに所属する社員の名前と位置のデータを照会する。 2. ツールは、照会した社員のデータにより、Supervisor・保持スキルのデータを照会する。
代替フロー	-
事後条件	ツールがリソースの名前・Supervisor・保持スキルのデータを照会した。

表 2. 11 組織のデータを照会するユースケース記述

2.3.8　容量グラフデータの保存機能

入力した容量グラフデータを保存する機能は「容量グラフデータを保存する」ユースケースで実現する。このユースケース記述を表 2.13 に示す。設計方針（2.1.1）を基に、この機能を実現するために、容量グラフ入力画面の「ファイル」メニューバーが必要となる。さらに、メニューバーでは「保存」タブが必要となる。保存する容量データファイルは以下のような構造である。

Line Number	保存する内容
1	Capacity Graph
2	組織名
3	機能チームの種類
4	機能チーム名
5	サブリーダの人数
6	リソースのアラー
7	位置のアラー
8	Supervisor のアレー
9	保持スキルの 2D アレー

表 2. 12 容量データを保存するファイルの型式

ユースケース名	負荷グラフデータを保存する
ユースケース ID	UC10
アクター	ユーザ
概要	このユースケースはユーザの入力した容量データを保存する機能についての説明である。

事前条件	• 容量グラフ入力画面が表示されている。 • ツールはリソースの名前・位置・Supervisor・保持スキルのデータを照会した。
基本フロー	1. ユーザは「ファイル」メニューバーの「保存」タブを押す。 2. ツールは、保存画面を表示する。 3. ユーザは保存するファイル名を入力する。 4. ユーザは保存画面の「保存」ボタンを押す。 5. ツールは表 2.12 のような型式のファイルを作成する。
代替フロー	-
事後条件	ツールが表 2.12 のような型式のファイルを作成した。

表 2. 13 負荷グラフデータを保存するユースケース記述

2.3.9 容量グラフの保存データを開く機能

保存した容量グラフのデータを開く機能は「容量グラフデータファイルを開く」ユースケースで実現する。設計方針（2.1.1）を基に、この機能を実現するために、容量グラフ入力画面の「ファイル」メニューバーの「開く」タブが必要となる。開く結果は、ツールが、ユーザが選択したデータファイルのデータを画面に表示する。設計方針と LCB の機能要求仕様により、「容量グラフデータファイルを開く」のユースケース記述を表 2.14 に示す。

ユースケース名	容量グラフデータを開くする
ユースケース ID	UC11
アクター	ユーザ
概要	このユースケースは保存した容量グラフのデータを開く機能についての説明である。
事前条件	• 容量グラフ入力画面が表示される。 • 容量データファイル、表 2.12 のような型式のファイルがある。
基本フロー	1. ユーザがはファイル」メニューバーの「開く」タブを押す。 2. ツールは開く画面を表示する、 3. ユーザは容量データファイルを選択する。 4. ツールは、ユーザが選択したデータファイルのデータを画面に表示する。
代替フロー	-
事後条件	容量データファイルから作成した容量グラフ入力画面が表示される。

表 2. 14「容量グラフデータファイルを開く」ユースケース記述

2.3.10 容量グラフの作成機能

容量グラフの作成機能は「容量グラフを作成する」ユースケースで実現する。設計方針（2.1.1）を基に、この機能は自動的に図 1.3 に示す負荷グラフを作成する。この機能を実現するためには、出力画面、容量グラフ画面が必要となる。

表 2.15 のようなリソースの保持スキル情報を表示する必要があるので、容量グラフ画面は二つの部分に分ける。ひとつは容量グラフを表示する部分であり、もうひとつはリソースの保持スキル情報を表示する部分である。

リソース名：	位置	保持スキル 1	保持スキル 2	保持スキル 3
リソース A	リーダ	○	○	○
リソース B	サブリーダ	○	○	

表 2. 15 リソースの保持スキル情報表の例

設計方針と LCB の機能要求仕様により、「容量グラフを作成する」のユースケース記述を表 2.16 に示す。

ユースケース名	容量グラフを作成する
ユースケース ID	UC12
アクター	ユーザ
概要	このユースケースは容量グラフを作成して、表示する処理についての説明である。
事前条件	• メイン画面が表している。 • 容量グラフデータを入力した。
基本フロー	1. ユーザがメイン画面の「容量グラフを表示」ボタンを押す。 2. ツールが入力した容量グラフのデータから、容量グラフとリソースの保持スキル表を作成する。 3. ツールは、容量グラフ画面が作成した負荷グラフとリソースの保持スキル表を表示する。
代替フロー	-
事後条件	容量グラフのデータから作成した容量グラフとリソースの保持スキル表が負荷グラフ画面に表示された。

表 2. 16「負荷グラフを作成する」のユースケース記述

2.3.11 目的関数の入力機能

2.1 の設計方針により、目的関数はリソースを割り当てる方針により定式化される。しかし、リソース割り当てる方針色々考えられる。本ツールは二種類のリソースを割り当てる項目を提供する。

- ひとつはリソースに負荷を割り当てるかなるべく割り当てないかという項目である。
- もうひとつはあるリソースが他リソース（例えば、キーパーソン）と一緒に働くかどうかという項目である。つまり、リソースマッチング目標である。

GUI で目的関数を入力する機能は「目的関数を入力する」ユースケースで実現する。設計方針を基に、負荷グラフの入力機能画面に、目的関数入力画面が必要となる。目的関数入力画面は 2 つの部分に分ける。ひとつはリソース割り当て表（表 2.17）を表示する。もう一つはリソースマッチングの部分である。リソースマッチングの部分で、ユーザは自由にリソースを追加できる。

リソース名：	位置	割り当て
リソース A	MTL	☑
リソース B	MSTL	☑
リソース C	Member	☐

表 2.17 リソース割り当て表の例

「目的関数を入力する」のユースケース記述を表 2.18 に述べる。

ユースケース名	目的関数を入力する
ユースケース ID	UC13
アクター	ユーザ
概要	このユースケースがユーザの目的関数入力処理についての説明である。
事前条件	• メイン画面が表示される。 • 容量グラフデータを入力した。
基本フロー	1. ユーザはメイン画面の「目的関数を入力」ボタンを押す。 2. ツールは入力した容量グラフのデータから、リソース割り当て表とリソースマッチング部分を作成する。 3. ツールは、目的関数画面を表示する。 4. ユーザは、リソース割り当て表とリソースマッチングを入力す

	る。 5. ユーザは目的関数画面の「OK」ボタンを押す。 6. ツールは入力したデータから、リソース割り当てアレーとリソースマッチングを作成する。 7. ツールは、目的関数入力画面を閉じ、メイン画面を表示する。
代替フロー	-
事後条件	リソース割り当てアレーとリソースマッチングを作成される、目的関数入力画面が閉じ、メイン画面が表示される。

表 2. 18「目的関数を入力する」のユースケース記述

2.3.12 目的関数編集機能

目的関数編集機能は「入力した目的関数を編集する」ユースケースで実現する。この機能は目的関数入力画面で実現する。「入力した目的関数を編集する」のユースケース記述を表 2.3 に示す。

ユースケース名	入力した目的関数を編集する
ユースケース ID	UC14
アクター	ユーザ
概要	このユースケースが入力した目的関数を編集する機能についての説明である。
事前条件	• メイン画面が表示されている。 • ユーザが目的関数を入力した。
基本フロー	1. ユーザはメイン画面の目的関数の「Edit」ボタンを押す。 2. ツールは、入力した目的関数入力画面を表示する。 3. ユーザは入力したデータを編集する。 4. ユーザは「OK」ボタンを押す。 8. ツールは編集したデータにより、リソース割り当てのアレーとリソースマッチングを作成する。 5. ツールは、目的関数入力画面を閉じ、メイン画面を表示する。
代替フロー	-
事後条件	目的関数データを編集された、目的関数入力画面が閉じ、メイン画面が表示される。

表 2. 19 入力した目的関数を編集するユースケース記述

2.3.13 目的関数の保存機能

入力した目的関数を保存する機能は「目的関数を保存する」ユースケースで実現する。この機能を実現するためには、目的関数入力画面に「ファイル」メニューバーが必要となる。メニューバーには「保存」タブが必要となる。保存するファイルは以下のような構造である。

行番号	保存内容
1	Objective Function
2	リソース割り当てのアレー
3	リソースマッチングの 2D アレー

表 2. 20 目的関数を保存するファイルの型式

「目的関数を保存する」のユースケース記述を表 2.21 に示す。

ユースケース名	目的関数を保存する
ユースケース ID	UC16
アクター	ユーザ
概要	このユースケースはユーザの入力した目的関数を保存する機能についての説明である。
事前条件	• 目的関数入力画面が表示されている。 • 目的関数を入力した。
基本フロー	1. ユーザがは「ファイル」メニューバーの「保存」タブを押す。 2. ツールは保存画面を表示する。 3. ユーザが保存するファイル名を入力する。 4. ユーザは保存画面の「保存」ボタンを押す。 5. ツールは表 2.20 のような型式のファイルを作成する。
代替フロー	-
事後条件	ツールが表 2.20 のような型式のファイルを作成した。

表 2. 21 負荷グラフデータを保存するユースケース記述

2.3.14 保存した目的関数のデータを開く機能

保存した目的関数のデータを開く機能は「目的関数を開く」ユースケースで実現する。この機能を実現するために、目的関数入力画面の「ファイル」メニューバーの「開く」タブが必要である。開く結果は、目的関数データファールの内容から作成する目的関数入力画面となる。設計方針と LCB の機能要求仕様により、「目的関数を開く」のユースケース記述を表 2.22 に示す。

ユースケース名	目的関数を開く
ユースケース ID	UC17
アクター	ユーザ
概要	このユースケースは保存した目的関数のデータを開く機能についての説明である。

事前条件	• 目的関数の入力画面が表示される。 • 目的関数のデータファイル、表 2.20 のような型式のファイルがある。
基本フロー	1. ユーザは「ファイル」メニューバーの「開く」タブを押す。 2. ツールは開く画面が表示する。 3. ユーザは目的関数ファイルを選択する。 4. ツールは目的関数ファイルのデータから新たな目的関数入力画面を作成して、表す。
代替フロー	-
事後条件	目的関数データファイルのデータを表示した目的関数入力画面が表示される。

表 2.22 目的関数を開くユースケース記述

2.3.15 GLPK 入力データの加工機能

GLPK を利用するために、GLPK 入力データの加工機能が必要である。負荷容量参照モデルに基づくプロジェクトスケジューリング法により、GLPK の入力は三つあり、決定変数・制約式・目的関数である。GLPK の入力の加工機能はその入力を準備して、GLPK を利用する機能である。

- 決定変数は負荷容量参照モデルに基づくプロジェクトスケジューリング法により、二つがあり、時間割り当てのための決定変数（$x_{il} \in \{0,1\}$）とリソース割り当てのための決定変数（$b_{ir} \in \{0,1\}$）である。これは LCB が準備する。
- 制約式は入力した負荷グラフ・容量グラフのデータから作成する。
- 目的関数は目的関数入力画面の入力したデータから作成する。

「GLPK 入力データを加工する」ユースケース記述を表 2.23 に示す。

ユースケース名	GLPK 入力データを加工する
ユースケース ID	UC18
アクター	ユーザ
概要	このユースケースは GLPK 入力データを加工する処理についての説明である。
事前条件	負荷グラフ・容量グラフ・目的関数のデータが入力された。
基本フロー	1. ツールは入力した負荷グラフ・容量グラフのデータから制約式作成する。 2. ツールは、入力した目的関数のデータから目的関数を作成する。 3. 準備した決定変数と作成した制約式・目的関数を GLPK に入力する。
代替フロー	-

事後条件	決定変数・制約式・目的関数を GLPK に入力した。

表 2. 23 GLPK 入力データを加工するユースケース記述

2.3.16 GLPK 出力の加工機能

GLPK 出力の加工機能は「GLPK 出力を加工する」ユースケースで実現する。負荷容量参照モデルに基づくプロジェクトスケジューリング法により、GLPK の出力（.sol ファイル）ha

二つの情報があり、WP の開始時刻とリソース割り当て結果である。だから、GLPK 出力の加工機能は GLPK の出力から WP の開始時刻とリソース割り当てのアレーを作成する機能である。

設計方針と LCB の機能要求仕様により、「GLPK 出力を加工する」ユースケース記述を表 2.23 に示す。

ユースケース名	GLPK 出力データを加工する
ユースケース ID	UC19
アクター	ユーザ
概要	このユースケースは GLPK 出力を加工する処理についての説明である。
事前条件	• GLPK の出力ファイルがある。
基本フロー	1. ツールが GLPK の出力ファイルを読む。 2. ツールは、読んだファイルから作業開始時刻のアレーとリソース割り当てのアレーを作成する。
代替フロー	-
事後条件	作業開始時刻のアレーとリソース割り当てのアレーが作成される。

表 2. 24 GLPK 出力を加工するユースケース記述

2.3.17 ガントチャート作成機能

定義：「ガントチャート」とはプロジェクトにおけるスケジュール管理に利用される表のことを指す。線表とも呼ばれる。各 WP の実施スケジュールを横棒グラフ形式で記述する。LCB が作成するガントチャートの形は図 2.2 のようになる。

- 縦軸：WP 名と割り当てるリソース名を記載する。
- 横軸：日程を記載する。

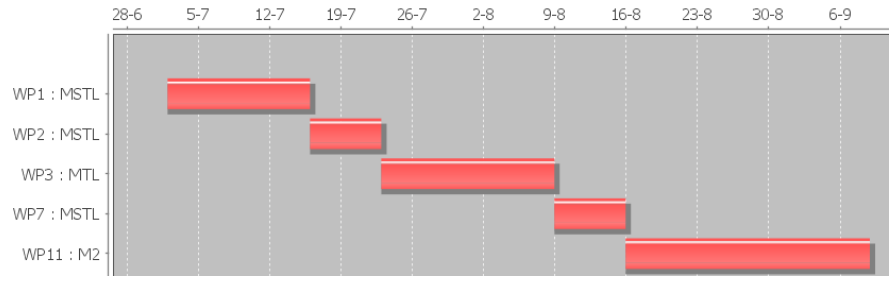


図 2. 2 ガントチャート

ガントチャート作成する機能はが GLPK を利用して、入力した負荷グラフ・容量グラフのデータと GLPK の出力からガントチャートを作成して、表示する機能。この機能は「ガントチャートを作成する」ユースケースで実現する。設計方針と LCB の機能要求仕様により、「ガントチャートを作成する」ユースケース記述を表 2.25 に示す。

ユースケース名	ガントチャートを作成する
ユースケース ID	UC20
アクター	ユーザ
概要	このユースケースはガントチャートを作成する処理についての説明である。
事前条件	• 負荷グラフ・容量グラフ・目的関数のデータを入力した。
基本フロー	1. ツールは *UC18 GLPK 入力データを加工する*で GLPK を利用する。 2. ツールは *UC19 GLPK 出力を加工する*で GLPK 出力から作業開始時刻のアレーとリソース割り当てのアレーを作成する。 3. ツールは、負荷グラフ・容量グラフのデータと作業開始時刻のアレーとリソース割り当てのアレーからガントチャートを作成する。 4. ツールは、ガントチャート画面に作成したガントチャートを表示する。
代替フロー	-
事後条件	ガントチャート画面に作成したガントチャートが表示される。

表 2. 25 ガントチャートを作成するユースケース記述

2. 3. 18 負荷容量図作成機能

負荷容量図作成する機能はが GLPK を利用して、入力した負荷グラフ・容量グラフのデータと GLPK の出力から負荷容量図を作成して、表示する機能。この機能は「負荷容量図を作成する」ユースケースで実現する。設計方針と LCB の機能要求仕様により、「負荷容量図を作成する」ユースケース記述を表 2.24 に示す。

ユースケース名	負荷容量図を作成する
ユースケース ID	UC21
アクター	ユーザ
概要	このユースケースが負荷容量図を作成する処理についての説明である。
事前条件	• 負荷グラフ・容量グラフ・目的関数のデータを入力した。
基本フロー	1. ツールが *UC18 GLPK 入力データを加工する*で GLPK を利用する。 2. ツールが *UC19 GLPK 出力を加工する*で GLPK 出力から作業開始時刻のアレーとリソース割り当てのアレーを作成する。 3. ツールは、負荷グラフと容量グラフのデータと作業開始時刻のアレーとリソース割り当てのアレーから負荷容量図を作成する。 4. ツールは、負荷容量図画面に作成した負荷容量図を表示する。
代替フロー	-
事後条件	負荷容量図画面に作成した負荷容量図を表示される。

表 2. 26 負荷容量図を作成するユースケース記述

2.4 ツールの構成

設計方針（2.1）と機能要求仕様（2.2）から、LCB の構成を図 2.3 に示すように設計する。構造を以下に述べる。

- ユーザ PC とデータベースは Ethernet で接続されている。
- ユーザ PC では本ツールと GLPK を実行する。
- 本ツールにおけるユーザからの入力は三つのデータセットが成り、負荷グラフデータ・容量グラフデータ・目的関数データである。
- 「LoadGraphInput」コンポーネントは負荷グラフデータの入力・編集・保存・開く・追加を取り扱う。
- 「CapacityGraphInput」コンポーネントは容量グラフデータの入力・保存・開くを取り扱う。
- 「ObjectiveFunctionInput」コンポーネントは目的関数データの入力・編集・保存・開くを取り扱う。
- 「GLPKInput」コンポーネントは三つの入力データセットを加工して、GLPK に入力する。
- GLPK の出力は「.sol」ファイルである。

- 「GLPKOutputDerive」コンポーネントは GLPK の出力（.sol）を加工して、必要なアレーデータを作成する。
- 本ツールの出力は 4 つあり、負荷グラフ・容量グラフ・負荷容量図・ガントチャートである。
- 「CreateLoadGraph」コンポーネントは負荷グラフを作成する機能を取り扱う。
- 「CreateCapacityGraph」コンポーネントは負荷グラフを作成する機能を取り扱う。
- 「CreateLoadCapacityGraph」コンポーネントは負荷容量図を作成する機能を取り扱う。
- 「CreateGanttChart」コンポーネントはガントチャートを作成する機能を取り扱う。

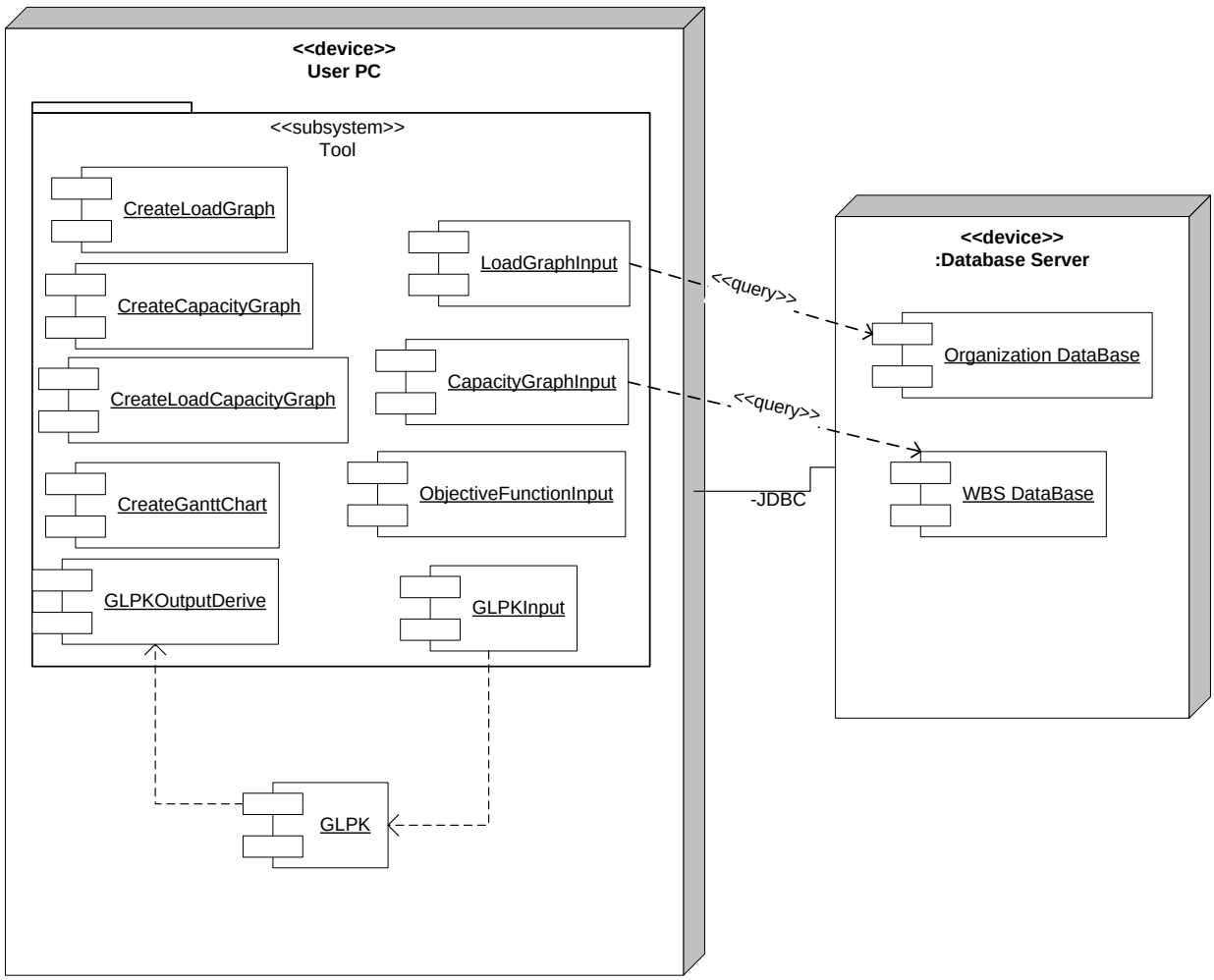


図 2. 3 ツールの構成

2.5 入力画面設計

ツールの画面について説明する。ツールには以下に示す 8 つの主な画面がある。

- メイン画面
- 負荷グラフのデータ入力画面
- 容量グラフ入力画面
- 目的関数入力画面
- 負荷グラフ画面
- 容量グラフ画面
- 負荷容量図画面
- ガントチャート画面

2.5.1 メイン画面

ツールを起動するとメイン画面が表示される。このメイン画面は 10 個のユースケースの始点である。ユースケースを起動するために、一つずつのボタンが必要である。以下の通りである。

- 容量グラフデータを入力する：容量グラフの「新規入力」ボタン
- 容量グラフを作成する：容量グラフの「表示」ボタン
- 負荷グラフデータを入力する：負荷グラフの「新規入力」ボタン
- 入力した負荷データを編集する：負荷グラフの「更新」ボタン
- 負荷グラフを追加する：負荷グラフの「負荷グラフを追加」ボタン
- 負荷グラフを作成する：負荷グラフの「表示」ボタン
- 目的関数を入力する：目的関数の「新規入力」ボタン
- 入力した目的関数を編集する：目的関数の「更新」ボタン
- ガントチャートを作成する：「ガントチャートを作成」ボタン
- 負荷容量図を作成する：「負荷容量図を作成」ボタン

さらに、負荷グラフを追加すると、負荷グラフの「新規入力」・「更新」・「表示」・「ガントチャートを作成」ボタンが追加される。設計したメイン画面は図 2.4 のようになる。

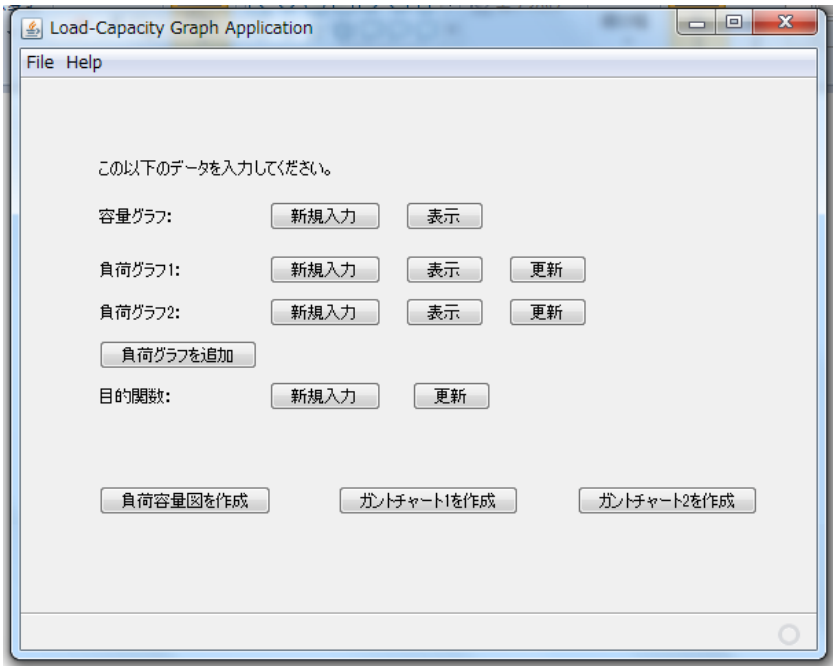


図 2.4 メイン画面

2.5.2 容量グラフ入力画面

容量グラフのデータ入力方法は組織のデータベースから、ユーザが入力した機能チームのデータを照会して、入力する。そのためには、組織の名前・機能チームの種類・機能チームの名前が必要となる。従って、この容量グラフ入力画面には三つのドロップダウンコンボボックスがひとつずつが必要となる。

機能チームの照会方針（表 2.9）により、ユーザはサブリーダの人数を設定できる。従って、この画面にはひとつのテキストボックスが必要となる。さらに、保存と開く機能を実現するために、「ファイル」メニューバーが必要となる。

入力するを完了するために、ひとつの「OK」ボタンが必要となる。容量グラフ入力画面の設計結果は図 2.5 のようになる。

図 2.5 容量グラフ入力画面

2.5.3 負荷グラフ入力画面

5 つのユースケース、「負荷グラフデータを入力する」・「入力した負荷データを編集する」・「負荷グラフデータを保存する」・「負荷グラフデータファイルを開く」が負荷グラフ入力画面で実現する。それぞれのユースケースを実現するために、以下のような GUI が必要となる。

- 「負荷グラフデータを入力する」と「入力した負荷データを編集する」
  - WBS 型の種類を選択する
    新規 WBS か派生開発 WBS を選択するために、二つのラジオボタンと一つの「OK」ボタンが必要となる。
  - WBS 型を選択する
    WBS 型を選択するために、一つの照会した WBS 型名を入れるドロップダウンコンボボックスが必要となる。
  - WBS インスタンスを選択する
    WBS インスタンスを選択するために、一つの照会した WBS インスタンスを入れるドロップダウンコンボボックスが必要となる。
  - WBS のデータを照会して、グラフのアレーデータを作成する。
    WP・先行制約・工数・必要スキルを照会して、それぞれのアレーを作成するために、「入力」ボタンが必要となる。
  - 照会したデータを表示する・負荷データを編集する

WP・先行制約・工数・必要スキルを表示する部分が必要となる。また、データを編集するために、以下の GUI が必要である。

- WP：テキストボックス
- 先行制約の始点：ラベル
- 先行制約の終点：ドロップダウンコンボボックス
- 工数：テキストボックス
- 必要スキル：ドロップダウンコンボボックス

- 「負荷グラフデータを保存する」・「負荷グラフデータファイルを開く」

  保存と開く機能を起動するために、「ファイル」メニューバーが必要となる。

以上の考え方で、負荷グラフのデータ入力画面は図 2.6 のように設計される。

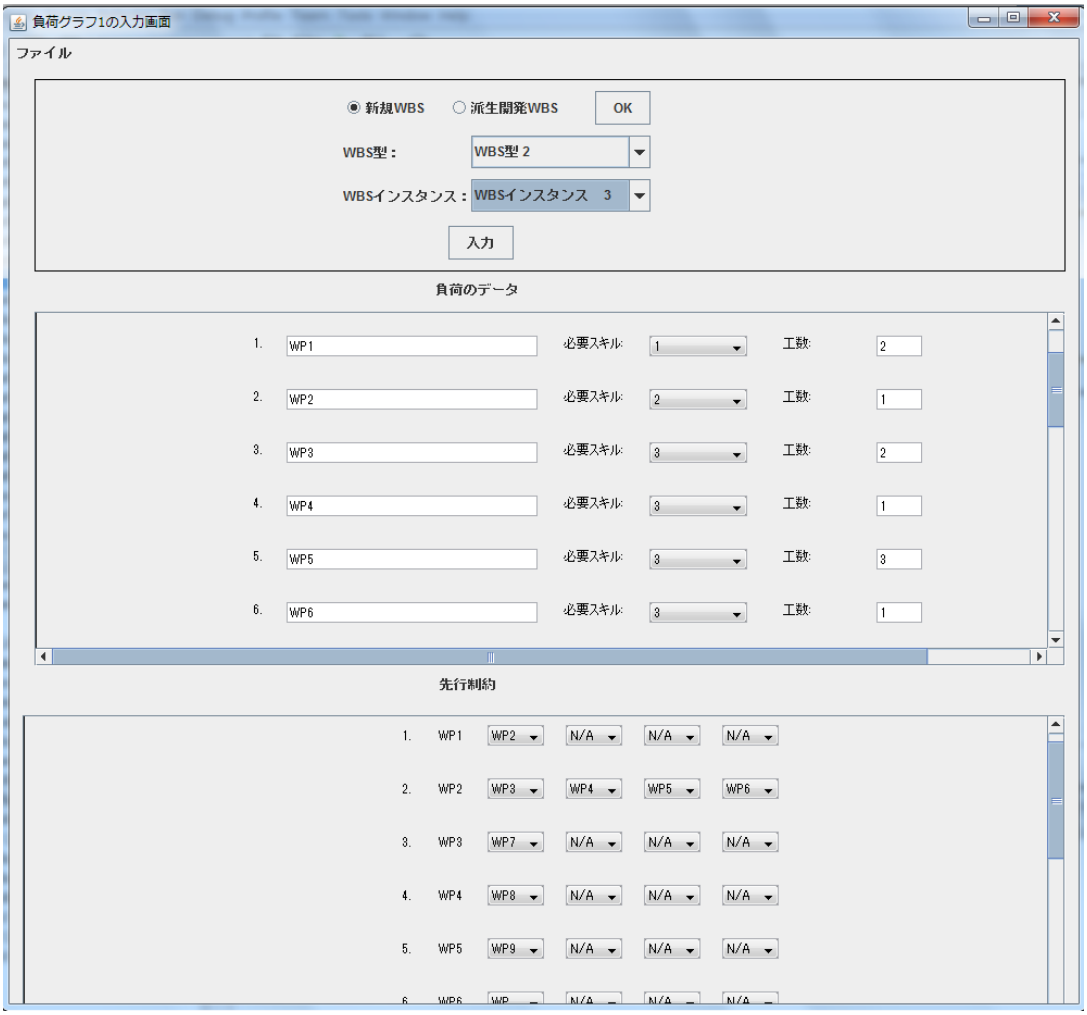


図 2. 6 負荷グラフのデータ入力画面

2.5.4 目的関数入力画面

目的関数入力機能の定義（2.3.11）により、目的関数入力画面は入力した容量グラフのデータから作成する。目的関数入力画面は容量グラフのデータを入力した後、操作可能となる。そして、LCB が提供する目的関数は二つあり、リソースに負荷を割り当てる目標とリソースマッチング目標である。従って、目的関数入力画面を二つの部分に分ける。

目的関数入力機能の定義により、リソースに負荷を割り当てる部分はチェックボックスで入力する。また、リソースマッチングが二つのリソース名を入れたドロップダウンコンボボックスで入力する。リソースマッチングの部分を追加できるようにするため、「追加」ボタンを用意した。

以上の考え方で目的関数入力画面の設計結果を図 2.7 に示す。

図 2.7 目的関数入力画面

2.5.5 保存・開く GUI

ファイルを保存・開くするために、Open/Save 画面（図 2.8）を使う。さらに、保存を確認するために、メッセージボックス（図 2.8）を用意する。

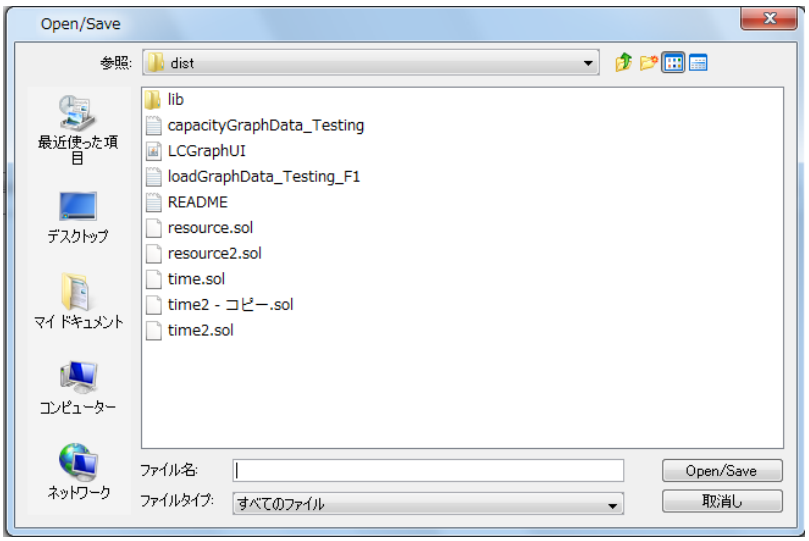


図 2. 8 Open/Save 画面

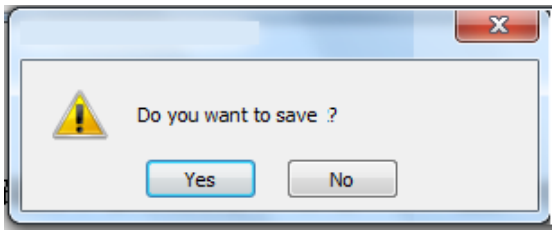


図 2. 9 保存を確認するメッセージボックス

2.6　出力設計

2.6.1 負荷グラフ画面

負荷グラフの作成機能（2.3.6）により、負荷グラフ画面は入力したデータから作成した負荷グラフを表示する。さらに、各頂点の下には属性（必要スキル・工数）を表示する。しかし、もし多数の頂点があったら、属性を読むことが困難になる。そこで、LCB は表の形で負荷グラフと属性を表示する。以上により、負荷グラフ画面は図 2.10 のように設計した。

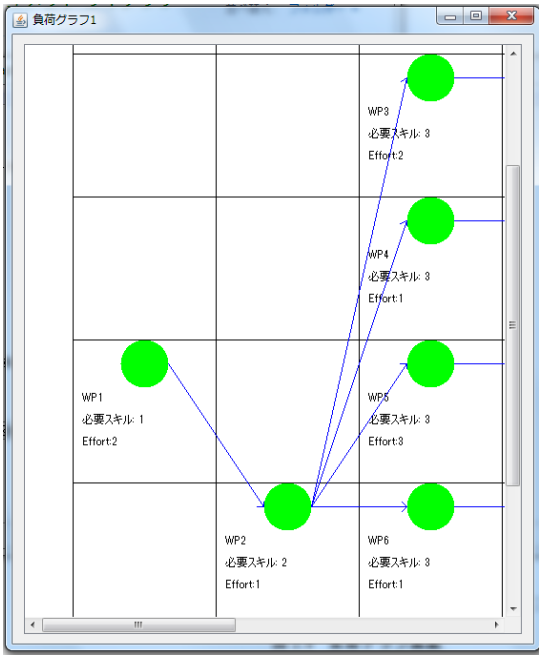


図 2. 10 負荷グラフ画面

2. 6. 2 容量グラフ画面

容量グラフの作成機能（2.3.6）により、容量グラフ画面は入力したデータから作成した容量グラフを表示する。さらに、保持スキル表（表 2. 15）と容量グラフを共に表示する。以上により、容量グラフ画面は図 2.11 のように設計した。

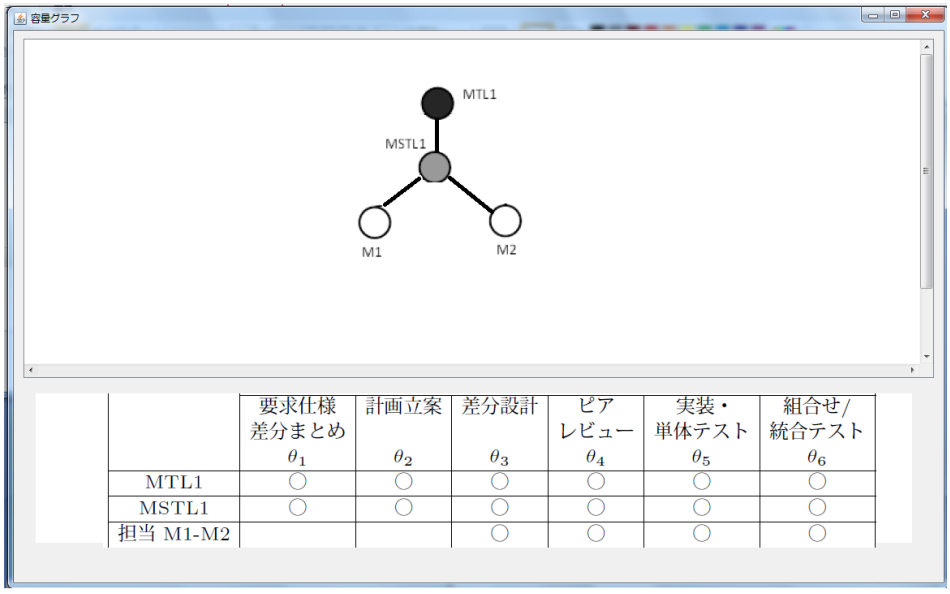


図 2. 11 容量グラフ画面

2.6.3 負荷容量図画面

負荷容量図の作成機能（2.3.6）により、負荷容量図画面は負荷グラフと容量グラフのデータと GLPK の出力から作成した負荷容量図を表示する。表示する負荷容量図の形は図 1.6 のようなになる。負荷容量図画面は図 2.12 のように設計した。

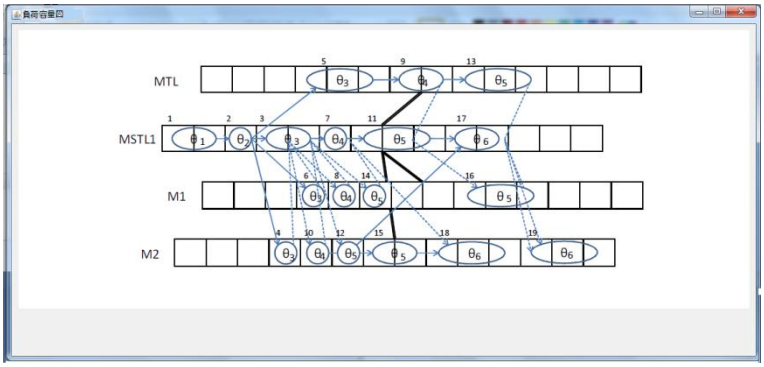


図 2. 12 負荷容量図画面

2.6.4 ガントチャート画面

ガントチャート画面は入力した負荷グラフと容量グラフのデータと GLPK の出力から作成したガントチャートを表示する。LCB が作成するガントチャートの形は図 2.2 のようになる。負荷容量図画面が図 2.13 のように設計した。

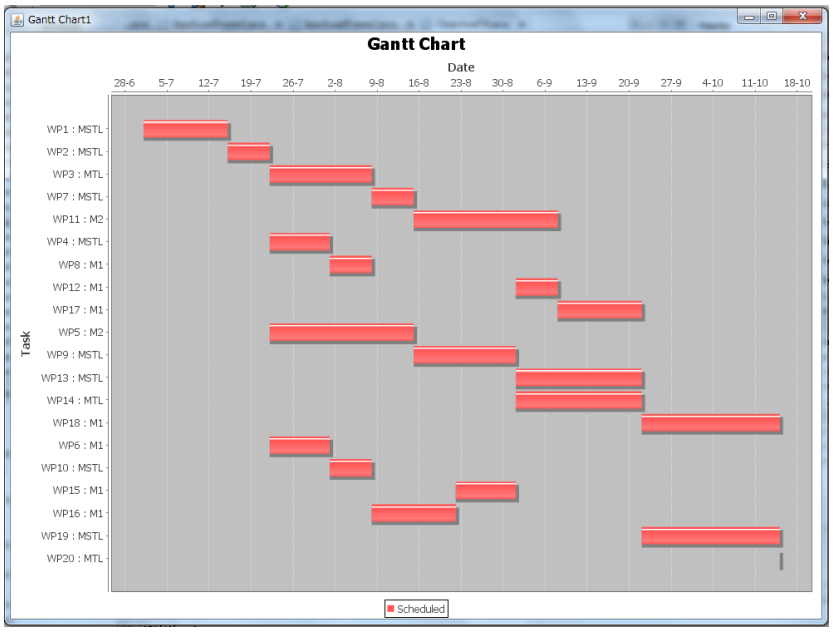


図 2. 13 ガントチャート画面

2.7　全体的な GUI の状態遷移

この筋では入力画面設計・出力画面設計・ユースケース記述に基づき、メイン画面を中心にして、GUI の状態遷移設計について述べる。

- メイン画面と容量グラフ入力画面の関係
    - ユーザはメイン画面の容量グラフの「新規入力」ボタンを押すと容量グラフ入力画面が表示される。
    - ユーザは容量グラフ入力画面の「OK」ボタンを押すと容量グラフ入力画面を閉じ、メイン画面に戻る。

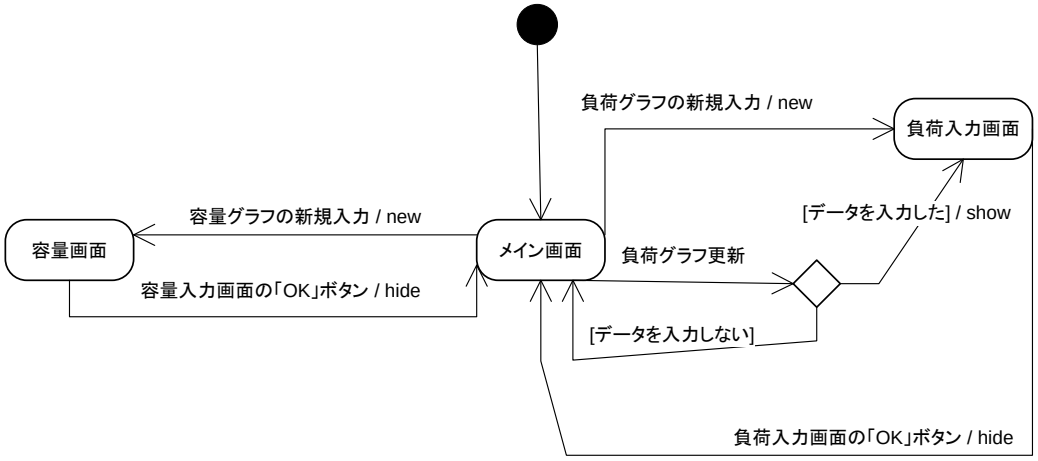


図 2. 14 メイン画面と容量グラフ入力画面と負荷グラフ入力画面の状態遷移

- メイン画面と負荷グラフ入力画面の関係
    - ユーザはメイン画面の負荷グラフの「新規入力」ボタンを押すと負荷グラフ入力画面が表示される。
    - ユーザは負荷グラフ入力画面の「OK」ボタンを押すと負荷グラフ入力画面が閉じ、メイン画面に戻る。
    - ユーザはメイン画面の負荷グラフの「更新」ボタンを押すと、
        - 負荷グラフのデータを入力した場合、負荷グラフ入力画面が表示される。
        - 負荷グラフのデータを入力しなかった場合、メイン画面のままとなる。

これでメイン画面と負荷グラフ入力画面と容量グラフ入力画面の状態遷移が図 2.14 のようになる。

- メイン画面と目的関数入力画面の関係
    - ユーザはメイン画面の目的関数の「新規入力」ボタンを押す

- 容量グラフのデータを入力した場合、目的関数入力画面が表示される。
- 容量グラフのデータを入力しなかった場合、メイン画面のままとなる。

○ ユーザは目的関数入力画面の「OK」ボタンを押すと目的関数入力画面が閉じ、メイン画面に戻る。

○ ユーザはメイン画面の目的関数の「更新」ボタンを押す

- 目的関数を入力した場合、目的関数入力画面が表示される。
- 目的関数入力しなかった場合、メイン画面のままとなる。

これでメイン画面と目的関数入力画面の状態遷移が図は図 2.15 のようになる。

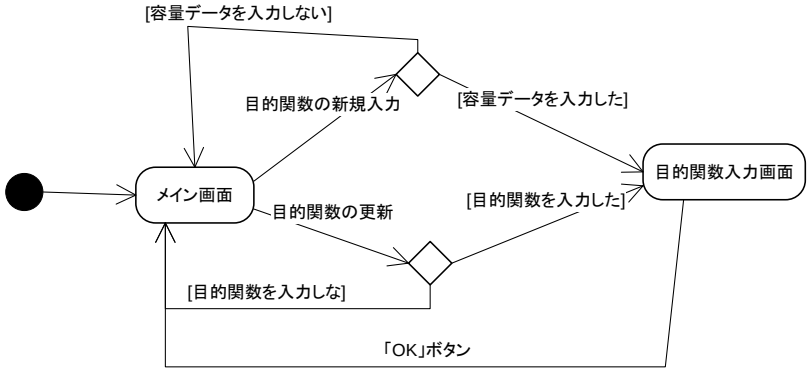


図 2. 15 メイン画面と目的関数入力画面の状態遷移

- メイン画面と負荷グラフ画面の関係

○ ユーザはメイン画面の負荷グラフの「表示」ボタンを押す

- 負荷グラフのデータを入力した場合、負荷グラフを作成して、負荷グラフ画面が表示される。
- 負荷グラフのデータを入力しなかった場合、メイン画面のままとなる。

○ ユーザは負荷グラフ画面を終了すると負荷グラフ画面が閉じ、メイン画面に戻る。

メイン画面と負荷グラフ画面の状態遷移図は図 2.16 のようになる

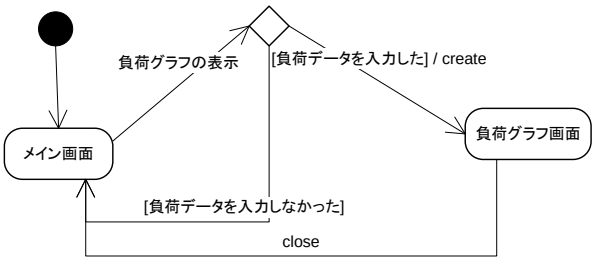


図 2. 16 メイン画面と負荷グラフ画面の状態遷移

- メイン画面と容量グラフ画面の関係
  - ユーザはメイン画面の容量グラフの「表示」ボタンを押す
    - 容量グラフのデータを入力した場合、容量グラフを作成して、容量グラフ画面を表示する。
    - 容量グラフのデータを入力しなかった場合、メイン画面のままとなる。
  - ユーザは容量グラフ画面を終了すると容量グラフ画面が閉じ、メイン画面に戻る。

メイン画面と容量グラフ画面の状態遷移図は図 2.17 のようになる

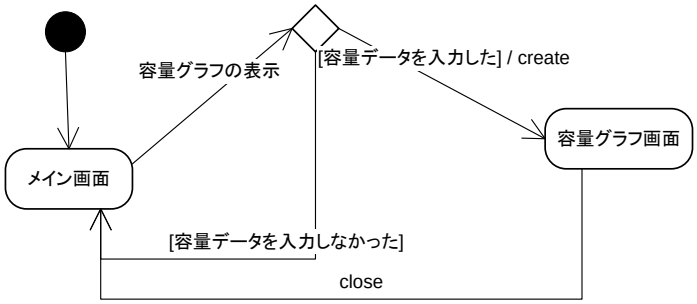


図 2. 17 メイン画面と容量グラフ画面の状態遷移

- メイン画面と負荷容量図画面の関係
  - メイン画面の「負荷容量図を作成」ボタンを押す
    - 三つのデータセット、負荷グラフ・容量グラフ・目的関数を入力した場合、負荷容量図を作成して、負荷容量図画面が表示される。
    - 三つのデータセットを入力しなかった場合、メイン画面のままとなる。
  - 負荷容量図画面を終了すると負荷容量図画面が閉じ、メイン画面に戻る。

メイン画面と負荷容量図画面の状態遷移図は図 2.18 のようになる

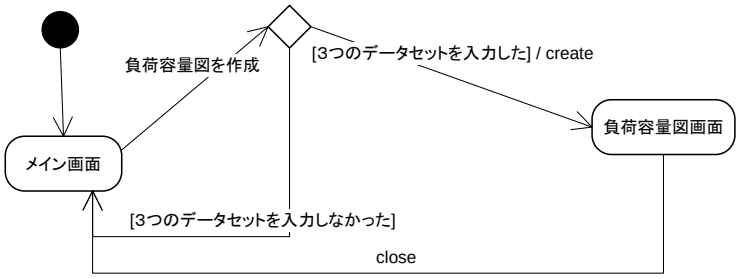


図 2. 18 メイン画面と負荷容量画面の状態遷移

- メイン画面とガントチャートの関係
  - メイン画面の「ガントチャートを作成」ボタンを押す

- 三つのデータセット、負荷グラフ・容量グラフ・目的関数のデータを入力した場合、ガントチャートを作成して、ガントチャート画面が表示される。
- 三つのデータセットを入力しなかった場合、メイン画面のままとなる。

○ ガントチャート画面を終了するとガントチャート画面が閉じ、メイン画面に戻る。

メイン画面とガントチャート画面の状態遷移図は図 2.19 のようになる

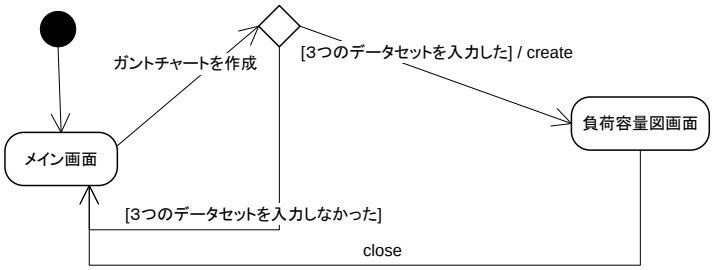


図 2. 19 メイン画面とガントチャート画面の状態遷移

2.8 LCB のクラス図

LCB の構成により、LCB の全体システムは 4 つの部分に分けられる。ツール、組織のデータベース、WBS データベース、GLPK である。しかし、GLPK はすでにあるライブラリであるので、このクラス図設計のパートでは他の三つ、ツールと組織のデータベースと WBS データベース、のサブシステム設計について述べる。

2.8.1 ツールのサブシステム

入出力画面設計結果により、ツールには 8 つの画面がある。一つ一つの画面を実現するために、ひとつずつのクラスが必要となる。GLPK の入力加工と出力加工を行うためには、それぞれのクラスが必要である。従って、ツールのサブシステムは 10 個のクラスを持つ。各クラスの詳細と属性は以下の通りである。ツールのサブシステムクラス図を図 2.20 に示す。

- Main Class

Main Class はメイン画面を実現するためのクラスである。メイン画面は他画面を作成し（create）、呼び出す。このメイン画面は別の属性がなし、他画面を作成する関数と呼び出す関数だけがから成る。一つのボタンは一つの関数を呼ぶ。Main Class の関数は以下の通りである。

関数	詳細
new LoadInput()	負荷グラフの「新規入力」ボタンにより呼び出される関数である。LoadGraphInput クラスを新規作成する。
editLoad ()	負荷グラフの「更新」ボタンにより呼び出される関数である。負荷グラフ入力画面を表示する。
new LoadGraph ()	負荷グラフの「表示」ボタンにより呼び出される関数である。LoadGraph クラスを新規作成する。
addLoadGraph()	新たな負荷グラフの「新規入力」・「更新」・「表示」ボタンを追加する。さらに、「ガントチャートを作成」ボタンを追加する。
new CapacityInput()	容量グラフの「新規入力」ボタンにより呼び出される関数である。CapacityGraphInput クラスを新規作成する。
new CapcityGraph ()	容量グラフの「表示」ボタンにより呼び出される関数である。CapacityGraph クラスを新規作成する。
new ObjectFnInput ()	目的関数の「新規入力」ボタンにより呼び出される関数である。ObjectiveFnInput クラスを新規作成する。
editObjectFn()	目的関数の「更新」ボタンの関数である。目的関数入力画面を表示する。
newGanttChart()	「ガントチャートを作成」ボタンにより呼び出される関数である。GanttChart クラスを新規作成する。
newLoadCapacity()	「負荷容量図を作成」ボタンにより呼び出される関数である。LoadCapacityGraph クラスを新規作成する。

- LoadGraphInput Class

LoadGraphInput Class は負荷グラフ入力画面を実現するためのクラスである。負荷グラフ入力画面では、操作する機能が 8 つある。各機能を実現するための関数が必要となる。入力データを保持する属性を用意する。LoadGraphInput Class の属性と関数は以下の通りである。

属性	詳細
WBSType	ユーザが選択した WBS 型の種類を保持する。
WBSTypeName	ユーザが選択した WBS 型名を保持する。
WBSInstance	ユーザが選択した WBS インスタンスを保持する。
WPArray	照会したデータから作成する WP のアレーを保持する。
effortArray	照会したデータから作成する工数のアレーを保持する。
needSkillArray	照会したデータから作成する必要のアレーを保持する。
edgeParentArray	照会したデータから作成する先行制約の始点 (preceder) のアレーを保持する。

edgeChildArray	照会したデータから作成する先行制約の終点 (successor) のアレーを保持する。
関数	詳細
queryWBSType()	WBS データベースから WBS 型名を照会して、表示する関数。
queryWBSInstance()	WBS データベースから WBS インスタンスを照会して、表す関数。
query WP()	WBS データベースから WP・工数・必要スキル・先行制約を照会する関数。
showQueryData()	照会した WP・工数・必要スキル・先行制約のデータを表示する関数。
create WPArray()	WPArray を作成する関数。
create NeedSkillArray()	needSkillArray を作成する関数。
create EffortArray()	effortArray を作成する関数。
create EdgeArray()	edgeParentArray と edgeChildArray を作成する関数。
save()	入力したデータを負荷グラフデータファイルとして保存する関数。
open()	負荷グラフデータファイルから負荷グラフ入力画面とデータアレーを作成する関数。

- LoadGraph Class

LoadGraphClass は負荷グラフ画面を実現するためのクラスである。負荷グラフ画面は入力した負荷グラフデータから負荷グラフを作成して表示する機能を持つ。従って、LoadGraph Class が属性はなく、負荷グラフを作成する createLoadGraph 関数だけから成る。createLoadGraph の関数は LoadGraphInput Class のアレー属性から、負荷グラフを作成して、表示する関数である。

- CapacityGraphInput Class

CapacityGraphInput Class は容量グラフ入力画面を実現するためのクラスである。容量グラフ入力画面で操作する機能は 8 つある。各機能を実現するための関数が必要となる。そして、全ての入力のデータを保持する属性が必要である。CapacityGraphInput Class の属性と関数は以下の通りである。

属性	詳細
orgName	ユーザが選択した組織名の種類を保持する。
teamType	ユーザが選択した機能チームの種類を保持する。
teamName	ユーザが選択した機能チーム名を保持する。
subLeaderNo	ユーザが入力したサブリーダの人数を保持する。
resourceArray	照会したデータから作成するリソース名のアレーを保持する。
resourcePositionArray	照会したデータから作成するリソースの位置のアレーを保持する。
edgeParentArray	照会したデータから作成する supervisor のアレーを保持する。

edgeChildArray	照会したデータから作成する supervisor の部下 のアレーを保持する。
skill2DArray	照会したデータから作成するリソースの保持スキル情報のアレーを保持する。
関数	詳細
queryOrgName()	組織のデータベースから組織名を照会する。
queryTeamType()	組織のデータベースから機能チームの種類を照会する。
query TeamName()	組織のデータベースから機能チーム名を照会する関数。
queryMemberData()	組織のデータベースからリソース名・supervisor を照会する関数。
queryMemberSkill()	組織のデータベースから保持スキルの情報を照会する関数。
createResourceArray ()	ResourceArray を作成する関数。
createSkill2DArray ()	skill2DArray を作成する関数。
createEdgeArray()	edgeParentArray と edgeChildArray を作成する関数。
save()	入力したデータを負荷グラフデータファイルとして保存する関数。
open()	負荷グラフデータファイルから容量グラフ入力画面とデータアレーを作成する関数。

- CapacityGraph Class

CapacityGraph Class は容量グラフ画面を実現するためのクラスである。容量グラフ画面は入力した容量グラフデータから容量グラフを作成して表示する機能を持つ。CapacityGraph Class の属性はなく、容量グラフを作成する createLoadGraph 関数から成る。createCapacityGraph 関数は CapacityGraphInput Class のアレー属性から、容量グラフを作成して、表示する関数である。

- ObjectiveFunction Class

ObjectiveFunction Class は目的関数入力画面を実現するためのクラスである。目的関数入力画面で操作する機能は 4 つある。リソース割り当てる表を作成機能・リソースマッチング部分を作成する機能・リソース割り当てアレーを作成する機能・リソースマッチングのアレーを作成する機能である。作成したアレー（リソースマッチングのアレー・リソース割り当てアレー）を保持する属性が二つ必要となる。

属性	詳細
resourceAssignTabel	作成されたリソース割り当てる表を保持する。
assignArray	表で入力したデータから作成すされたリソース割り当てアレーを保持する。
keyPersonArray	入力したキーパーソンのデータから作成されたキーパーソンのアレーを保持する。
pairMemberArray	入力したペアメンバーのデータから作成されたペアメンバーのアレーを保持する。
関数	詳細
createAssignTabel()	組織のデータベースから組織名を照会する関数。
createResourceMatch()	組織のデータベースから機能チームの種類を照会する関数。

addResourceMatch ()	組織のデータベースから機能チーム名を照会する関数。
createKeyPersonArray ()	組織のデータベースからリソース名・supervisor を照会する関数。
createPairMemberArray ()	組織のデータベースから保持スキルの情報を照会する関数。

- GLPKInput Class

GLPKInput Class は GLPK 入力データの加工機能を実現するためのクラスである。GLPK 入力データの加工機能は入力したデータから制約式・目的関数を作成する。そして、準備しておく決定変数と作成した制約式・目的関数を、GLPK に入力する。

以上により、GLPKInput Class の必要な属性と関数は以下の通りである。

属性	詳細
timeDecision2DArray	準備しておく時間割り当ての決定変数（x_{il}）である。
resourceDecision2DArray	準備しておくリソース割り当ての決定変数（b_{ir}）である。
constraintArray	入力した負荷グラフ・容量グラフのデータから作成された制約式モデルを保持する。
objectiveArray	入力した目的関数のデータから作成された目的関数を保持する。
関数	詳細
createConstraintArray()	入力した負荷グラフ・容量グラフのデータから constraintArray を作成する関数。
createConstraintArray()	入力した目的関数のデータから objectiveArray を作成する関数。
inputGLPK()	timeDecision2DArray・resourceDecision2DArray・constraintArray・objectiveArray を、GLPK に入力する。

- GLPKOutputDerive Class

GLPKOutputDerive Class は GLPK 出力を加工する機能を実現ためのクラスである。GLPK 出力の加工機能は 2 つの GLPK の出力（.sol）から WP の開始時刻とリソース割り当てのアレーを作成する機能である。GLPK に対して、属性と関数が必要となるので、GLPKOutputDerive Class は 2 つの属性と 2 つの関数がある。以下の通りである。

属性	詳細
startTimeArray	GLPK の出力から作成する WP の開始時刻のアレーである。
resoruceBindArray	GLPK の出力から作成するリソース割り当てのアレーである。
関数	詳細
createStartTimeArray ()	GLPK の出力を加工して、startTimeArray を作成する。
createResoruceBindArray ()	GLPK の出力を加工して、resoruceBindArray を作成する。

- GanttChart Class

GanttChart Class はガントチャート画面を実現するためのクラスである。ガントチャート画面は入力した負荷グラフ・容量グラフのデータを GLPK に入力して、入力したデータと GLPK の出力からガントチャートを作成する機能を持つ。GLPK に入力するのために、GLPKInput Class を利用する。そして、GLPK の出力を加工するために、GLPKOutputDerive Class を利用する。GanttChart Class の属性はなく、ガントチャートを作成する createGanttChart 関数だけ必要から成る。

- LoadCapacityGraph Class

LoadCapacityGraph Class は負荷容量図画面を実現するためのクラスである。負荷容量図画面はユーザが入力した負荷グラフ・容量グラフのデータを GLPK に入力して、入力したデータと GLPK の出力から負荷容量図を作成する機能を持つ。GLPK にデータを入力するのために、GLPKInput Class を利用する。そして、GLPK の出力を加工するために、GLPKOutputDerive Class を利用する。LoadCapacityGraph Class の属性はなく、負荷容量図を作成する create LoadCapacityGraph 関数だけ必要である。

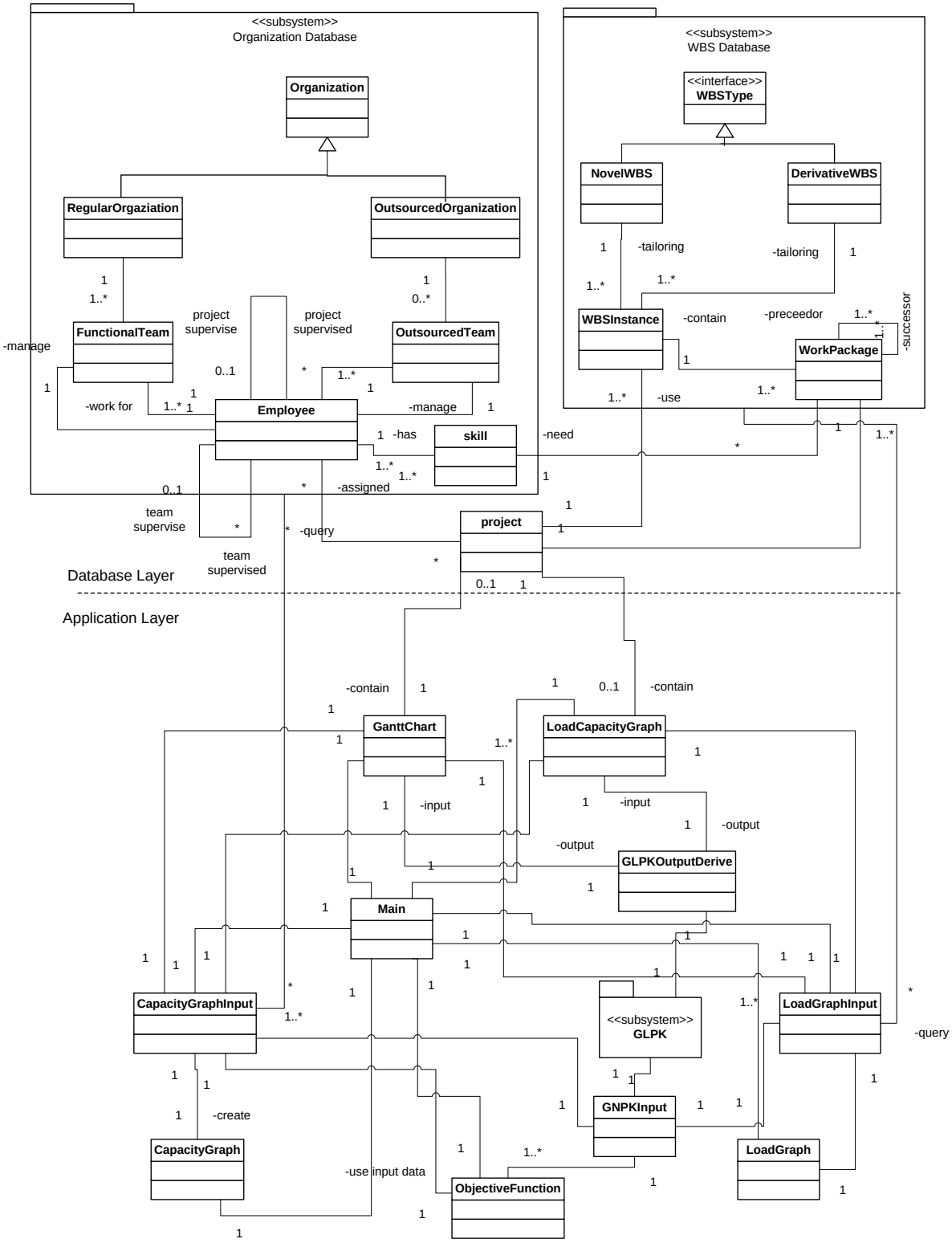


図 2. 20 ツールのサブシステムのクラス図

2.8.2 WBS Database サブシステム

負荷容量参照モデルでは、負荷を生み出す構造は 3 つの主なコンポーネントから成る、WBS 型・WBS インスタンス・WP である。WBS 型には 2 つ種類があり、新規開発の WBS 型と派生開発の WBS 型である。WBS Database サブシステムは 1 つの abstract class と 4 つの concrete class が必要となる。各クラスの詳細は以下の通りである。

- WBSType Class

  WBSType Class は WBS 型に対応するクラスである。LCB に関する必要な属性は WBSName・WBSID・WBSType である。それぞれの属性はデータベースの照会のために利用される。

- NovelWBS Class

  NovelWBS Class は新規開発 WBS 型に対応するクラスである。NovelWBS は WBSType の特化クラスとなる。LCB に関する必要な関数は新たな WBS 型を作成する関数、new()である。

- DerivateWBS Class

  DerivateWBS Class は派生開発 WBS 型に対応するクラスである。DerivateWBS も WBSType の特化クラスとなる。LCB に関する必要な関数は既存している WBS 型を編集する関数、edit()である。

- WBSInstance Class

  WBSInstance Class は WBS インスタンスに対応するクラスである。WBSInstance は DerivateWBS・NovelWBS とテーラーリング関係がある。LCB に関する必要な属性は WBSInstanceName と WBSInstanceID である。それぞれの属性はデータベースの照会のためである。

- WorkPackage Class

  WorkPackage Class は WP に対応するクラスである。だから、WorkPackage は WBSInstance と関係がある。LCB に関する必要な属性は WPName・WPID・needSkill・effort・preceder・succesor である。WPName・WPID はデータベースの照会のために利用される。そして、needSkill・effort は WP の属性である。preceder・succesor は先行制約を示す。

作成した WBS Database サブシステムのクラス図を図 2.21 に示す。

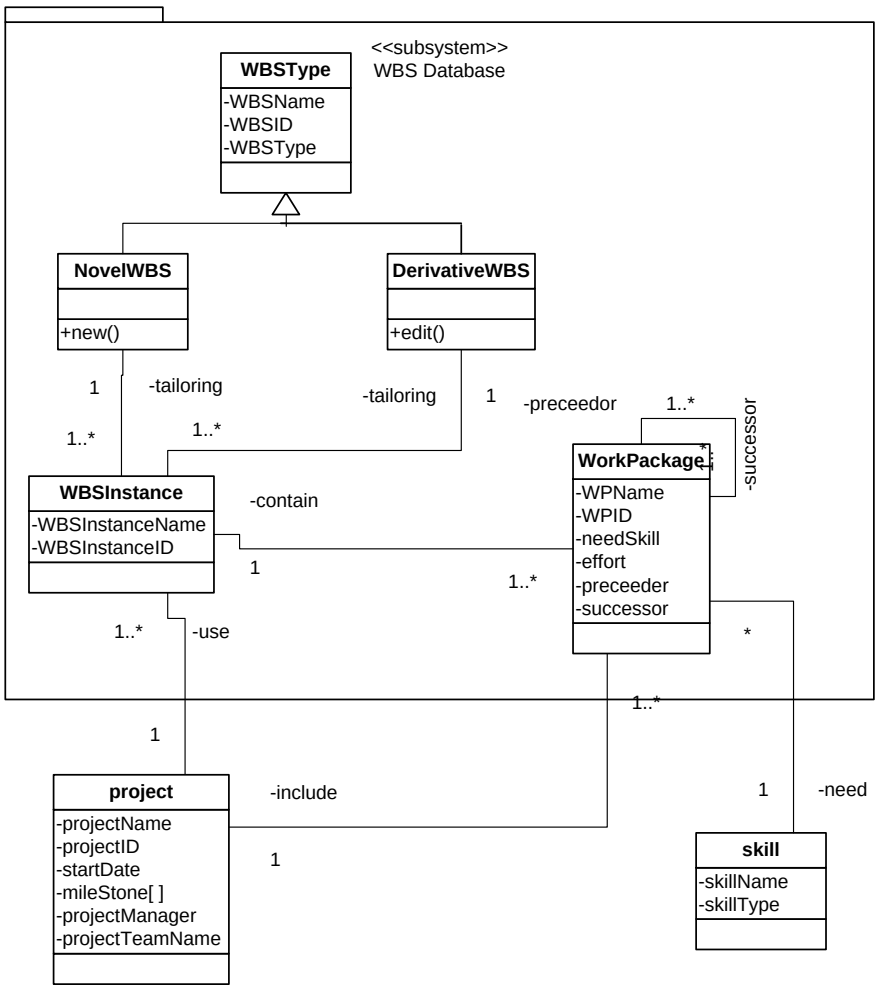


図 2. 21 WBS Database サブシステムのクラス図

2.8.3 Organization Database サブシステム

負荷容量参照モデルでは、容量を生み出す構造は 3 つの主なコンポーネントからなる。組織・機能チーム・リソース（社員）である。組織は 2 つの種類、プロパーと外部委託先に分かれている。プロパー・外部委託先のクラス ,RegularOrganization と OutsourcedOgranization クラスとする。プロパーの機能チームを外部委託先の機能チームを区別する必要があるため、機能チームのクラスは 2 つ必要となる。Functional Team と Outsourced Team クラスである。機能チームを持つリソースは Employee クラスで示す。リソースが持つスキルは skill クラスで示す。従って、Organization Database サブシステムは 7 つのクラスを持つ。Organization Database サブシステムのクラス図を図 2.22 に示す。

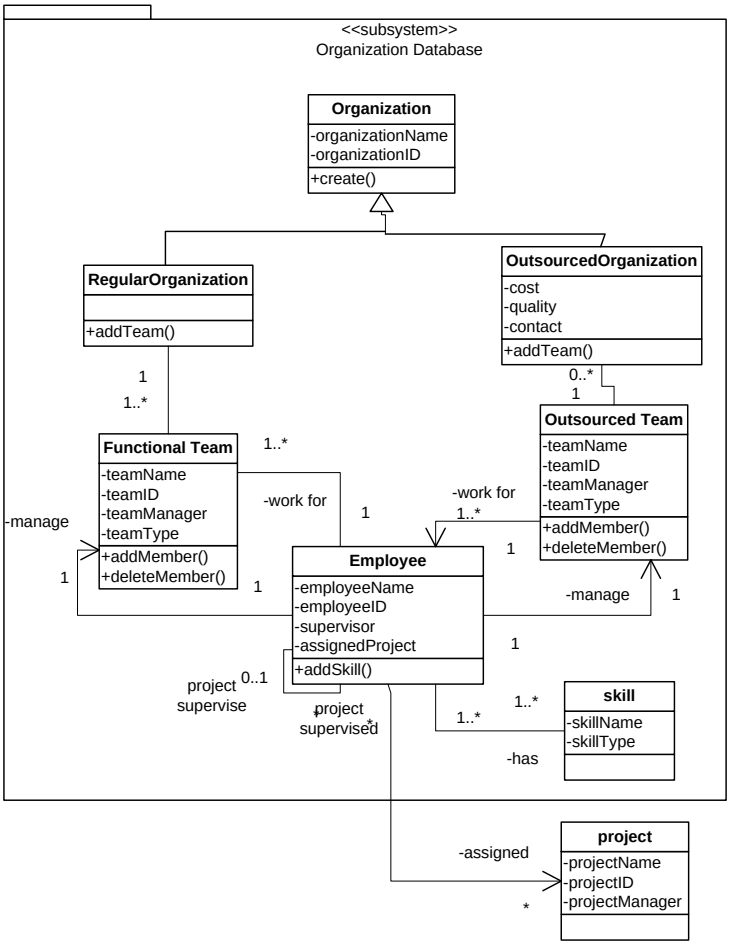


図 2. 22 Organization Database サブシステムのクラス図

2.8.4 クラス図

三つのサブシステムのクラス図を組み合わせて、LCB の全体のクラス図を図 2.23 のように設計する。

2.9　データベース設計

本ツールには 2 つのデータベースがあり、組織のデータベースと WBS のデータベースである。この 2 つのデータベースはクラス図と負荷容量参照モデルを基に設計される。本修士論文でのデータベース設計は ER Diagram と Relational Database で表現する。

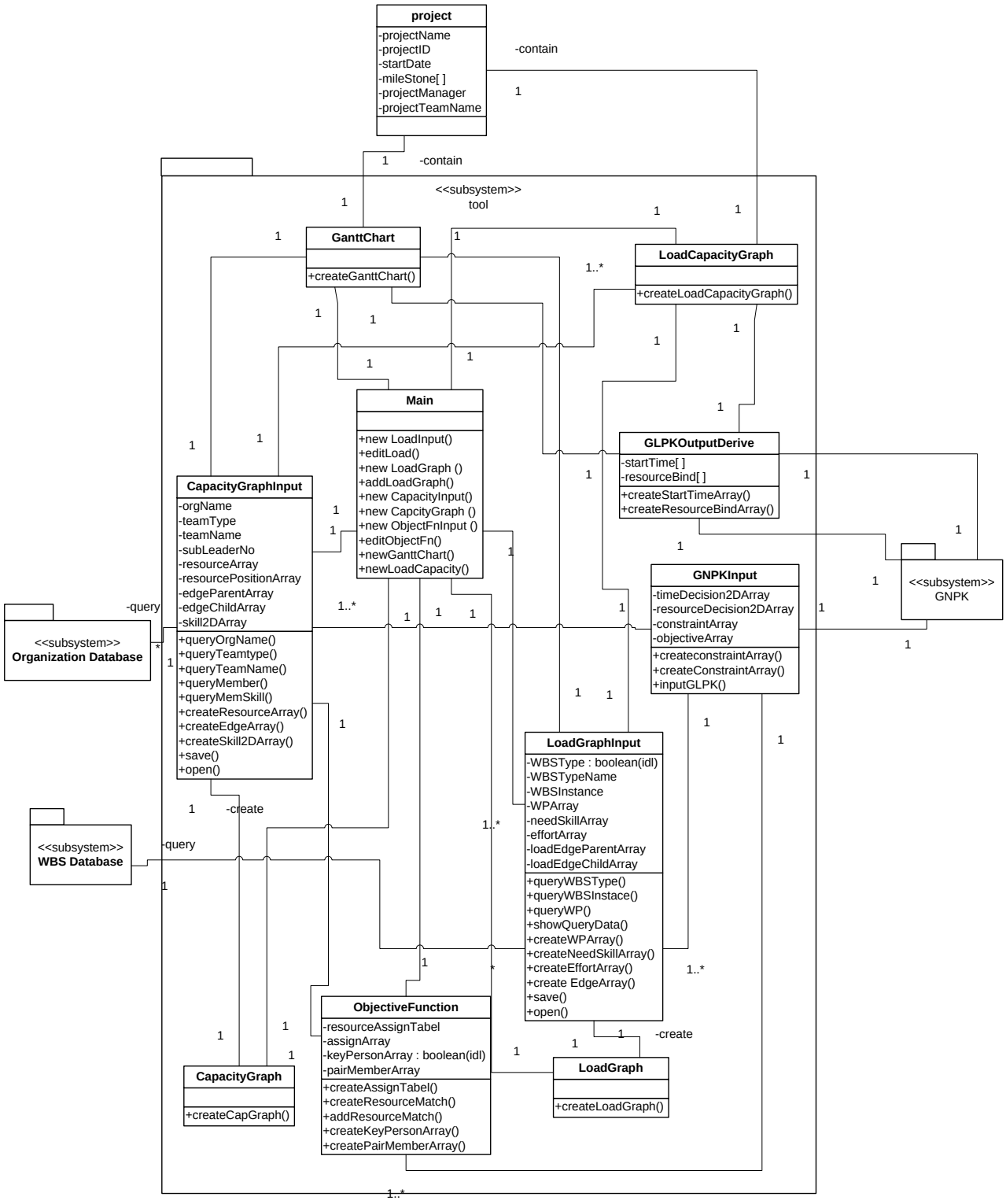


図 2. 23 LCB の全体のクラス図

2.9.1 組織のデータベース設計

組織のデータベースはプロパー・外部委託先・機能チーム・社員・スキルのデータを保持する。それぞれの属性や関係などは Organization Database サブシステムのクラス図（図 2.22）の通りである。

Organization Database サブシステムのクラス図には 7 つのエンティティがあり、Organization・RegularOrganization・OutsourcedOrganization・FunctionalTeam・Employee・OutsourcedTeam・Skill である。それぞれのデータは一つ一つの ER のエンティティとなる。各エンティティの名前はクラス図と同じである。各エンティティの属性はクラスの属性と同じである。そして、クラス図の関係（Association）と負荷容量参照モデルの構造を基に、エンティティの関連(Relationship)を設計する。以上により、組織のデータベースの ER Diagram を図 2.24 のように設計した。

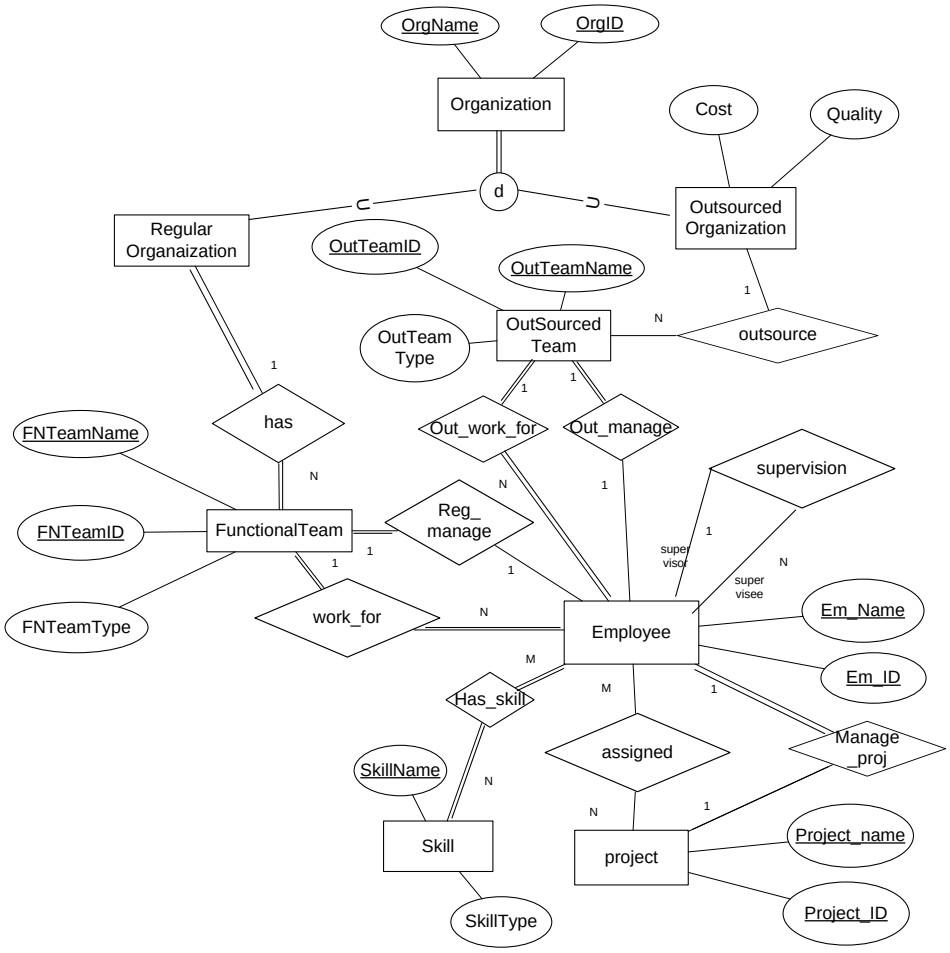


図 2. 24 組織のデータベースの ER Diagram

2.9.2 WBS のデータベース設計

WBS のデータベースは WBS 型・WBS インスタンス・WP のデータを保持する。それぞれの属性や関係など WBS サブシステムのクラス図（図 2.21）の通りである。

WBS サブシステムのクラス図には 5 つのクラスがあり、WBSType・Novel WBS・DerivedWBS・WBSInstance・Workpage である。それぞれのデータが一つ一つの ER のエンティティとなる。各エンティティの名前はクラス図と同じである。各エンティティの属性はクラスの属性と同じである。そして、エンティティの関連(Relationship)はクラス図の関係（Association）からを設計する。以上により、WBS のデータベースの ER Diagram を図 2.25 のように設計した。

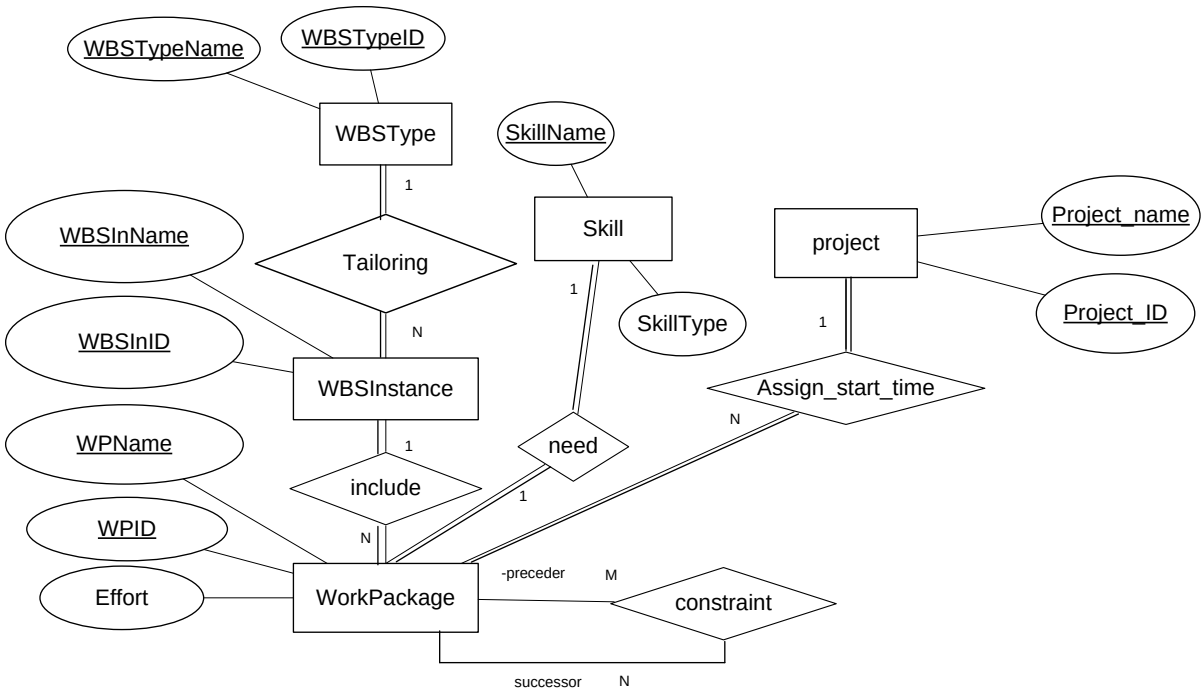


図 2. 25 WBS のデータベースの ER Diagram

2.9.3 LCB データベースの全体像

組織のデータベースの ER Diagram（図 2.24）と WBS のデータベースの ER Diagram（図 2.25）を統合した LCB のデータベース全体の ER Diagram を図 2.26 に示す。

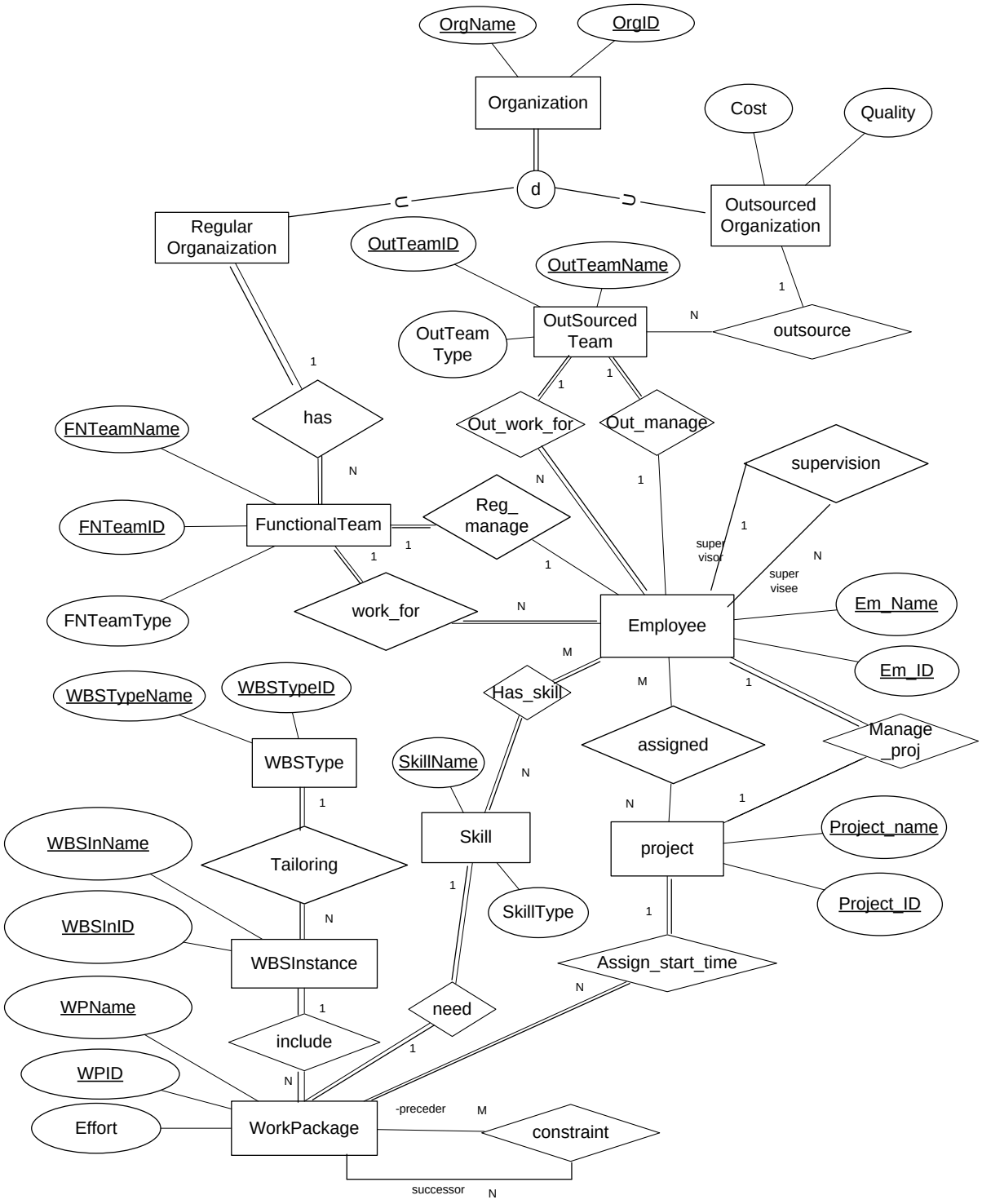


図 2. 26 LCB のデータベース全体の ER Diagram

2.9.4 リレーショナルデータベース設計

本修士論文のリレーショナルデータベース設計は ER 図式から変更で設計する。ER 図式から関係スキーマへの変換方法は以下の通りである。

1. 各エンティティを 1 つずつ関係に変換する
   - エンティティの名前を，関係名とする
   - エンティティの属性を，対応する関係の属性とする
   - エンティティのキーを，関係の主キーとする
2. リレーションシップは、カーディナリティにより、「1 対 1」「1 対多」「多対多」のいずれかに分類できる。
   - １対１、１対多のときは、
     １のほうが、主キー、多のほうが外部キーになることで表す。
   - 多対多のとき
     A テーブルと B テーブルが多対多のとき、A テーブルのどのレコードと B テーブルのどのレコードが対応しているかを表したテーブル、AB 対応テーブルを作成する。

そして、作成したリレーショナルデータベースを、以下の第 1 正規化から第 3 正規化で正規化する。

- 第 1 正規化

関係 (リレーション) がスカラ値のみを持ちうるとき、そのリレーションを第 1 正規形 (first normal form; 1NF) であるという。スカラ値とはそれ以上分割できない値のことをいい、単一の数値や単語は一般にスカラ値だが、表や配列、カンマで区切った文字列などはふつうスカラ値ではない。第 1 正規形を満たさないリレーションは、その中の値を必ずしもリレーショナル演算 (関係代数ないし関係論理による演算) の対象とすることができないという問題を持つ。

- 第 2 正規化

あるリレーションが、第 1 正規形で、かつ、すべての非キー属性が、すべての候補キーに対して完全従属するとき、第 2 正規形 (second normal form; 2NF) であるという。つまり、第 2 正規形では、候補キーの一部に関数従属する非キー属性があってはならない。

- 第 3 正規化

あるリレーションが、第 2 正規形で、かつ、非キー属性があるならば、それら全てが候補キーに非推移的に関数従属するとき、第 3 正規形 (third normal form; 3NF) であるという。

定義：A → B とは、A を入力すると、B を照会できる。

候補キーA 及び非キー属性 B, C を含むリレーションがあり、A → B かつ B → C のとき、C は候補キーA に推移的に関数従属するという。推移的に従属する属性に従属する非キー属性も同様である。非推移的に従属するとは、関数従属するが推移的に関数従属してはいないことをいう。

第 3 正規化 LCB のリレーショナルデータベースを図 2.27 のように設計する。

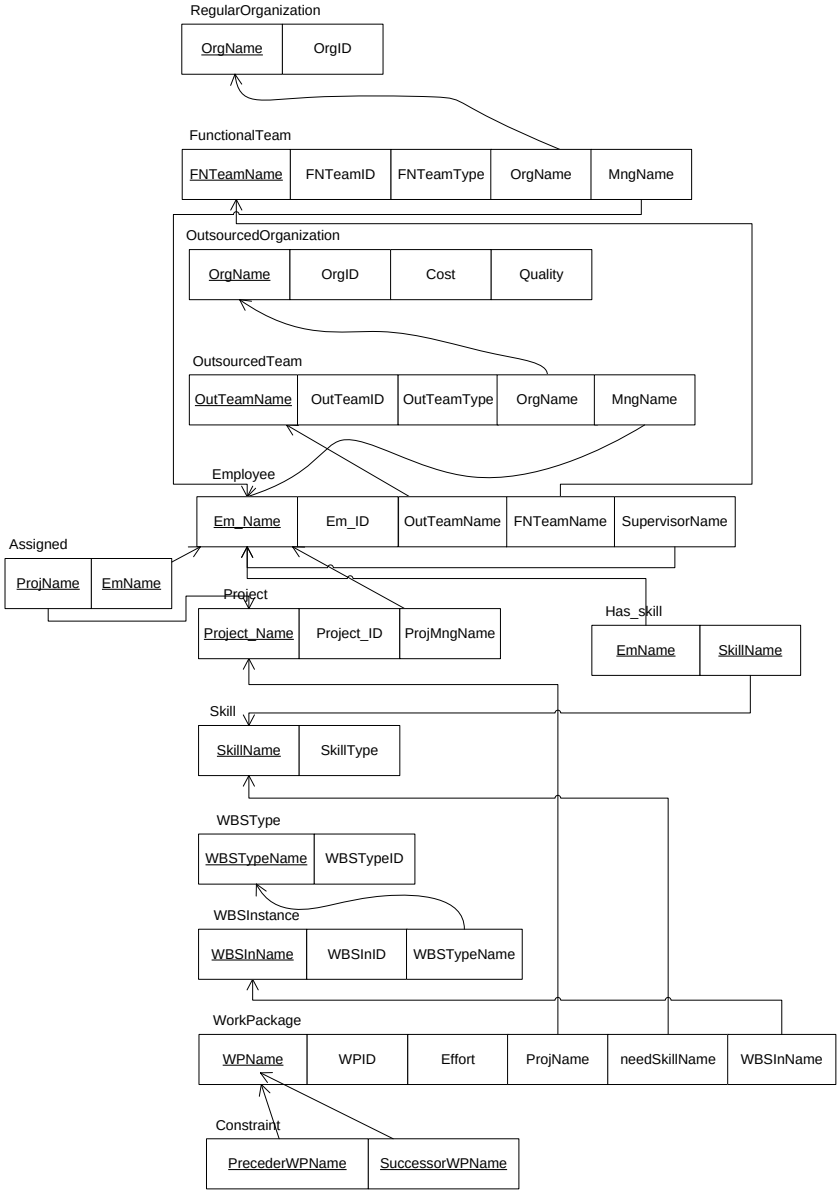


図 2. 27 LCB のリレーショナルデータベース

2.3.10 コミュニケーション図

クラス間のコミュニケーションがよくあるユースケースが 5 つあり、負荷グラフデータを入力する UC01・容量グラフデータを入力する UC08・目的関数を入力する UC13・ガントチャートを作成する UC20・負荷容量図を作成する UC21 のユースケースである。それぞれのユースケースのコミュニケーションをコミュニケーション図で示す。

2.3.10.1 容量グラフデータを入力するコミュニケーション図

容量グラフデータを入力するユースケースの記述（表 2.10）と全体クラスにより、容量グラフデータを入力するユースケースのコミュニケーション図を図 2.28 のように設計する。

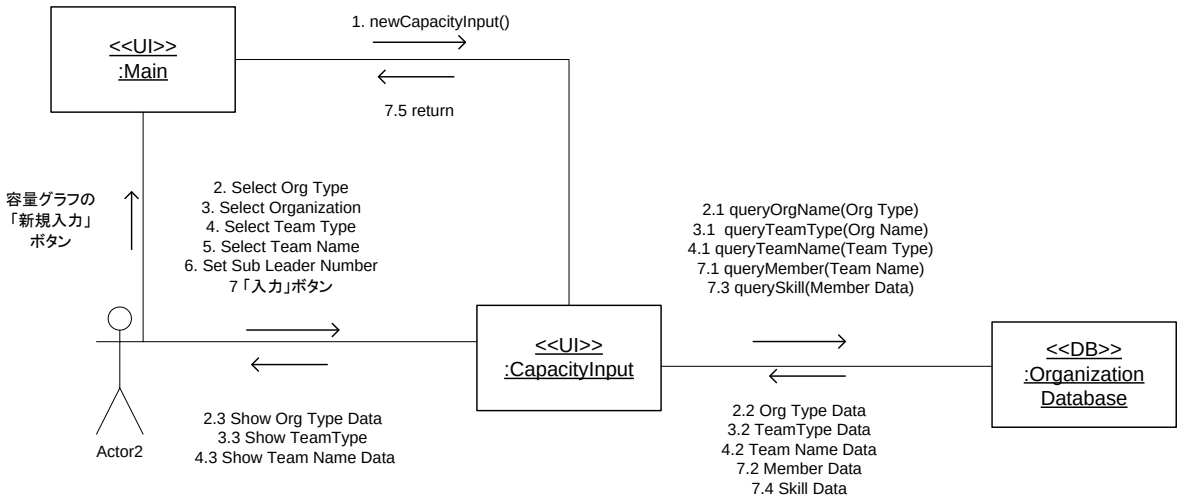


図 2. 28 容量グラフデータを入力するユースケースのコミュニケーション図

2.3.10.2 負荷グラフデータを入力するコミュニケーション図

負荷グラフデータを入力するユースケースの記述（表 2.10）と全体のクラス図により、負荷グラフデータを入力するユースケースのコミュニケーション図を図 2.29 に示す。

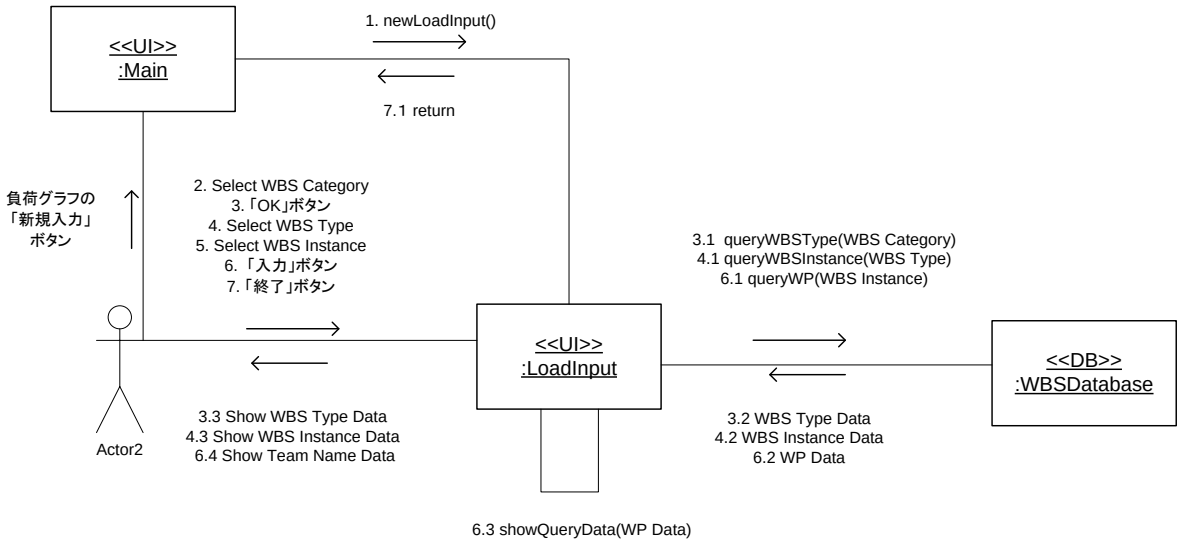


図 2. 29 負荷グラフデータを入力するユースケースのコミュニケーション図

2.3.10.3 目的関数を入力するコミュニケーション図

目的関数を入力するユースケースの記述（表 2.18）と全体のクラス図により、目的関数を入力するユースケースのコミュニケーション図を図 2.30 に示す。

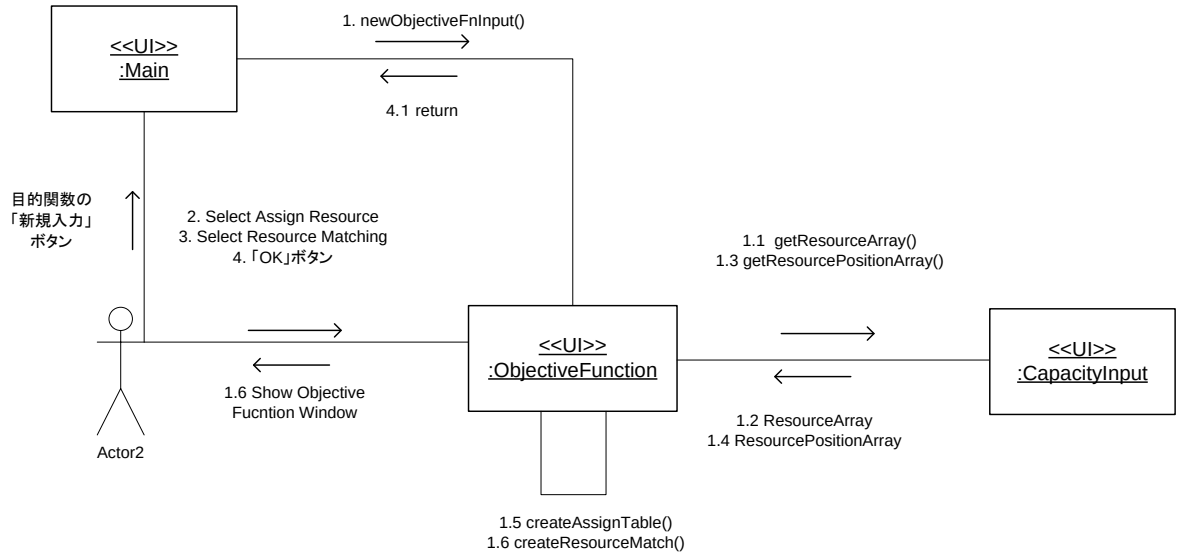


図 2. 30 目的関数を入力するユースケースのコミュニケーション図

2.3.10.4 負荷容量図を作成するコミュニケーション図

負荷容量図を作成するユースケースの記述（表 2.26）と全体のクラス図により、負荷容量図を作成するユースケースのコミュニケーション図を図 2.31 に示す。

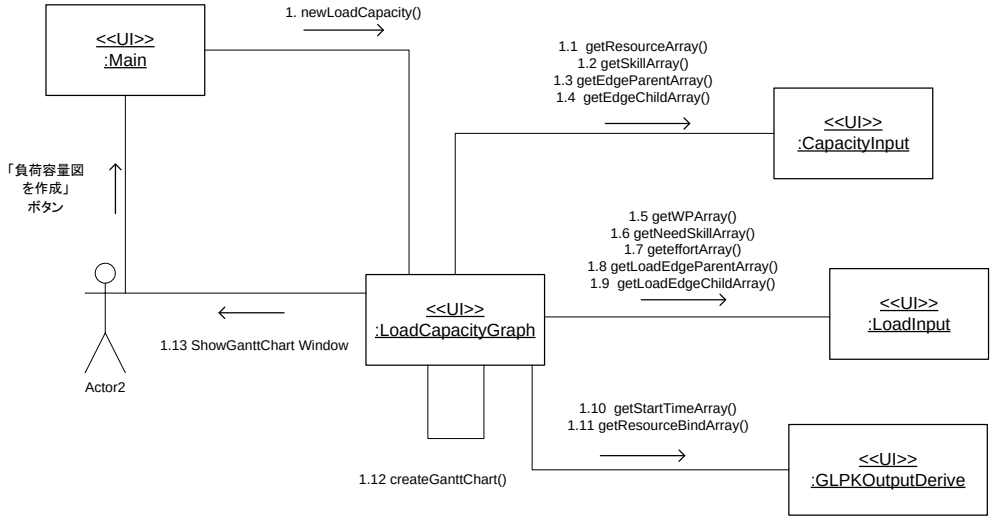


図 2.31 負荷容量図をを作成するユースケースのコミュニケーション図

2.3.10.5 ガントチャートを作成するコミュニケーション図

ガントチャートを作成するユースケースの記述（表 2.25）と全体のクラス図により、ガントチャートを作成するユースケースのコミュニケーション図を図 2.32 に示す。

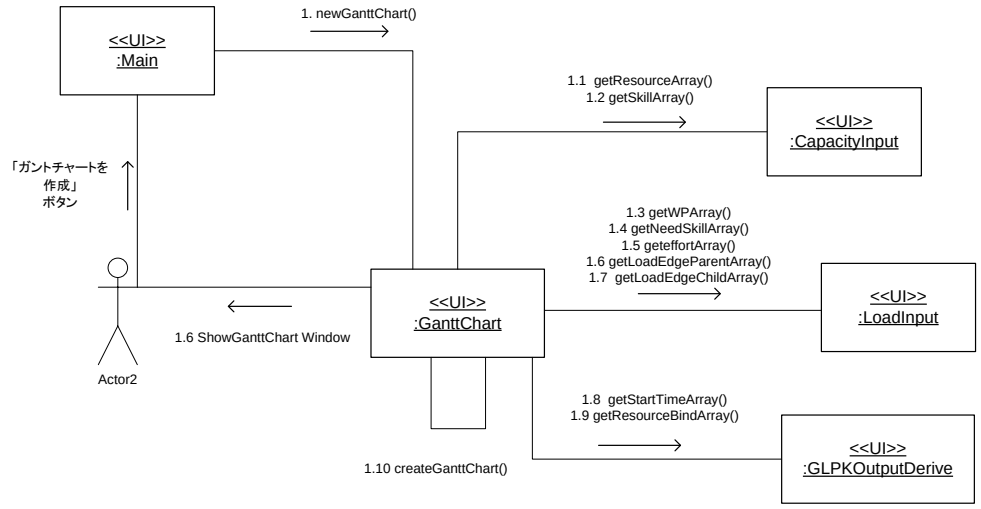


図 2.32 ガントチャートを作成するユースケースのコミュニケーション図

第 3 章 実現方法

3.1 負荷グラフと容量グラフの作成方法

負荷グラフと容量グラフの作成方法は幅優先探索のアルゴリズムでグラフを割り付ける。

- 幅優先探索(Breadth first search)

幅優先探索(はグラフ理論(Graph theory)において木構造(tree structure)やグラフ(graph)の探索に用いられる。アルゴリズムは根ノードで始まり隣接した全てのノードを探索する。それからこれらの最も近いノードのそれぞれに対して同様のことを繰り返して探索対象ノードをみつける。「横型探索」とも言われる。図 3.1 が幅優先探索の順番を表示する。

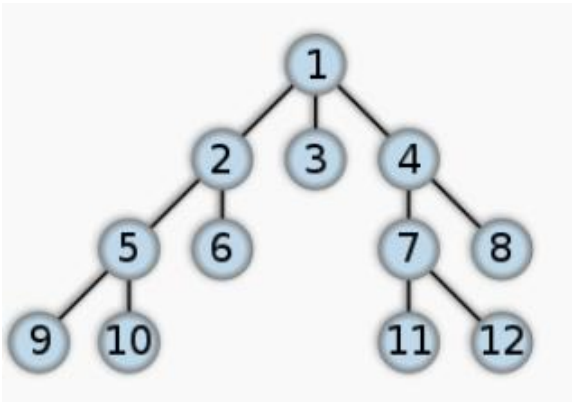


図 3. 1 幅優先探索

幅優先探索のアルゴリズムは以下の通りである。

1) 根ノードを空のキューに加える。
2) ノードをキューの先頭から取り出し、以下の処理を行う。
    - ノードが探索対象であれば、探索をやめ結果を返す。
    - そうでない場合、ノードの子で未探索のものを全てキューに追加する。
3) もしキューが空ならば、グラフ内の全てのノードに対して処理が行われたので、探索をやめ"not found"と結果を返す。
4) 2 に戻る。

以上のアルゴリズム、から以下のような擬似コード (pseudo-code) を作成する。

```
bfs(v)
    enqueue(v)
    mark v as visited
    while queue not empty
        v=dequeue()
        process(v)
        for all unvisited vertices i adjacent to v
            mark i as visited
            enqueue(i)
```

負荷グラフと容量グラフを以下の方針で作成する。

1. 画面の広さ(width)と高さ(height)を決める。
2. グラフレベルの深さ(levelHeight)を決める。
3. 各レベルに存在するノード(currentLevelSize)の数を計算する。
4. BFS（Breath First Search）の順番で各頂点の位置を計算する。
5. 各頂点の画面表示位置により、頂点を作成する。
6. 辺のデータにより、始点・終点の画面表示位置における辺を作成する。

- 容量グラフの計算方法：

定義：CLCT (Current Level Calculation Time)とは、現在の深さの探索位置を示す変数。他追えば、図 3.1 のノード 3 を計算すると、CLCT は 2 となる。ノード 8 を計算すると、CLCT は 4 となる。

容量グラフの頂点の位置を計算する方法は以下

```
X = (width*(CLCT)) / (currentLevelSize+1);
Y = (currentLevel-1)* levelHeight
```

ノート：currentLevelSize+1 の理由

CLCT = currentLevelSize のとき、ノードが画面の中に表示するために、1 を加える。

例：画面のサイズを 500X500 として、グラフレベルの高さが 100 とする。図 3.1 のノード 1 から 8 までの位置は以下のように計算する。

ノード 1:　X = (500/2) = 250,

Y = (1-1)*100 =0

ノード 2:　X = (500*1) / (3+1) =125,

Y = (2-1)*100 =100

ノード 3:　X = (500*2) / (3+1) =250,

Y = (2-1)*100 =100

ノード 4:　X = (500*3) / (3+1) =375,

Y = (2-1)*100 =100

ノード 5:　X = (500*1) / (4+1) =100,

Y = (3-1)*100 =200

ノード 6:　X = (500*2) / (4+1) =200,

Y = (3-1)*100 =200

ノード 7:　X = (500*3) / (4+1) =300,

Y = (3-1)*100 =200

ノード 8:　X = (500*4) / (4+1) =400,

Y = (3-1)*100 =200

計算した結果は図 3.2 に示すようになる。

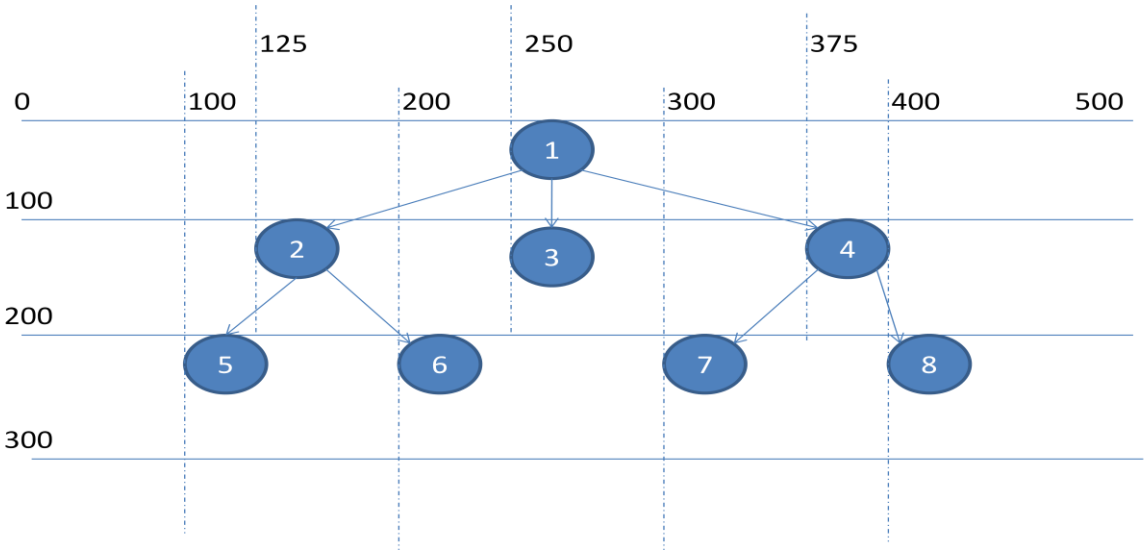


図 3. 2 容量グラフの頂点の位置の計算例

- 負荷グラフの計算方法：

負荷グラフは横型で表示するから、計算方法は容量グラフと逆である。つまり、容量グラフの X 軸計算方法は負荷グラフの Y 軸計算方法となる。そして、容量グラフの Y 軸の計算方法は負荷グラフの X 軸の計算方法となる。

負荷グラフの頂点の位置を計算する方法は以下の通りである。

```
Y = (width*(CLCT)) / (currentLevelSize+1);
X = (currentLevel-1)* levelHeight
```

3.2 負荷容量図の作成方法

負荷容量参照モデルにより、負荷容量図には以下のコンポーネントがある。

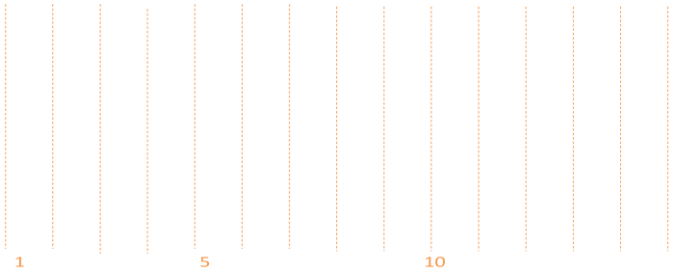
- リソース（容量グラフの頂点）
- 組織の関係（容量グラフの辺）
- 作業負荷（負荷グラフの頂点）
- タイムスロット
- 作業の先行制約（負荷グラフの辺）
- コミュニケーションパス

コミュニケーションパス：作業を遂行するためのコミュニケーション伝達業務を可視化したものである。作業間に依存関係（先行制約）のある始点と終点が違うリソースに割り当てられ時、作業の先行制約線の変わりにコミュニケーションパス線で表示する。

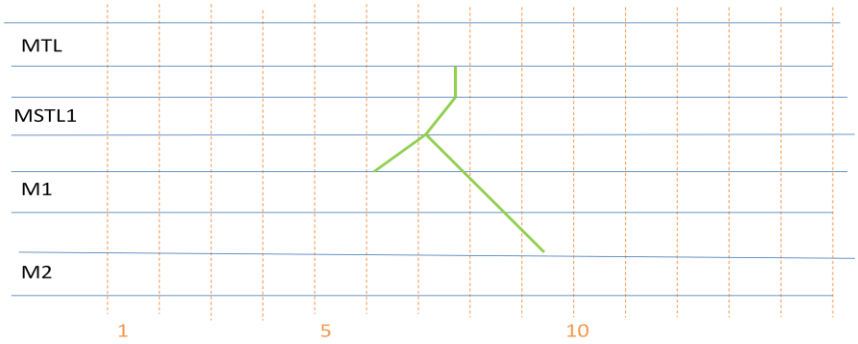
負荷容量図を作成する際、それぞれのコンポーネントは以下の方針で作成する。（以下の例は図 1.5 のような負荷容量図を作成する例である。）

負荷容量図を作成する方針：

1. タイムスロットを表示する線と番号を作成する。以下の通りである。



2. 幅優先探索の順番で、上から下まで、リソースのブロックを作成する。そして、組織の関係（職階）を示す線を作成する。以下に例を示す。



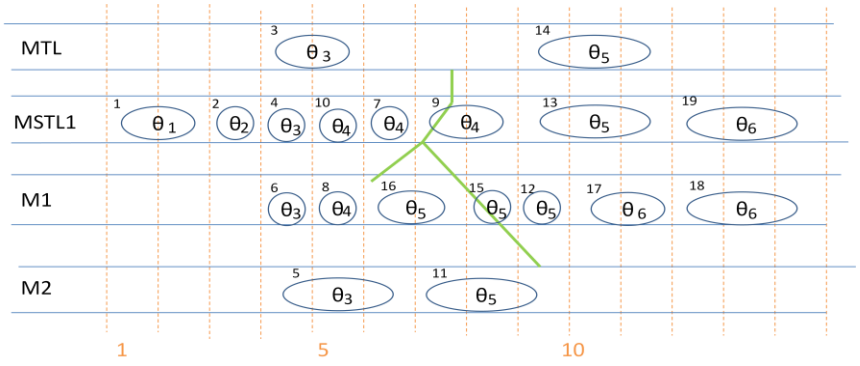
3. GLPK の出力から負荷グラフの各頂点位置を計算する。そして、頂点の工数の属性により、負荷グラフの頂点の径を求めて、作成する。

   計算方法：

   - GLPK の作業開始時刻の出力ファイル（x_{il}）から、負荷グラフの頂点を割り当てるスロットの情報を取得する。
   - GLPK のリソース割り当ての出力ファイル（b_{ir}）から、負荷グラフの頂点の割り当てるリソースの情報を取得する。割り当てるリソースのブロック y の位置は頂点の y の位置となる。

負荷グラフの頂点の x = 割り当てるタイムスロットの x の位置 負荷グラフの頂点の y = 割り当てるリソースのブロック y の位置

   以上の計算方法で以下の例のような図を作成する。



4. 作業の先行制約線を作成する。
始点と終点が同じリソースに割り当てられた場合、始点と終点の位置により、作業の先行制約線を作成する。

5. コミュニケーションパスを作成する。
始点と終点が別のリソースに割り当てられた場合、コミュニケーションパスを作成する。コミュニケーションパスの探索は深さ優先探索でする。

- 深さ優先探索（depth-first search, DFS）

深さ優先探索は、木やグラフを探索するためのアルゴリズムである。アルゴリズムは根から(グラフの場合はどのノードを根にするか決定する)始まり、バックトラックするまで可能な限り探索を行う。「縦型探索」とも呼ばれる。図 3.3 二例を示す。

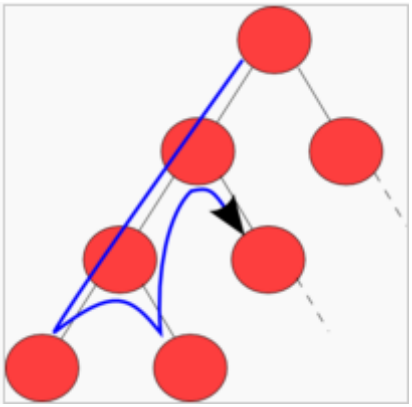


図 3.3 深さ優先探索

深さ優先探索の擬似コードは以下の通りである。

```
1   procedure DFS(G,v):
2       label v as explored
3       for all edges e in G.incidentEdges(v) do
4           if edge e is unexplored then
5               w ← G.opposite(v,e)
6               if vertex w is unexplored then
7                   label e as a discovery edge
8                   recursively call DFS(G,w)
9           else
10              label e as a back edge
```

コミュニケーションパスを探索方法は以下の通りである。

定義： 容量のグラフの根ノードは「R」となる。

始点が割り当てられた容量グラフのノードを「Start」とする。

終点が割り当てられた容量グラフのノードを「Goal」とする。

1) 深さ優先探索で R から Start までのパスを探索して、「start」セットとしてを保存する。
2) 深さ優先探索で R から Goal までのパスを探索して、「goal」セットとしてを保存する。
3) そして、「start」セットと「goal」セットの共通ノードを探索する。
4) 共通ノードで「start」セットと「goal」セットを融合して、パスを作成できる。

例：図 3.1 の 9 番ノードから 6 番ノードまでのパスを探索する。

1) DFS(1,9) = {1,2,5,9}
2) DFS(1,6) = {1,2,6}
3) 共通ノードは「2」である。
4) パス = {9,5,2,6}

以上の方針で図 3.4 のような負荷容量図を作成する。

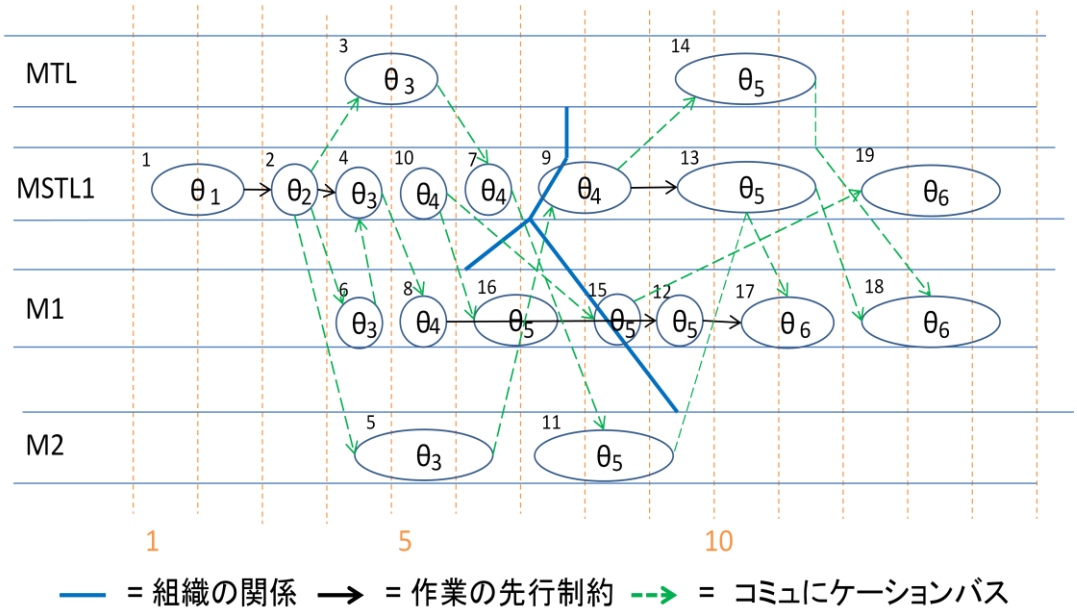


図 3. 4 負荷容量図の例

3.3 ガントチャートの作成方法

ガントチャート作成機能（2.3.17）により、LCB が作成するガントチャートの縦軸には WP 名と割り当てるリソース名を表示する。そして、横軸は日程を表示する。ガントチャートを見やすくするためには、表示する作業（WP）の順番が大切である。以上の考え方を基に、以下のようなガントチャートを作成する方法を考えた。

1. 負荷グラフにより、深さ優先探索の順番で WP のリストを作成する。
2. WP のリストの順番により、ガントチャートの縦軸に WP 名を表示する。
3. WP 名と GLPK のリソース割り当ての出力ファイル（b_{ir}）により、縦軸リソース名を表示する。
4. プロジェクトの開始時刻・WP 名・GLPK の作業開始時刻の出力ファイル（x_{il}）により、タスクバーを作成する。
5. タスクバーの長さは WP の工数から計算する。つまり、タスクバーの始点（start point）は WP の開始時刻から計算する。そして、タスクバーの終点（end point）は WP の開始時刻と工数の合計から計算する。

以上のような作成方法で、図 1.5 の負荷容量図のデータにより、図 3.5 のようなガントチャートを作成する。

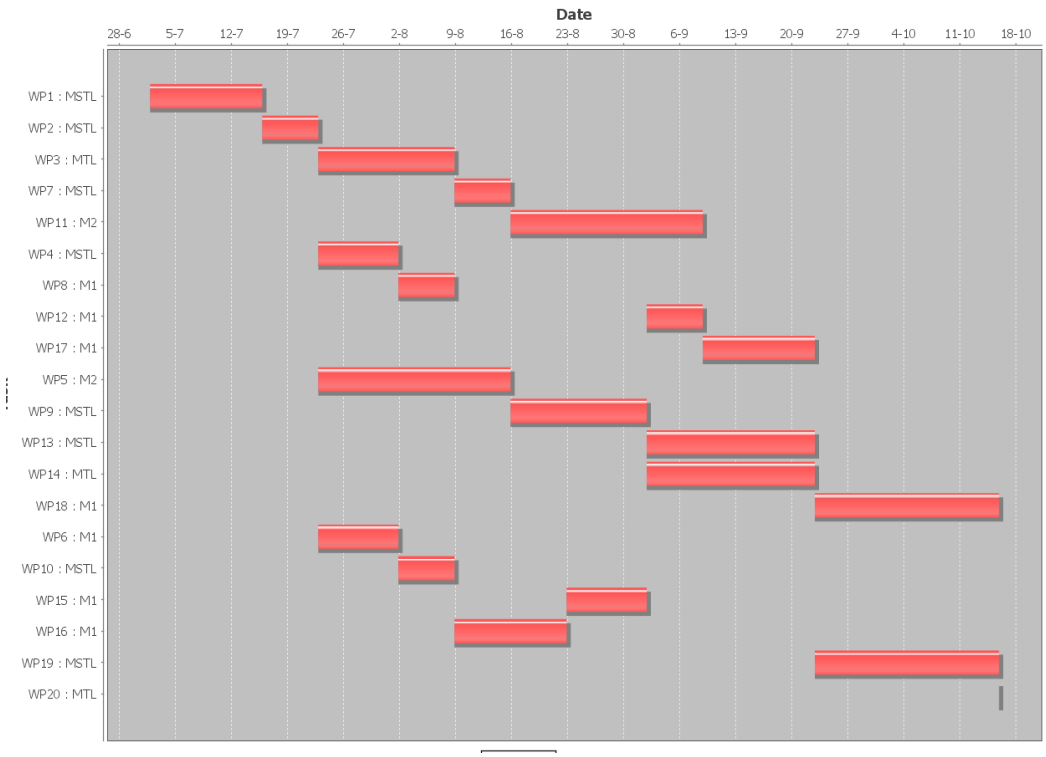


図 3.5 負荷容量図（図 1.5）のデータから作成するガントチャート

第 4 章 実現結果

4.1 LCB プロトタイプの概要

現在の LCB はプロトタイプである。LCB の構成（2.4）により、プロトタイプはツールのサブシステムだけを実現した。プロトタイプではまだデータベースサーバーはない。GUI で負荷グラフを入力する機能と GUI で容量グラフを入力する機能については、ユーザが全て自分で入力することになる。そこで、ユーザの入力支援のため、入力した容量グラフのデータを編集する機能を加えた。また、データベースがないので、ユーザはメイン画面でプロジェクトの開始時刻を入力しなければならない。

また、負荷グラフを追加する機能も開発していないので、プロトタイプでは 2 つの負荷グラフを入力できる。

開発したプロトタイプが持つ機能は図 4.1 のユースケースモデルに示す。図 2.1 に比べて、プロトタイプのユースケースモデルは三つのユースケース、組織のデータベースを照会する・WBS のデータベースを照会する・負荷グラフを追加するユースケースはない。そして、入力した容量データを編集するユースケースが追加される。

4.2 LCB プロトタイプの構成

この LCB のプロトタイプは Java の IDE、NetBeans 7.0 で開発した。そして、Java で GLPK を操作するために、Java ILP を利用する。

さらに、ガントチャートを作成するために、Java のライブラリ 、JFreeChart [10]を利用した。開発したプロトタイプの構成は図 4.2 ようである。図 2.3 に比べて、プロトタイプの構成は Database Server がない。そして、JFreeChart のコンポーネントが追加される。

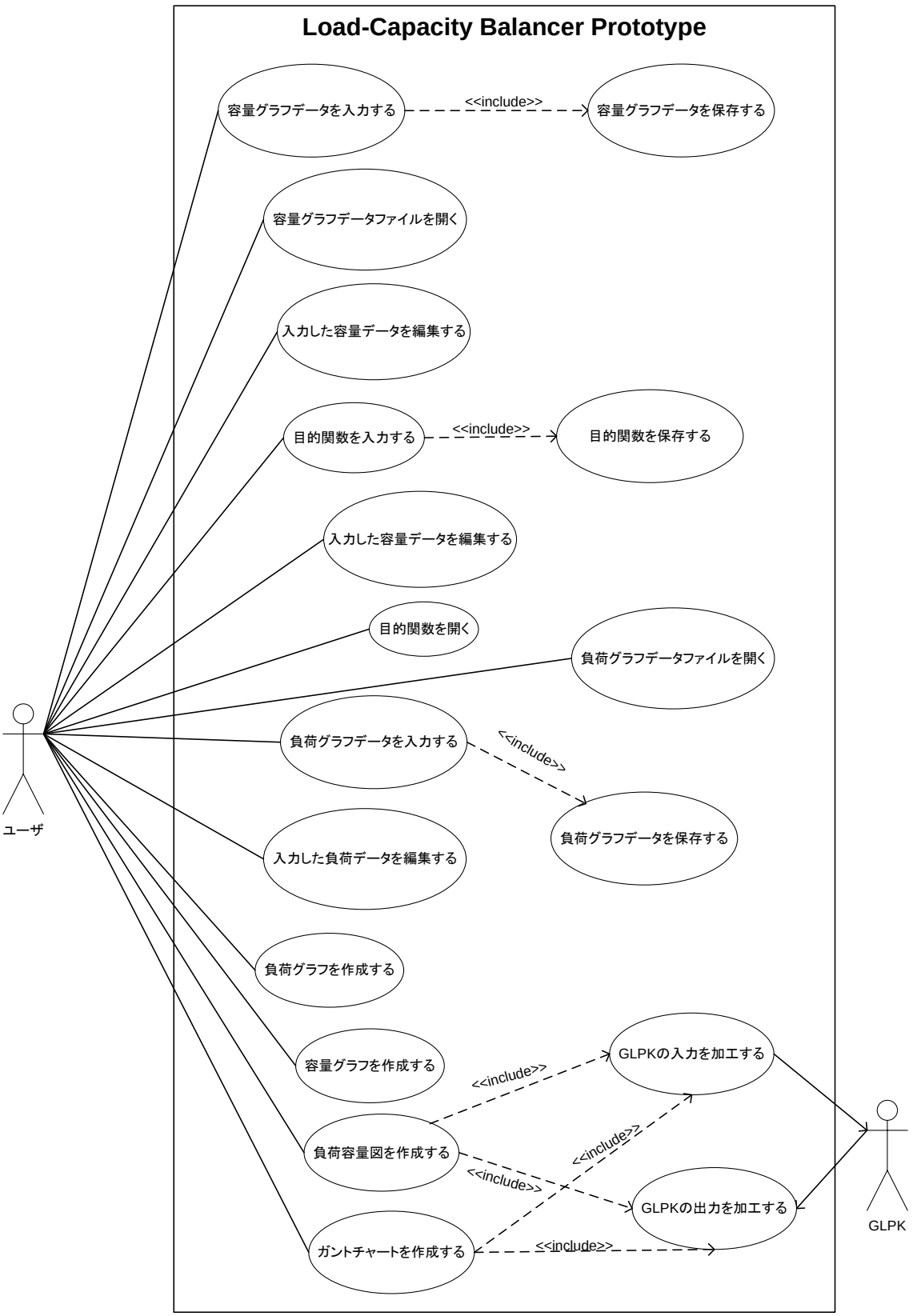


図 4. 1 Load-Capacity Balancer Prototype のユースケース

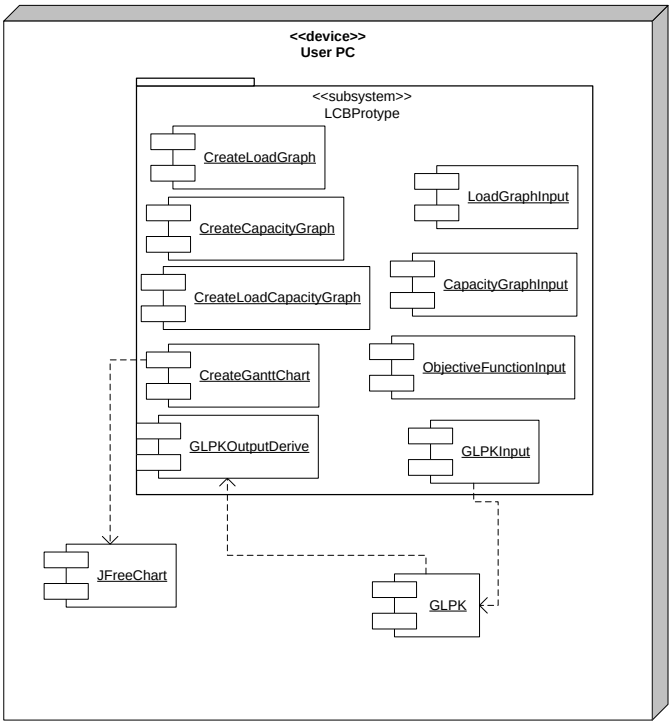


図 4. 2 LCB のプロトタイプの構成

4.3 LCB プロトタイプの機能

開発したプロトタイプは設計通り、5 つの入力画面と 4 つの出力画面から成る。画面遷移は設計（2.7 全体的の GUI の状態遷移）通りである。しかし、このプロトタイプではデータベースのがないので、メイン画面・負荷グラフ入力画面と容量グラフ入力画面が設計内容と異なる。他の画面の機能は設計内容通り開発した。各画面の概要と機能は 4.3.1 -4.3.11 で述べる。

4.3.1 メイン画面

メイン画面は画面の設計通り開発した。しかし、負荷グラフを追加する機能は開発しなかったので、負荷グラフのボタンがない。そして、現在は 2 つの負荷グラフを入力できる仕様となっているので、負荷グラフの入力項目が 2 つある。また、データベースを開発しなかったので、ユーザがプロジェクトの開始時刻を入力しなければならないという制限がある。開発したメインを画面は図 4.3 に示す。

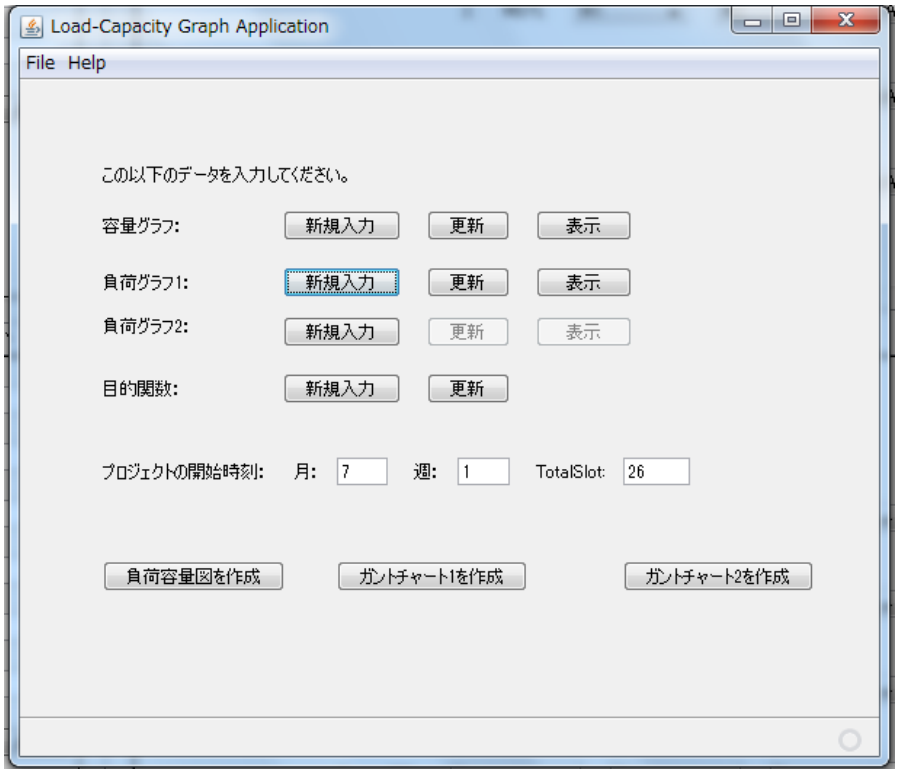


図 4. 3 LCB プロトタイプのメイン画面

メイン画面では以下の操作を行うことができる。

- 容量グラフを新規入力する機能を起動する

  ユーザは容量グラフの「新規入力」ボタンを押すと、ツールは新たな容量グラフ入力画面を表示する。

- 入力した負荷グラフデータを編集機能を起動する

  ユーザは負荷グラフの「更新」ボタンを押すと、ツールは入力した「容量グラフ入力画面」を表示する。

- 容量グラフを作成する機能を起動する

  ユーザは容量グラフの「表示」ボタンを押すと、ツールは容量グラフ画面を起動する。

- 負荷グラフを新規入力する機能を起動する

  ユーザは負荷グラフの「新規入力」ボタンを押すと、ツールは新たな負荷グラフ入力画面を表示する。

- 入力した負荷グラフデータを編集する機能を起動する

  ユーザは負荷グラフの「更新」ボタンを押すと、ツールは入力した「負荷グラフ入力画面」を表示する。

- 負荷グラフを作成する機能を起動する

  ユーザは負荷グラフの「表示」ボタンを押すと、ツールは負荷グラフ画面を起動する。
- 目的関数を入力する機能を起動する

  ユーザは「目的関数入力」のボタンを押すと、ツールは目的関数入力画面（2.5.4）を表示する。
- 入力した目的関数を編集する機能を起動する

  ユーザは目的関数の「エディト」のボタンを押すと、ツールは入力した「入力した目的関　　数入力画面」が表示する。
- プロジェクトの開始時刻を入力する

  データベースを開発しなかったので、ユーザはプロジェクトの開始時刻、何月何週を入力する。また、プロジェクトの開発期間を入力する。
- 負荷容量図作成機能を起動する

  「負荷容量図を作成」のボタンを押すと、ツールは負荷容量図画面を起動する。
- ガントチャート作成

  「ガントチャートを作成」のボタンを押すと、ツールはガントチャート画面を起動する。

4.3.2 容量グラフ入力画面

プロトタイプではデータベースがないので、ユーザは、自分で全てデータを入力しなければならない。これは、負荷グラフ入力画面がの設計とは全く異なる。ユーザは自分で、全てデータを入力しなければならない。開発した負荷グラフ入力画面を図 4.4 に示す。この画面には、4 つのパネルがあり、リソース入力パネル（左上）・組織の関係の入力パネル（右上）・スキル名の入力パネル（左下）・リソースの保持スキルの入力パネル（右下）である。ユーザはこの順に入力する。

容量グラフ入力画面の重要な機能は、ユーザの入力したデータから量グラフのアレーデータを作成する機能である。

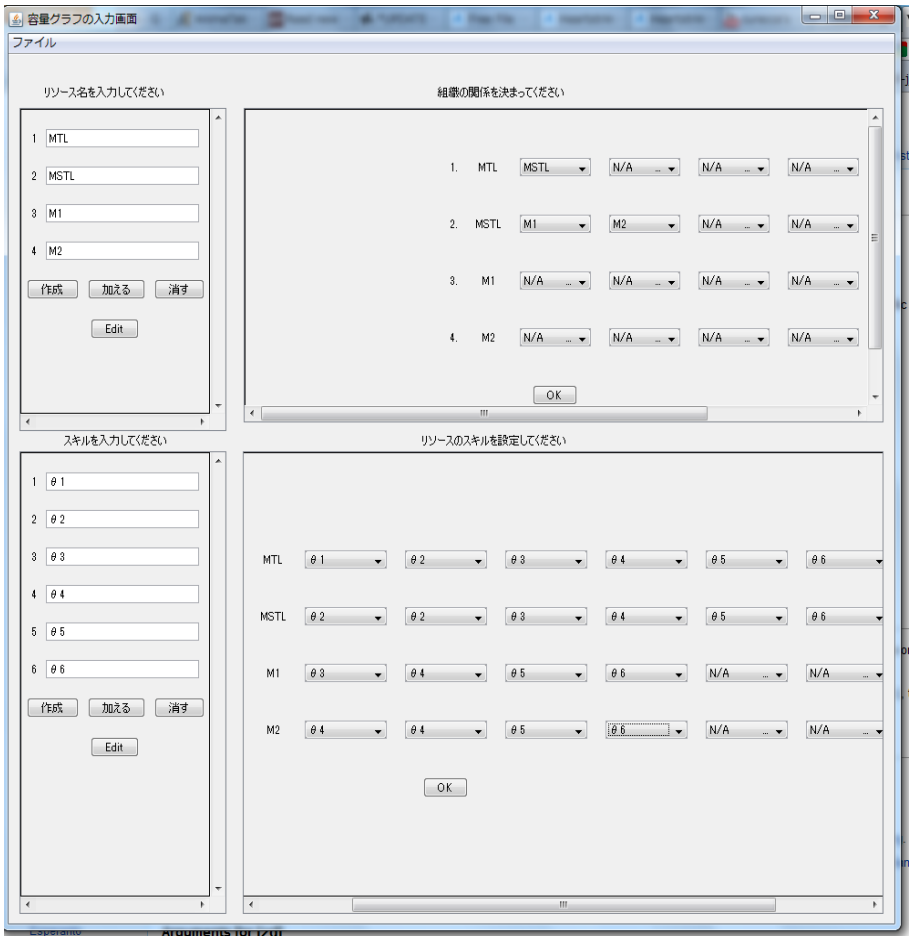


図 4. 4 LCB プロトタイプの容量グラフ入力画面

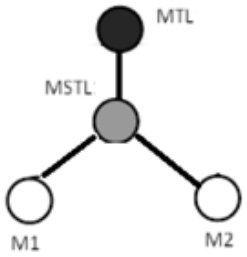


図 4. 5 容量グラフの入力例

	要求仕様差分まとめ θ_1	計画立案 θ_2	差分設計 θ_3	ピアレビュー θ_4	実装・単体テスト θ_5	組合せ/統合テスト θ_6
MTL1	○	○	○	○	○	○
MSTL1	○	○	○	○	○	○
担当 M1-M2			○	○	○	○

表 4. 1 保持スキルの例

例えば、図 4.5 のような容量グラフと表 4.1 のような保持スキル表データを入力すると、以下に示す結果 1 のようなアレーを作成する。

結果 1：

```
resource[ ] = { MTL,MSTL,M1,M2}
edgeParent[ ] = { 1,2,2}
edgeChild[ ] = { 2,3,4}
skills [ ]= {θ1,θ2,θ3,θ4,θ5,θ6}
skillVector[ ] [ ] = {      {1,1,1,1,1,1},
                            {1,1,1,1,1,1},
                            {0,0,1,1,1,1},
                            {0,0,1,1,1,1}     }
```

定義：skillVector[i] [j] ϵ {1,0} ; i = index of resource[] , j = index of skill[ ]

4.3.3 負荷グラフ入力画面

プロトタイプは、データベースがないので、プロトタイプの負荷グラフ入力画面は設計と少し異なる。設計（図 2.6）と違う部分はデータベースから照会するための入力部分がである。ユーザは自分で、全てデータを入力しなければならない。WP のデータ入力パネルと先行制約パネルは設計結果に準じている。

データベースがないので、必要スキルの入力ドロップダウンコンボボックスを作成するために、容量グラフのスキル名入力データから取らなければならない。従って、負荷グラフを入力する前に、容量グラフを入力しなければならない。これも設計結果と異なることである。

開発した負荷グラフ入力画面を図 4.6 に示す。この画面には 2 つのパネルがある。

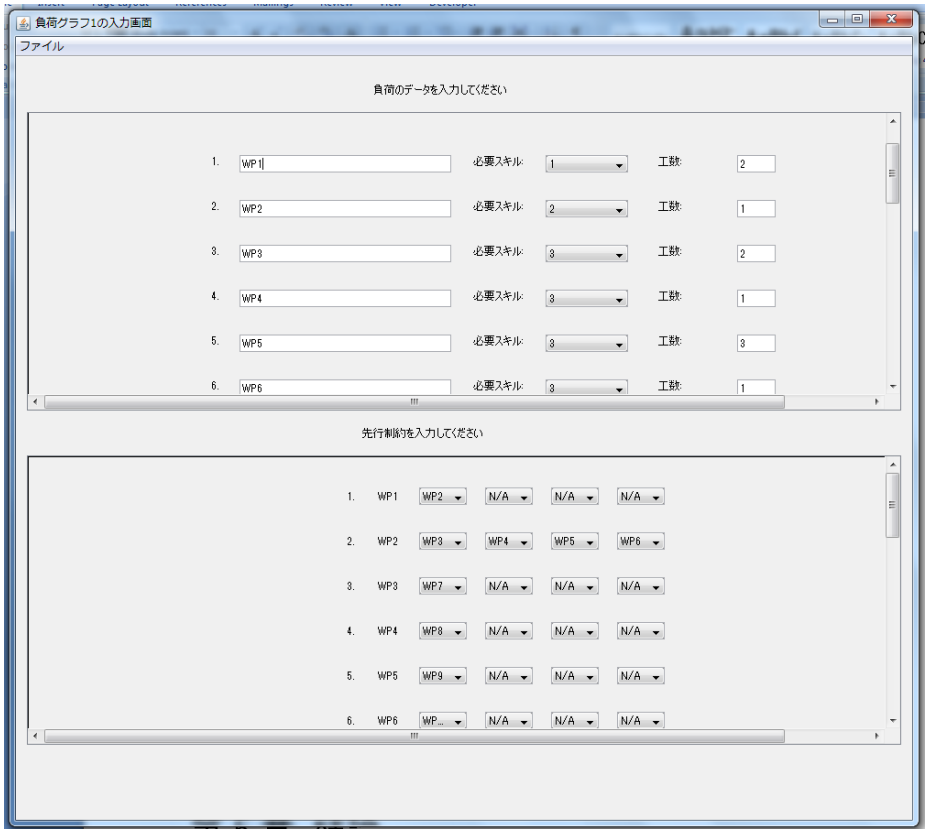


図 4. 6 LCB プロトタイプの負荷グラフ入力画面

- WP データの入力パネル（上）
  このパネルは負荷グラフの頂点のデータを入力するパネルである。このパネルで入力するデータは WP 名・必要スキル・工数である。WP 名と工数はテキストボックスで入力する。必要スキルはドロップダウンコンボボックスで入力する。
- 先行制約入力パネル（下）
  このパネルは負荷グラフの先行制約データを入力するパネルである。入力方法は、始点のラベルで表示し、終点をドロップダウンコンボボックスの中から選択する。

負荷グラフ入力画面の重要な機能はユーザが入力したデータから負荷グラフのアレーデータを作成する機能である。例えば、図 4.7 のような負荷グラフを入力すると、結果 2 のようなアレーを作成する。

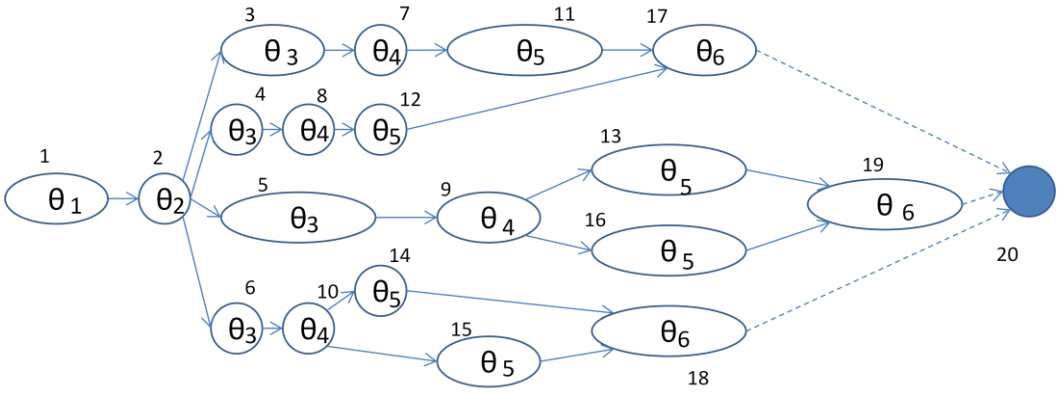


図 4. 7 負荷グラフの入力例

結果 2：

WP[ ] = {WP1,WP2,WP3,WP4,WP5,WP6,WP7,WP8,WP9,
WP10,WP11,WP12,WP13,WP14,WP15,WP16,WP17,WP18,WP19,WP20}
effort[ ] = {2,1,2,1,3,1,1,1,2,1,3,1,3,1,2,3,2,3,3,0}
needSkill [ ] = {1,2,3,3,3,3,4,4,4,4,5,5,5,5,5,5,6,6,6,6}
loadEdgeParent [ ] = {1,2,2,2,2,3,4,5,6 ,7 ,8 ,9 ,9 ,10,10,11,12,13,14,15,16,17,18,19}
loadEdgeChild [ ] = {2,3,4,5,6,7,8,9,10,11,12,13,16,15,16,17,17,19,18,18,19,20,20,20}

4.3.4 目的関数入力画面

プロトタイプの目的関数入力画面は機能定義（2.3.11）と画面設計（2.5.4）の通りに開発した。目的関数入力画面では入力した容量グラフからリソース割り当て目標の表を作成するので、容量グラフのデータを入力した後、操作できる。開発した目的関数入力画面を図 4.8 のに示す。図 4.8 のようにを入力すると、結果 3 のようなアレーを作成する。

図 4. 8 LCB プロトタイプの負荷グラフ入力画面

結果 3：

```
assign[ ] = { 0,1,1,1}  //なるべく MTL に割り当てない
keyPerson[ ] = {2,2}
pairMember[ ] = {3,4}
```

4.3.5 保存・開く画面

保存・開く画面は JAVA のライブラリ、JFileChooser を利用した。保存・開く画面は容量グラフ入力画面・負荷グラフ入力画面・目的関数のファイルメニューバーから利用する。開発した保存・開く画面を図 4.9 に示す。

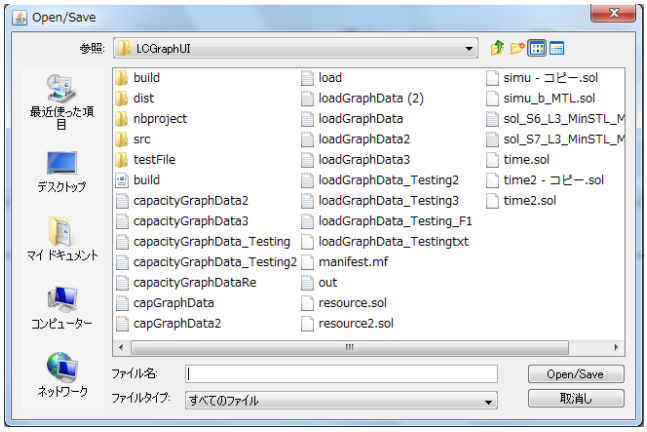


図 4. 9 LCB プロトタイプの保存・開く画面

4.3.6 容量グラフ画面

LCB のプロトタイプの容量グラフ画面は、設計結果と容量グラフの作成方法 (3.1) に通り、開発した。この画面は、入力された容量データから作成した容量グラフのアレーデータ（結果 1）を用いて、容量グラフと保持スキル表を作成し、表示する。開発した容量グラフ画面には 2 つのパネルがあり、容量グラフを表示するパネルと保持スキル表を表示するパネルである。容量グラフが resource[ ]・edgeParent[ ]・edgeChild[ ]から作成する。そして、保持スキル情報をまとめる表が resource[ ]・skills [ ]・ skillVector[ ] [ ]から作成する。

結果 1 から作成した容量グラフ画面を図 4.10 に示す。

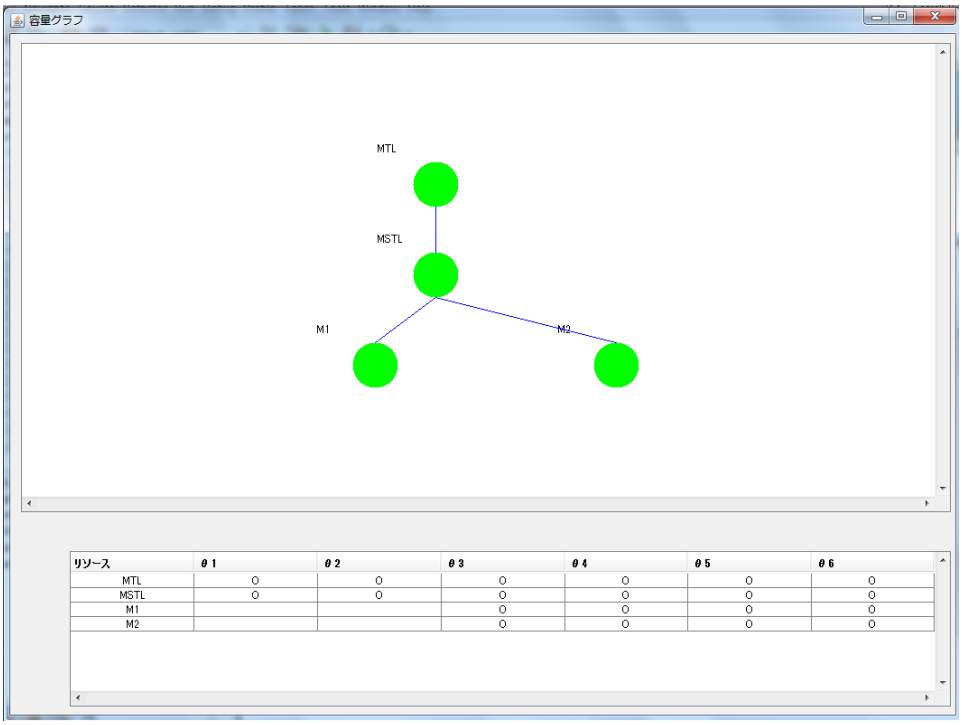


図 4. 10 LCB のプロトタイプの容量グラフ画面

4.3.7 負荷グラフ画面

プロトタイプの負荷グラフ画面は設計結果と負荷グラフの作成方法 (3.1)に通り、開発した。この画面は入力された負荷データから作成した負荷グラフのアレーデータ（結果 2）により、負荷グラフを作成し、表示する。また、作成した負荷グラフの各頂点付近に属性情報 (必要スキルと工数) を表示する。

結果 2 から作成する負荷グラフ画面を図 4.11 に示す。

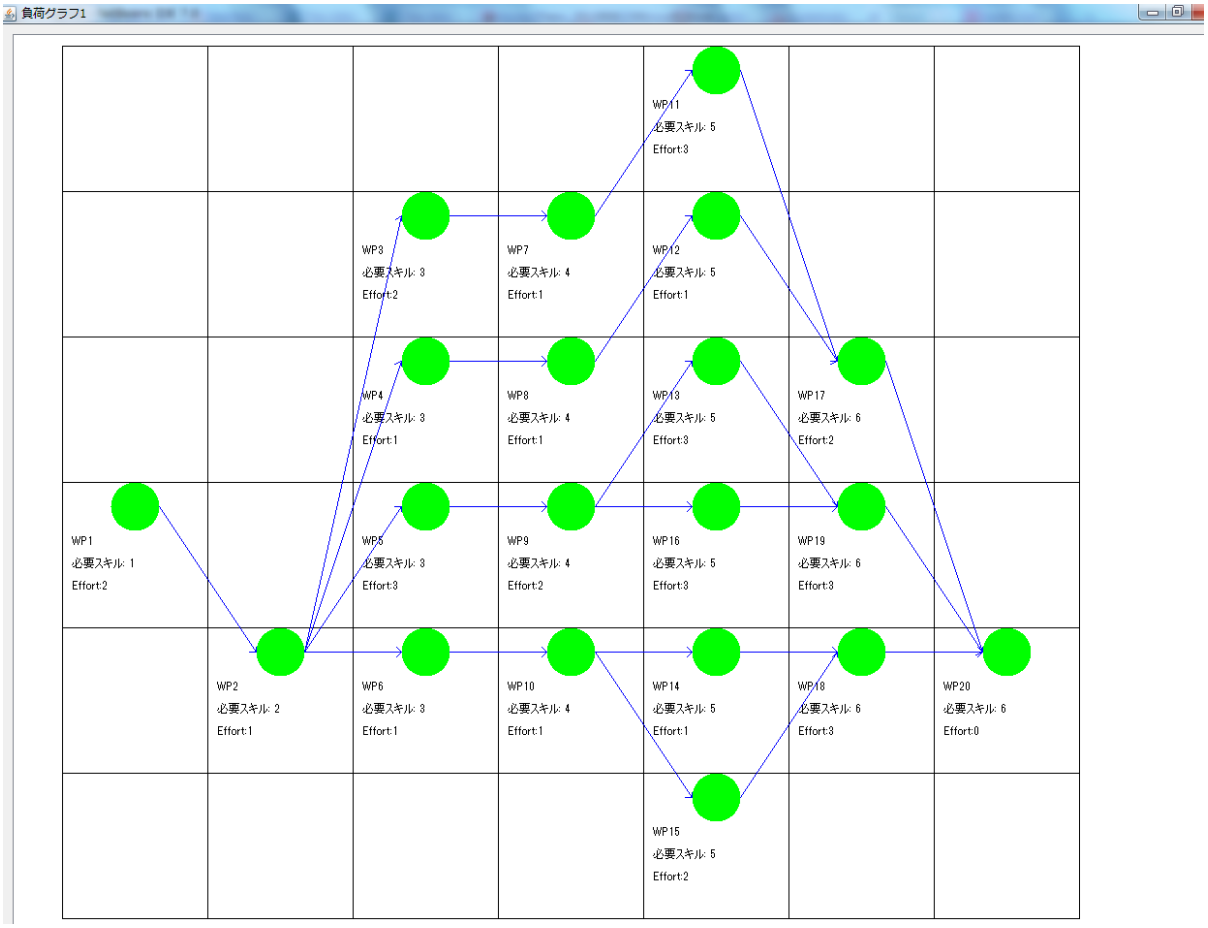


図 4. 11 LCB のプロトタイプの負荷グラフ画面

4.3.8 GLPKInput クラス

GLPKInput クラスは機能定義 (2.3.15) とクラス図設計 結果(2.8) の通り開発した。このクラスは、入力したデータから作成したアレー（結果 1・結果 2・結果 3）から制約式と目的関数を作成する。そして、準備しておく決定変数と共に GLPK に入力する。例えば、結果 1・結果 2・結果 3 から作成する制約式と目的関数は結果 4 の通りである。

結果 4：

- 制約式

1. $\sum_{l=1}^{L+1} x_{il} = 1, \quad i = 1, 2, \cdots, 20$
2. $\sum_{l=1}^{L+1} l \cdot x_{il} \geqq \sum_{l=1}^{L+1} l \cdot x_{jl} + h_j, \quad i, j = 1, 2, \cdots, 20 \quad : (v_j, v_i) \in G(E)$
3. $\sum_{i:S_{req}(v_i)=\theta_s} \sum_{m=\alpha}^{l} x_{im} \leqq N^l(\theta_s), \quad s = 1, 2, \cdots, skl, \quad l = 1, 2, \cdots, L+1$
4. $\sum_{i=1}^{n} \sum_{m=\alpha}^{l} x_{im} \leqq N^l, \ l = 1, 2, \cdots, L+1$
5. $\sum_{r:S_{ret}(r)=\theta_s} b_{ir} = 1, \forall i : S_{req}(v_i) = \theta_s \ , \quad s = 1, 2, \cdots, skl$
6. $\sum_{i:S_{req}(v_i)=\theta_s} b_{ir} \sum_{m=\alpha}^{l} x_{im} \leqq f(r, m), \ l = 1, 2, \cdots, L+1, \quad r = 1, 2, \cdots, N(\theta_s)$

- 目的関数

1. $\left(\sum_{i=1}^{n} t_i =\right) \sum_{i=1}^{n} \sum_{l=1}^{L+1} l \cdot x_{il}$
2. $\min \sum_{i=1}^{n} b_i \, MTL1$

4.3.9 GLPKOutputDerive クラス

GLPKOutputDerive クラス ha 機能定義 (2.3.16) とクラス図設計 (2.8) の通り開発した。このクラス ha2 つの GLPK の出力、.sol ファイルから WP の開始時刻のアレーとリソース割り当てのアレーを作成する。例えば、結果 4 を GLPK に入力し、このクラスで GLPK の出力（Appendix A1）を加工すると、結果 5 のようなアレーが作成される。

結果 5：

resourceBind[ ] = {1,1,0,1,3,2,1,2,1,1,3,2,1,0,2,2,2,2,1,0}

startTime [ ] = { 1,3,4,4,4,4,6,5,7,5,7,9,9,9,8,6,10,12,12,15}

定義：

- resoureBind[i] = a ; i = index of WP[ ] , a = index of resource[] + 1
- startTime [i] = a ; i = index of WP[ ] , a = number of time slot

4.3.10 負荷容量図画面

負荷容量図画面は設計結果と負荷容量図の作成方法(3.2)の通り開発した。この画面は、入力したデータから作成した負荷グラフ・容量グラフのアレーデータ（結果 1・結果 2）とメイン画面入力したプロジェクト開始時刻(何月何週)と GLPK 出力（付録 A）から作成した WP の開始時刻のアレーとリソース割り当てのアレー(結果 5 のような)から負荷容量図を作成する。

結果 1・結果 2・結果 5 から作成した負荷容量図画面を図 4.9 に示す。

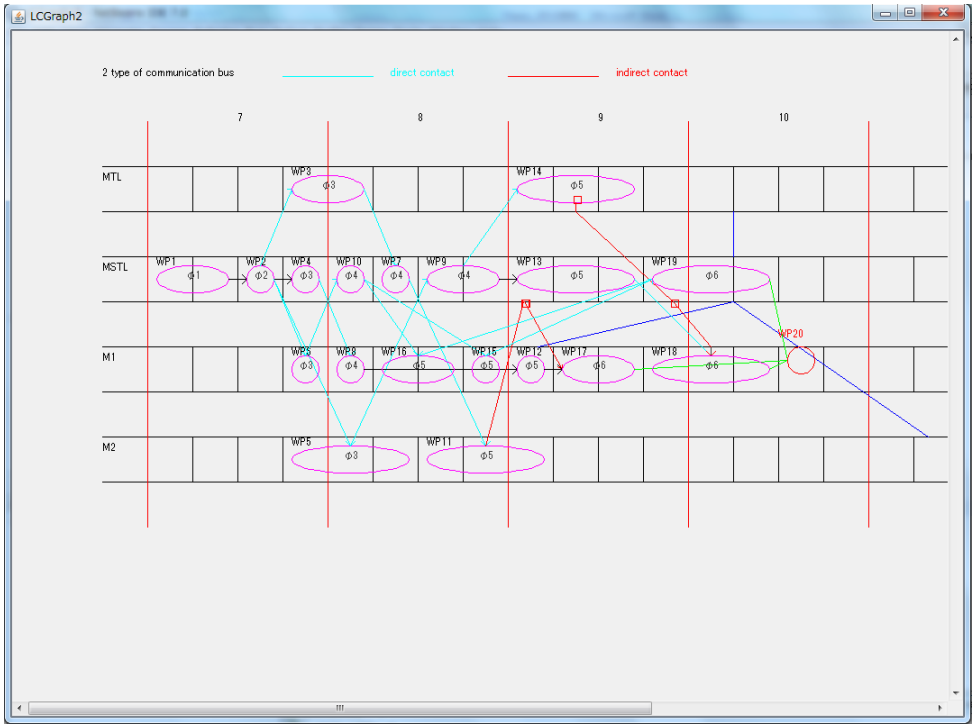


図 4. 12 LCB プロトタイプの負荷容量図画面

4.3.11 ガントチャート画面

プロトタイプのガントチャート画面は機能定義と画面設計結果の通り開発した。ガントチャート画面は、JAVA のライブラリ、JFreeChart1.0.13[10]を利用して開発された。この画面を管理するクラス、役目は、入力したグラフのデータから作成したアレーデータ（結果 1 と結果 2）と GLPK の出力から作成したアレー(結果 5)から一つ一つの JFreeChart の入力データを作成して、DFS の順番に従い、JFreeChart に入力する。詳しい作成方法は第 3 章のガントチャートの作成方法(3.4)に記載する。

例えば、結果 1・結果 2・結果 5 から作成したガントチャート画面を図 4.13 に示す。

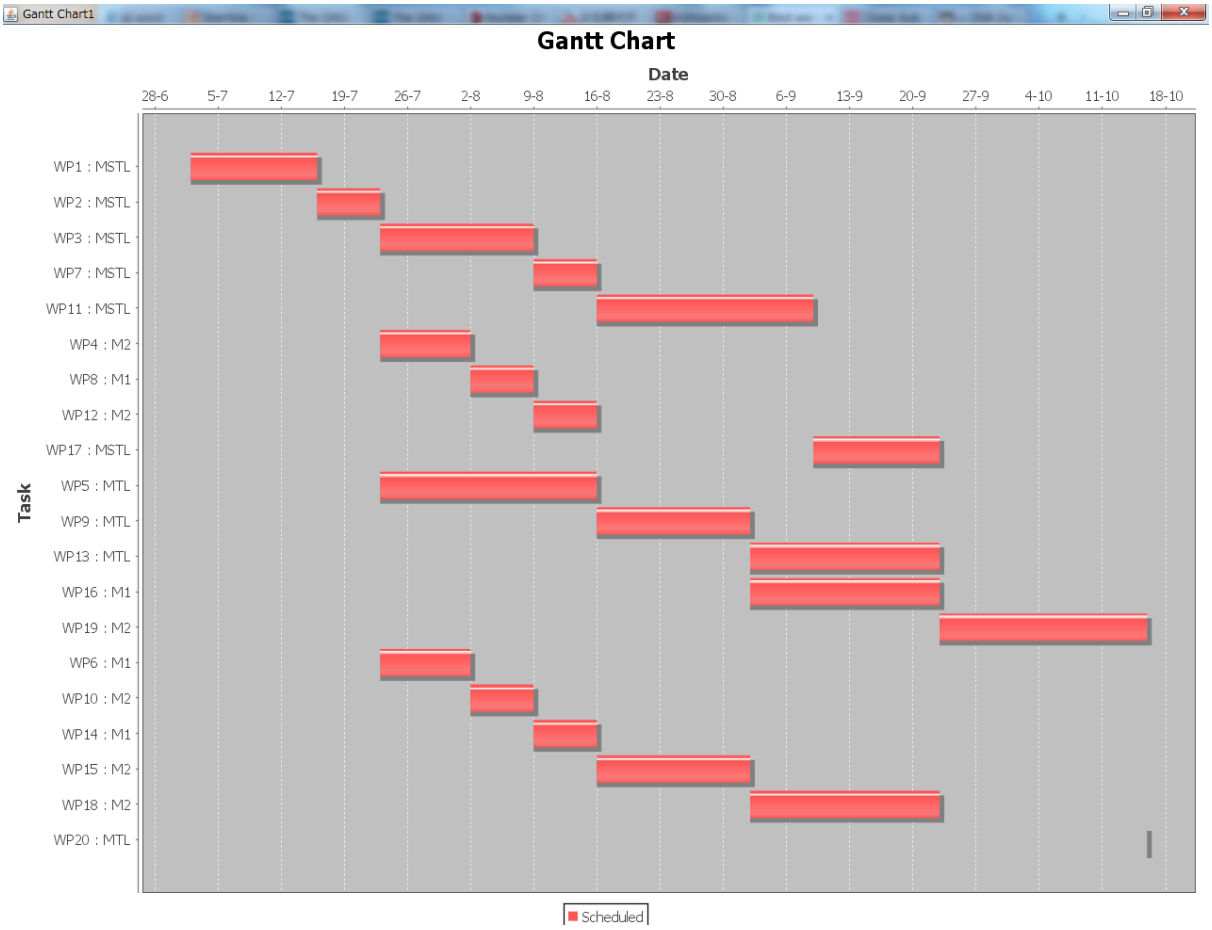


図 4. 13 LCB プロトタイプのガントチャート画面

第 5 章 むすび

5.1 本研究のまとめ

本研究は、負荷容量参照モデルに基づくプロジェクトスケジューリング法を支援するツール、Load-Capacity Balancer (LCB) の機能定義と設計を提案し、その一実装例としてプロトタイプを開発した。提案した設計結果は、ユーザの仕事を最小化するために、負荷グラフ・容量グラフのデータ入力作業をデータベースからデータを照会して選択入力する方針であった。しかし、開発した LCB のプロトタイプは、データベースの部分がないので、ユーザは自分でこれ等のデータを入力しなければならない。

開発した LCB のプロトタイプはユーザが入力した負荷グラフ・容量グラフ・目的関数に関するデータを使って、負荷グラフと容量グラフを作成する。さらに、GLPK を利用して、作業の開始時刻とリソース割り当てを行い、負荷容量図とガントチャートを作成する。

5.2 今後の課題

本研究の残り課題は 2 つある。以下の通りである。

- LCB を完成させ、評価する

現在の開発した LCB にはデータベースがない。そこで、データベースを開発して、現在の LCB と繋げることが課題となる。さらに、完成した LCB のテストや、評価も課題となる。

- GUI の改善

開発した LCB の GUI はまだ使いやすいと言えない。例えば、データを編集するきのうは、表示したグラフで編集できれば、もっと使いやすくなる。これで、使いやすさを向上するため、GUI を改善するひとうがある。これも今後の課題である。

謝辞

本研究を行うにあたり、終始適切な御助言を賜り、また丁寧に御指導頂いた北陸先端科学技術大学院大学、情報科学研究科、落水浩一郎教授に心より深く感謝申し上げます。

本研究の審査員として、大変有益な御意見と御助言を頂きました北陸先端科学技術大学院大学、情報科学研究科、鈴木正人准教授と青木利晃准教授に感謝の意を表します。

共同研究者である、北陸先端科学技術大学院大学情報科学研究科、博士後期課程の艸薙匠様と齋藤彰儀様には、調査のあり方や考察の方法など、細部にわたるご指導をいただきました。ここに感謝いたします。

最後に学業と生活を支えて頂きました家族、友人に本当にありがとうございました。

皆様、本当にありがとうございました。

参考文献

[1] 艸薙匠，落水浩一郎，“プロジェクトの実現可能性を可視化する負荷容量図の提案”，第 31 回 SEA ソフトウェア・シンポジウム 2011，SEA，2011

[2] 艸薙匠，齋藤彰儀，落水浩一郎，“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”，情報処理学会ソフトウェア工学研究会，SE-170-4，2010

[3] Carl Chatfield, Timothy Johnson, “Microsoft Project 2010 Step by Step”, Microsoft Press, Washington, 2010

[4] 齋藤彰儀，艸薙匠，落水浩一郎，“負荷容量参照モデルに基づくプロジェクトスケジューリング法”，信学技報，SS2010-39，2010.

[5] Akinori Saito, Takumi Kunasagi, Koichiro Ochimizu. “A Similtaneous Project Scheduling and Resource Binding Method Based on the Load-Capacity Model”, Asia-Pacific Software Engineering Conference, apsec2011, 2011

[6] A. L. Rosenberg, L. S. Heath, “Graph Separators, with Applications”, Springer, New York, 2001.

[7] Project Management Institute, "A Guide to the Project Management Body of Knowledge", Project Management Institute, Pennsylvania, 2008.

[8] GLPK ver.4.45.Retrieved August 05, 2011, http://www.gnu.org/software/glpk/

[9] 経済産業省，“2009 年版組込みソフトウェア産業実態調査報告書”， Retrieved August 05, 2011, http://www.meti.go.jp/

[10] JFreeChart.Retrieved August 05, 2011, http://www.jfree.org/jfreechart/

付録 A　GLPK の出力

A.1 GLPK 出力の作業開始時刻 (x_{il})

Problem:　simu

Rows:　185

Columns:　400 (400 integer, 400 binary)

Non-zeros:　3174

Status:　INTEGER OPTIMAL

Objective:　Z = 133 (MINimum)

No.	Row name	Activity	Lower bound	Upper bound
1	Z	133		
2	Unique_1	1	1	=
3	Unique_2	1	1	=
4	Unique_3	1	1	=
5	Unique_4	1	1	=
6	Unique_5	1	1	=
7	Unique_6	1	1	=
8	Unique_7	1	1	=
9	Unique_8	1	1	=
10	Unique_9	1	1	=
11	Unique_10	1	1	=
12	Unique_11	1	1	=
13	Unique_12	1	1	=
14	Unique_13	1	1	=
15	Unique_14	1	1	=
16	Unique_15	1	1	=
17	Unique_16	1	1	=
18	Unique_17	1	1	=
19	Unique_18	1	1	=
20	Unique_19	1	1	=
21	Unique_20	1	1	=
22	SR_1	2	2	
23	SR_2	1	1	
24	SR_3	1	1	
25	SR_4	1	1	
26	SR_5	1	1	
27	SR_6	2	2	
28	SR_7	1	1	
29	SR_8	3	3	
30	SR_9	1	1	
31	SR_10	1	1	
32	SR_11	2	2	
33	SR_12	2	2	
34	SR_13	1	1	
35	SR_14	3	3	
36	SR_15	1	1	
37	SR_16	2	1	
38	SR_17	3	3	
39	SR_18	2	1	
40	SR_19	6	3	
41	SR_20	4	1	
42	SR_21	3	3	
43	SR_22	3	2	
44	SR_23	5	2	
45	SR_24	3	3	
46	R_t1_L1	1		2
47	R_t1_L2	1		2
48	R_t1_L3	0		2
49	R_t1_L4	0		2
50	R_t1_L5	0		2
51	R_t1_L6	0		2
52	R_t1_L7	0		2
53	R_t1_L8	0		2
54	R_t1_L9	0		2
55	R_t1_L10	0		2
56	R_t1_L11	0		2
57	R_t1_L12	0		2
58	R_t1_L13	0		2
59	R_t1_L14	0		2
60	R_t1_L15	0		2
61	R_t1_L16	0		2
62	R_t1_L17	0		2
63	R_t1_L18	0		2
64	R_t1_L19	0		2
65	R_t1_L20	0		2
66	R_t2_L1	0		2
67	R_t2_L2	0		2
68	R_t2_L3	1		2
69	R_t2_L4	0		2
70	R_t2_L5	0		2
71	R_t2_L6	0		2
72	R_t2_L7	0		2
73	R_t2_L8	0		2
74	R_t2_L9	0		2
75	R_t2_L10	0		2
76	R_t2_L11	0		2
77	R_t2_L12	0		2
78	R_t2_L13	0		2
79	R_t2_L14	0		2
80	R_t2_L15	0		2
81	R_t2_L16	0		2
82	R_t2_L17	0		2
83	R_t2_L18	0		2
84	R_t2_L19	0		2
85	R_t2_L20	0		2
86	R_t3_L1	0		4
87	R_t3_L2	0		4
88	R_t3_L3	0		4
89	R_t3_L4	4		4
90	R_t3_L5	2		4
91	R_t3_L6	1		4
92	R_t3_L7	0		4
93	R_t3_L8	0		4
94	R_t3_L9	0		4
95	R_t3_L10	0		4
96	R_t3_L11	0		4
97	R_t3_L12	0		4
98	R_t3_L13	0		4
99	R_t3_L14	0		4
100	R_t3_L15	0		4
101	R_t3_L16	0		4
102	R_t3_L17	0		4
103	R_t3_L18	0		4
104	R_t3_L19	0		4
105	R_t3_L20	0		4
106	R_t4_L1	0		4
107	R_t4_L2	0		4
108	R_t4_L3	0		4
109	R_t4_L4	0		4
110	R_t4_L5	2		4
111	R_t4_L6	1		4
112	R_t4_L7	1		4
113	R_t4_L8	1		4
114	R_t4_L9	0		4
115	R_t4_L10	0		4
116	R_t4_L11	0		4
117	R_t4_L12	0		4
118	R_t4_L13	0		4
119	R_t4_L14	0		4
120	R_t4_L15	0		4
121	R_t4_L16	0		4
122	R_t4_L17	0		4
123	R_t4_L18	0		4

124	R_t4_L19	0		4
125	R_t4_L20	0		4
126	R_t5_L1	0		4
127	R_t5_L2	0		4
128	R_t5_L3	0		4
129	R_t5_L4	0		4
130	R_t5_L5	0		4
131	R_t5_L6	2		4
132	R_t5_L7	3		4
133	R_t5_L8	2		4
134	R_t5_L9	3		4
135	R_t5_L10	2		4
136	R_t5_L11	1		4
137	R_t5_L12	0		4
138	R_t5_L13	0		4
139	R_t5_L14	0		4
140	R_t5_L15	0		4
141	R_t5_L16	0		4
142	R_t5_L17	0		4
143	R_t5_L18	0		4
144	R_t5_L19	0		4
145	R_t5_L20	0		4
146	R_t6_L1	0		4
147	R_t6_L2	0		4
148	R_t6_L3	0		4
149	R_t6_L4	0		4
150	R_t6_L5	0		4
151	R_t6_L6	0		4
152	R_t6_L7	0		4
153	R_t6_L8	0		4
154	R_t6_L9	1		4
155	R_t6_L10	2		4
156	R_t6_L11	2		4
157	R_t6_L12	1		4
158	R_t6_L13	1		4
159	R_t6_L14	1		4
160	R_t6_L15	0		4
161	R_t6_L16	0		4
162	R_t6_L17	0		4
163	R_t6_L18	0		4
164	R_t6_L19	0		4
165	R_t6_L20	0		4
166	Concur_L1	1		4
167	Concur_L2	1		4
168	Concur_L3	1		4
169	Concur_L4	4		4
170	Concur_L5	4		4
171	Concur_L6	4		4
172	Concur_L7	4		4
173	Concur_L8	3		4
174	Concur_L9	4		4
175	Concur_L10	4		4
176	Concur_L11	3		4
177	Concur_L12	1		4
178	Concur_L13	1		4
179	Concur_L14	1		4
180	Concur_L15	1		4
181	Concur_L16	0		4
182	Concur_L17	0		4
183	Concur_L18	0		4
184	Concur_L19	0		4
185	Concur_L20	0		4

No.	Column name		Activity	Lower bound	Upper bound
1	x1_1	*	1	0	1
2	x2_1	*	0	0	1
3	x3_1	*	0	0	1
4	x4_1	*	0	0	1
5	x5_1	*	0	0	1
6	x6_1	*	0	0	1
7	x7_1	*	0	0	1
8	x8_1	*	0	0	1
9	x9_1	*	0	0	1
10	x10_1	*	0	0	1
11	x11_1	*	0	0	1
12	x12_1	*	0	0	1
13	x13_1	*	0	0	1
14	x14_1	*	0	0	1
15	x15_1	*	0	0	1
16	x16_1	*	0	0	1
17	x17_1	*	0	0	1
18	x18_1	*	0	0	1
19	x19_1	*	0	0	1
20	x20_1	*	0	0	1
21	x1_2	*	0	0	1
22	x2_2	*	0	0	1
23	x3_2	*	0	0	1
24	x4_2	*	0	0	1
25	x5_2	*	0	0	1
26	x6_2	*	0	0	1
27	x7_2	*	0	0	1
28	x8_2	*	0	0	1
29	x9_2	*	0	0	1
30	x10_2	*	0	0	1
31	x11_2	*	0	0	1
32	x12_2	*	0	0	1
33	x13_2	*	0	0	1
34	x14_2	*	0	0	1
35	x15_2	*	0	0	1
36	x16_2	*	0	0	1
37	x17_2	*	0	0	1
38	x18_2	*	0	0	1
39	x19_2	*	0	0	1
40	x20_2	*	0	0	1
41	x1_3	*	0	0	1
42	x2_3	*	1	0	1
43	x3_3	*	0	0	1
44	x4_3	*	0	0	1
45	x5_3	*	0	0	1
46	x6_3	*	0	0	1
47	x7_3	*	0	0	1
48	x8_3	*	0	0	1
49	x9_3	*	0	0	1
50	x10_3	*	0	0	1
51	x11_3	*	0	0	1
52	x12_3	*	0	0	1
53	x13_3	*	0	0	1
54	x14_3	*	0	0	1
55	x15_3	*	0	0	1
56	x16_3	*	0	0	1
57	x17_3	*	0	0	1
58	x18_3	*	0	0	1
59	x19_3	*	0	0	1
60	x20_3	*	0	0	1
61	x1_4	*	0	0	1
62	x2_4	*	0	0	1
63	x3_4	*	1	0	1
64	x4_4	*	1	0	1
65	x5_4	*	1	0	1
66	x6_4	*	1	0	1
67	x7_4	*	0	0	1
68	x8_4	*	0	0	1
69	x9_4	*	0	0	1
70	x10_4	*	0	0	1
71	x11_4	*	0	0	1
72	x12_4	*	0	0	1
73	x13_4	*	0	0	1
74	x14_4	*	0	0	1
75	x15_4	*	0	0	1
76	x16_4	*	0	0	1
77	x17_4	*	0	0	1
78	x18_4	*	0	0	1
79	x19_4	*	0	0	1
80	x20_4	*	0	0	1
81	x1_5	*	0	0	1
82	x2_5	*	0	0	1
83	x3_5	*	0	0	1
84	x4_5	*	0	0	1
85	x5_5	*	0	0	1
86	x6_5	*	0	0	1
87	x7_5	*	0	0	1
88	x8_5	*	1	0	1
89	x9_5	*	0	0	1
90	x10_5	*	1	0	1
91	x11_5	*	0	0	1
92	x12_5	*	0	0	1
93	x13_5	*	0	0	1
94	x14_5	*	0	0	1
95	x15_5	*	0	0	1
96	x16_5	*	0	0	1
97	x17_5	*	0	0	1
98	x18_5	*	0	0	1
99	x19_5	*	0	0	1
100	x20_5	*	0	0	1
101	x1_6	*	0	0	1
102	x2_6	*	0	0	1
103	x3_6	*	0	0	1
104	x4_6	*	0	0	1
105	x5_6	*	0	0	1
106	x6_6	*	0	0	1
107	x7_6	*	1	0	1
108	x8_6	*	0	0	1
109	x9_6	*	0	0	1
110	x10_6	*	0	0	1
111	x11_6	*	0	0	1
112	x12_6	*	0	0	1
113	x13_6	*	0	0	1

114	x14_6	*	0	0	1	203	x3_11	*	0	0	1
115	x15_6	*	0	0	1	204	x4_11	*	0	0	1
116	x16_6	*	1	0	1	205	x5_11	*	0	0	1
117	x17_6	*	0	0	1	206	x6_11	*	0	0	1
118	x18_6	*	0	0	1	207	x7_11	*	0	0	1
119	x19_6	*	0	0	1	208	x8_11	*	0	0	1
120	x20_6	*	0	0	1	209	x9_11	*	0	0	1
121	x1_7	*	0	0	1	210	x10_11	*	0	0	1
122	x2_7	*	0	0	1	211	x11_11	*	0	0	1
123	x3_7	*	0	0	1	212	x12_11	*	0	0	1
124	x4_7	*	0	0	1	213	x13_11	*	0	0	1
125	x5_7	*	0	0	1	214	x14_11	*	0	0	1
126	x6_7	*	0	0	1	215	x15_11	*	0	0	1
127	x7_7	*	0	0	1	216	x16_11	*	0	0	1
128	x8_7	*	0	0	1	217	x17_11	*	0	0	1
129	x9_7	*	1	0	1	218	x18_11	*	0	0	1
130	x10_7	*	0	0	1	219	x19_11	*	0	0	1
131	x11_7	*	1	0	1	220	x20_11	*	0	0	1
132	x12_7	*	0	0	1	221	x1_12	*	0	0	1
133	x13_7	*	0	0	1	222	x2_12	*	0	0	1
134	x14_7	*	0	0	1	223	x3_12	*	0	0	1
135	x15_7	*	0	0	1	224	x4_12	*	0	0	1
136	x16_7	*	0	0	1	225	x5_12	*	0	0	1
137	x17_7	*	0	0	1	226	x6_12	*	0	0	1
138	x18_7	*	0	0	1	227	x7_12	*	0	0	1
139	x19_7	*	0	0	1	228	x8_12	*	0	0	1
140	x20_7	*	0	0	1	229	x9_12	*	0	0	1
141	x1_8	*	0	0	1	230	x10_12	*	0	0	1
142	x2_8	*	0	0	1	231	x11_12	*	0	0	1
143	x3_8	*	0	0	1	232	x12_12	*	0	0	1
144	x4_8	*	0	0	1	233	x13_12	*	0	0	1
145	x5_8	*	0	0	1	234	x14_12	*	0	0	1
146	x6_8	*	0	0	1	235	x15_12	*	0	0	1
147	x7_8	*	0	0	1	236	x16_12	*	0	0	1
148	x8_8	*	0	0	1	237	x17_12	*	0	0	1
149	x9_8	*	0	0	1	238	x18_12	*	1	0	1
150	x10_8	*	0	0	1	239	x19_12	*	1	0	1
151	x11_8	*	0	0	1	240	x20_12	*	0	0	1
152	x12_8	*	0	0	1	241	x1_13	*	0	0	1
153	x13_8	*	0	0	1	242	x2_13	*	0	0	1
154	x14_8	*	0	0	1	243	x3_13	*	0	0	1
155	x15_8	*	1	0	1	244	x4_13	*	0	0	1
156	x16_8	*	0	0	1	245	x5_13	*	0	0	1
157	x17_8	*	0	0	1	246	x6_13	*	0	0	1
158	x18_8	*	0	0	1	247	x7_13	*	0	0	1
159	x19_8	*	0	0	1	248	x8_13	*	0	0	1
160	x20_8	*	0	0	1	249	x9_13	*	0	0	1
161	x1_9	*	0	0	1	250	x10_13	*	0	0	1
162	x2_9	*	0	0	1	251	x11_13	*	0	0	1
163	x3_9	*	0	0	1	252	x12_13	*	0	0	1
164	x4_9	*	0	0	1	253	x13_13	*	0	0	1
165	x5_9	*	0	0	1	254	x14_13	*	0	0	1
166	x6_9	*	0	0	1	255	x15_13	*	0	0	1
167	x7_9	*	0	0	1	256	x16_13	*	0	0	1
168	x8_9	*	0	0	1	257	x17_13	*	0	0	1
169	x9_9	*	0	0	1	258	x18_13	*	0	0	1
170	x10_9	*	0	0	1	259	x19_13	*	0	0	1
171	x11_9	*	0	0	1	260	x20_13	*	0	0	1
172	x12_9	*	1	0	1	261	x1_14	*	0	0	1
173	x13_9	*	1	0	1	262	x2_14	*	0	0	1
174	x14_9	*	1	0	1	263	x3_14	*	0	0	1
175	x15_9	*	0	0	1	264	x4_14	*	0	0	1
176	x16_9	*	0	0	1	265	x5_14	*	0	0	1
177	x17_9	*	0	0	1	266	x6_14	*	0	0	1
178	x18_9	*	0	0	1	267	x7_14	*	0	0	1
179	x19_9	*	0	0	1	268	x8_14	*	0	0	1
180	x20_9	*	0	0	1	269	x9_14	*	0	0	1
181	x1_10	*	0	0	1	270	x10_14	*	0	0	1
182	x2_10	*	0	0	1	271	x11_14	*	0	0	1
183	x3_10	*	0	0	1	272	x12_14	*	0	0	1
184	x4_10	*	0	0	1	273	x13_14	*	0	0	1
185	x5_10	*	0	0	1	274	x14_14	*	0	0	1
186	x6_10	*	0	0	1	275	x15_14	*	0	0	1
187	x7_10	*	0	0	1	276	x16_14	*	0	0	1
188	x8_10	*	0	0	1	277	x17_14	*	0	0	1
189	x9_10	*	0	0	1	278	x18_14	*	0	0	1
190	x10_10	*	0	0	1	279	x19_14	*	0	0	1
191	x11_10	*	0	0	1	280	x20_14	*	0	0	1
192	x12_10	*	0	0	1	281	x1_15	*	0	0	1
193	x13_10	*	0	0	1	282	x2_15	*	0	0	1
194	x14_10	*	0	0	1	283	x3_15	*	0	0	1
195	x15_10	*	0	0	1	284	x4_15	*	0	0	1
196	x16_10	*	0	0	1	285	x5_15	*	0	0	1
197	x17_10	*	1	0	1	286	x6_15	*	0	0	1
198	x18_10	*	0	0	1	287	x7_15	*	0	0	1
199	x19_10	*	0	0	1	288	x8_15	*	0	0	1
200	x20_10	*	0	0	1	289	x9_15	*	0	0	1
201	x1_11	*	0	0	1	290	x10_15	*	0	0	1
202	x2_11	*	0	0	1	291	x11_15	*	0	0	1

```
292 x12_15      *              0             0             1
293 x13_15      *              0             0             1
294 x14_15      *              0             0             1
295 x15_15      *              0             0             1
296 x16_15      *              0             0             1
297 x17_15      *              0             0             1
298 x18_15      *              0             0             1
299 x19_15      *              0             0             1
300 x20_15      *              1             0             1
301 x1_16       *              0             0             1
302 x2_16       *              0             0             1
303 x3_16       *              0             0             1
304 x4_16       *              0             0             1
305 x5_16       *              0             0             1
306 x6_16       *              0             0             1
307 x7_16       *              0             0             1
308 x8_16       *              0             0             1
309 x9_16       *              0             0             1
310 x10_16      *              0             0             1
311 x11_16      *              0             0             1
312 x12_16      *              0             0             1
313 x13_16      *              0             0             1
314 x14_16      *              0             0             1
315 x15_16      *              0             0             1
316 x16_16      *              0             0             1
317 x17_16      *              0             0             1
318 x18_16      *              0             0             1
319 x19_16      *              0             0             1
320 x20_16      *              0             0             1
321 x1_17       *              0             0             1
322 x2_17       *              0             0             1
323 x3_17       *              0             0             1
324 x4_17       *              0             0             1
325 x5_17       *              0             0             1
326 x6_17       *              0             0             1
327 x7_17       *              0             0             1
328 x8_17       *              0             0             1
329 x9_17       *              0             0             1
330 x10_17      *              0             0             1
331 x11_17      *              0             0             1
332 x12_17      *              0             0             1
333 x13_17      *              0             0             1
334 x14_17      *              0             0             1
335 x15_17      *              0             0             1
336 x16_17      *              0             0             1
337 x17_17      *              0             0             1
338 x18_17      *              0             0             1
339 x19_17      *              0             0             1
340 x20_17      *              0             0             1
341 x1_18       *              0             0             1
342 x2_18       *              0             0             1
343 x3_18       *              0             0             1
344 x4_18       *              0             0             1
345 x5_18       *              0             0             1
346 x6_18       *              0             0             1
347 x7_18       *              0             0             1
348 x8_18       *              0             0             1
349 x9_18       *              0             0             1
350 x10_18      *              0             0             1
351 x11_18      *              0             0             1
352 x12_18      *              0             0             1
353 x13_18      *              0             0             1
354 x14_18      *              0             0             1
355 x15_18      *              0             0             1
356 x16_18      *              0             0             1
357 x17_18      *              0             0             1
358 x18_18      *              0             0             1
359 x19_18      *              0             0             1
360 x20_18      *              0             0             1
361 x1_19       *              0             0             1
362 x2_19       *              0             0             1
363 x3_19       *              0             0             1
364 x4_19       *              0             0             1
365 x5_19       *              0             0             1
366 x6_19       *              0             0             1
367 x7_19       *              0             0             1
368 x8_19       *              0             0             1
369 x9_19       *              0             0             1
370 x10_19      *              0             0             1
371 x11_19      *              0             0             1
372 x12_19      *              0             0             1
373 x13_19      *              0             0             1
374 x14_19      *              0             0             1
375 x15_19      *              0             0             1
376 x16_19      *              0             0             1
377 x17_19      *              0             0             1
378 x18_19      *              0             0             1
379 x19_19      *              0             0             1
380 x20_19      *              0             0             1
381 x1_20       *              0             0             1
382 x2_20       *              0             0             1
383 x3_20       *              0             0             1
384 x4_20       *              0             0             1
385 x5_20       *              0             0             1
386 x6_20       *              0             0             1
387 x7_20       *              0             0             1
388 x8_20       *              0             0             1
389 x9_20       *              0             0             1
390 x10_20      *              0             0             1
391 x11_20      *              0             0             1
392 x12_20      *              0             0             1
393 x13_20      *              0             0             1
394 x14_20      *              0             0             1
395 x15_20      *              0             0             1
396 x16_20      *              0             0             1
397 x17_20      *              0             0             1
398 x18_20      *              0             0             1
399 x19_20      *              0             0             1
400 x20_20      *              0             0             1

Integer feasibility conditions:

KKT.PE: max.abs.err = 0.00e+000 on row 0
        max.rel.err = 0.00e+000 on row 0
        High quality

KKT.PB: max.abs.err = 0.00e+000 on row 0
        max.rel.err = 0.00e+000 on row 0
        High quality

End of output
```

A.2 GLPK 出力のリソース割り当て (b_{ir})

Problem: simu_b

Rows: 140

Columns: 72 (72 integer, 72 binary)

Non-zeros: 229

Status: INTEGER OPTIMAL

Objective: Z = 3 (MINimum)

No.	Row name	Activity	Lower bound	Upper bound	
1	Z	3			
2	Uni_b1	1	1		=
3	Uni_b2	1	1		=
4	Uni_b3	1	1		=
5	Uni_b4	1	1		=
6	Uni_b5	1	1		=
7	Uni_b6	1	1		=
8	Uni_b7	1	1		=
9	Uni_b8	1	1		=
10	Uni_b9	1	1		=
11	Uni_b10	1	1		=
12	Uni_b11	1	1		=
13	Uni_b12	1	1		=
14	Uni_b13	1	1		=
15	Uni_b14	1	1		=
16	Uni_b15	1	1		=
17	Uni_b16	1	1		=
18	Uni_b17	1	1		=

19 Uni_b18	1	1	=			0	1
20 Uni_b19	1	1	=		65 RB_t2_RMTL_L15		
21 RB_t1_RMTL_L1				0	1	66 RB_t2_RMSTL_L1	
	0		1			0	1
22 RB_t1_RMTL_L2				0	1	67 RB_t2_RMSTL_L2	
	0		1			0	1
23 RB_t1_RMTL_L3				0	1	68 RB_t2_RMSTL_L3	
	0		1			1	1
24 RB_t1_RMTL_L4				0	1	69 RB_t2_RMSTL_L4	
	0		1			0	1
25 RB_t1_RMTL_L5				0	1	70 RB_t2_RMSTL_L5	
	0		1			0	1
26 RB_t1_RMTL_L6				0	1	71 RB_t2_RMSTL_L6	
	0		1			0	1
27 RB_t1_RMTL_L7				0	1	72 RB_t2_RMSTL_L7	
	0		1			0	1
28 RB_t1_RMTL_L8				0	1	73 RB_t2_RMSTL_L8	
	0		1			0	1
29 RB_t1_RMTL_L9				0	1	74 RB_t2_RMSTL_L9	
	0		1			0	1
30 RB_t1_RMTL_L10				0	1	75 RB_t2_RMSTL_L10	
	0		1			0	1
31 RB_t1_RMTL_L11				0	1	76 RB_t2_RMSTL_L11	
	0		1			0	1
32 RB_t1_RMTL_L12				0	1	77 RB_t2_RMSTL_L12	
	0		1			0	1
33 RB_t1_RMTL_L13				0	1	78 RB_t2_RMSTL_L13	
	0		1			0	1
34 RB_t1_RMTL_L14				0	1	79 RB_t2_RMSTL_L14	
	0		1			0	1
35 RB_t1_RMTL_L15				0	1	80 RB_t2_RMSTL_L15	
	0		1			0	1
36 RB_t1_RMSTL_L1				1	1	81 RB_t3_6_RMTL_L1	
	1		1			0	1
37 RB_t1_RMSTL_L2				1	1	82 RB_t3_6_RMTL_L2	
	1		1			0	1
38 RB_t1_RMSTL_L3				0	1	83 RB_t3_6_RMTL_L3	
	0		1			0	1
39 RB_t1_RMSTL_L4				0	1	84 RB_t3_6_RMTL_L4	
	0		1			1	1
40 RB_t1_RMSTL_L5				0	1	85 RB_t3_6_RMTL_L5	
	0		1			1	1
41 RB_t1_RMSTL_L6				0	1	86 RB_t3_6_RMTL_L6	
	0		1			1	1
42 RB_t1_RMSTL_L7				0	1	87 RB_t3_6_RMTL_L7	
	0		1			1	1
43 RB_t1_RMSTL_L8				0	1	88 RB_t3_6_RMTL_L8	
	0		1			1	1
44 RB_t1_RMSTL_L9				0	1	89 RB_t3_6_RMTL_L9	
	0		1			1	1
45 RB_t1_RMSTL_L10				0	1	90 RB_t3_6_RMTL_L10	
	0		1			1	1
46 RB_t1_RMSTL_L11				0	1	91 RB_t3_6_RMTL_L11	
	0		1			1	1
47 RB_t1_RMSTL_L12				0	1	92 RB_t3_6_RMTL_L12	
	0		1			0	1
48 RB_t1_RMSTL_L13				0	1	93 RB_t3_6_RMTL_L13	
	0		1			0	1
49 RB_t1_RMSTL_L14				0	1	94 RB_t3_6_RMTL_L14	
	0		1			0	1
50 RB_t1_RMSTL_L15				0	1	95 RB_t3_6_RMTL_L15	
	0		1			0	1
51 RB_t2_RMTL_L1				0	1	96 RB_t3_6_RMSTL_L1	
	0		1			0	1
52 RB_t2_RMTL_L2				0	1	97 RB_t3_6_RMSTL_L2	
	0		1			0	1
53 RB_t2_RMTL_L3				0	1	98 RB_t3_6_RMSTL_L3	
	0		1			0	1
54 RB_t2_RMTL_L4				0	1	99 RB_t3_6_RMSTL_L4	
	0		1			1	1
55 RB_t2_RMTL_L5				0	1	100 RB_t3_6_RMSTL_L5	
	0		1			1	1
56 RB_t2_RMTL_L6				0	1	101 RB_t3_6_RMSTL_L6	
	0		1			1	1
57 RB_t2_RMTL_L7				0	1	102 RB_t3_6_RMSTL_L7	
	0		1			1	1
58 RB_t2_RMTL_L8				0	1	103 RB_t3_6_RMSTL_L8	
	0		1			1	1
59 RB_t2_RMTL_L9				0	1	104 RB_t3_6_RMSTL_L9	
	0		1			1	1
60 RB_t2_RMTL_L10				0	1	105 RB_t3_6_RMSTL_L10	
	0		1			1	1
61 RB_t2_RMTL_L11				0	1	106 RB_t3_6_RMSTL_L11	
	0		1			1	1
62 RB_t2_RMTL_L12				0	1	107 RB_t3_6_RMSTL_L12	
	0		1			0	1
63 RB_t2_RMTL_L13				0	1	108 RB_t3_6_RMSTL_L13	
	0		1			0	1
64 RB_t2_RMTL_L14							

```
109 RB_t3_6_RMSTL_L14
                                0                       1
110 RB_t3_6_RMSTL_L15
                                0                       1
111 RB_t3_6_RM1_L1
                                0                       1
112 RB_t3_6_RM1_L2
                                0                       1
113 RB_t3_6_RM1_L3
                                0                       1
114 RB_t3_6_RM1_L4
                                1                       1
115 RB_t3_6_RM1_L5
                                1                       1
116 RB_t3_6_RM1_L6
                                1                       1
117 RB_t3_6_RM1_L7
                                1                       1
118 RB_t3_6_RM1_L8
                                1                       1
119 RB_t3_6_RM1_L9
                                1                       1
120 RB_t3_6_RM1_L10
                                1                       1
121 RB_t3_6_RM1_L11
                                0                       1
122 RB_t3_6_RM1_L12
                                0                       1
123 RB_t3_6_RM1_L13
                                0                       1
124 RB_t3_6_RM1_L14
                                0                       1
125 RB_t3_6_RM1_L15
                                0                       1
126 RB_t3_6_RM2_L1
                                0                       1
127 RB_t3_6_RM2_L2
                                0                       1
128 RB_t3_6_RM2_L3
                                0                       1
129 RB_t3_6_RM2_L4
                                1                       1
130 RB_t3_6_RM2_L5
                                1                       1
131 RB_t3_6_RM2_L6
                                1                       1
132 RB_t3_6_RM2_L7
                                1                       1
133 RB_t3_6_RM2_L8
                                0                       1
134 RB_t3_6_RM2_L9
                                1                       1
135 RB_t3_6_RM2_L10
                                1                       1
136 RB_t3_6_RM2_L11
                                1                       1
137 RB_t3_6_RM2_L12
                                1                       1
138 RB_t3_6_RM2_L13
                                1                       1
139 RB_t3_6_RM2_L14
                                1                       1
140 RB_t3_6_RM2_L15
                                0                       1
```

No.	Column name		Activity	Lower bound	Upper bound
1	b1_MTL	*	0	0	1
2	b2_MTL	*	0	0	1
3	b3_MTL	*	1	0	1
4	b4_MTL	*	0	0	1
5	b5_MTL	*	0	0	1
6	b6_MTL	*	0	0	1
7	b7_MTL	*	0	0	1
8	b8_MTL	*	0	0	1
9	b9_MTL	*	0	0	1
10	b10_MTL	*	0	0	1
11	b11_MTL	*	0	0	1
12	b12_MTL	*	0	0	1
13	b13_MTL	*	0	0	1
14	b14_MTL	*	1	0	1
15	b15_MTL	*	0	0	1
16	b16_MTL	*	0	0	1
17	b17_MTL	*	0	0	1
18	b18_MTL	*	0	0	1
19	b19_MTL	*	0	0	1
20	b1_MSTL	*	1	0	1
21	b2_MSTL	*	1	0	1
22	b3_MSTL	*	0	0	1
23	b4_MSTL	*	1	0	1
24	b5_MSTL	*	0	0	1
25	b6_MSTL	*	0	0	1
26	b7_MSTL	*	1	0	1
27	b8_MSTL	*	0	0	1
28	b9_MSTL	*	1	0	1
29	b10_MSTL	*	1	0	1
30	b11_MSTL	*	0	0	1
31	b12_MSTL	*	0	0	1
32	b13_MSTL	*	1	0	1
33	b14_MSTL	*	0	0	1
34	b15_MSTL	*	0	0	1
35	b16_MSTL	*	0	0	1
36	b17_MSTL	*	0	0	1
37	b18_MSTL	*	0	0	1
38	b19_MSTL	*	1	0	1
39	b3_M1	*	0	0	1
40	b4_M1	*	0	0	1
41	b5_M1	*	0	0	1
42	b6_M1	*	1	0	1
43	b7_M1	*	0	0	1
44	b8_M1	*	1	0	1
45	b9_M1	*	0	0	1
46	b10_M1	*	0	0	1
47	b11_M1	*	0	0	1
48	b12_M1	*	1	0	1
49	b13_M1	*	0	0	1
50	b14_M1	*	0	0	1
51	b15_M1	*	1	0	1
52	b16_M1	*	1	0	1
53	b17_M1	*	1	0	1
54	b18_M1	*	1	0	1
55	b19_M1	*	0	0	1
56	b3_M2	*	0	0	1
57	b4_M2	*	0	0	1
58	b5_M2	*	1	0	1
59	b6_M2	*	0	0	1
60	b7_M2	*	0	0	1
61	b8_M2	*	0	0	1
62	b9_M2	*	0	0	1
63	b10_M2	*	0	0	1
64	b11_M2	*	1	0	1
65	b12_M2	*	0	0	1
66	b13_M2	*	0	0	1
67	b14_M2	*	0	0	1
68	b15_M2	*	0	0	1
69	b16_M2	*	0	0	1
70	b17_M2	*	0	0	1
71	b18_M2	*	0	0	1
72	b19_M2	*	0	0	1

```
Integer feasibility conditions:

KKT.PE: max.abs.err = 0.00e+000 on row 0
        max.rel.err = 0.00e+000 on row 0
        High quality

KKT.PB: max.abs.err = 0.00e+000 on row 0
        max.rel.err = 0.00e+000 on row 0
        High quality

End of output
```