日本機械学会論文集 (C 編)
65 巻 630 号 (1999-2)

論文 No. 98-0537

システム構造に着目したエージェント方式による分散並行型設計支援システムの構成方法*

藤田 喜久雄†, 赤木 新介†

Agent-Based Distributed Concurrent Design System Architecture Synchronized with System Structure

Kikuo FUJITA and Shinsuke AKAGI

This paper proposes an agent-based distributed concurrent design system architecture for complicated, large engineering systems and develops its experimental implementation for basic ship design. First, dependency and concurrency in design activities of engineering systems are discussed from the viewpoints of hierarchical system structure and entity-and-attribute relationship, and it reveals the possibilities of concurrent execution of design operations under systematic viewpoint. Second, the design operations are classified into assigning system attributes, selecting system components, adjusting system structures and configuring spatial arrangement. In the architecture, design operations for attributes that are major parts are modeled with object orientation, and they are distributed to agents by extending an object-oriented programming technique to an agent approach. The others are encapsulated as independent task agents. After describing key concepts for its computer implementation, this paper shows an application to basic ship design.

Key Words : Concurrent Engineering, Distributed Design System, Agents, Engineering Systems Design, Computer-Aided Design, Design Engineering, Ship Design

1 緒 言

エンジニアリングシステムの設計は，その対象規模も大きく，関連する技術領域も広範囲に及ぶことから，分散的に行われるとともに，膨大な設計情報の処理を必要とする．このような内容はコンカレント・エンジニアリングの範疇に属する問題であり，個別の設計処理がコンピュータ化されつつある現状を踏まえれば，本来の設計処理やコンピュータ援用処理を統合化することによって，新たな次元からのより一層の設計効率化が望まれるようになってくるものと考えられる．現に，データベース技術に基づいた設計情報の集中管理による統合化が進められつつあるが，設計は本来，情報に関する動的な過程であり，柔軟で有機的な統合化のためには，設計情報とシンセシス処理との双方を一括した上で，分散連繋化を進めることが望まれる．このとき，エンジニアリングシステムの設計においては設計処理の分散化が本質的であることから，シンセシスにおける情報構造に付随させて設計を分散させる一方，それらの間の融合や連携を行えるようにすることが重要である．そのための要件は多岐にわたるが，まずは，設計処理と関連情報を同時に分散させ連携させることが第一の課題であると考えられる．

本研究では，以上の認識のもと，設計対象のシステム構造に着目した分散並行型の設計支援システムを構成するためのエージェント方式によるアーキテクチャを提案する．すなわち，エンジニアリングシステムの設計作業における依存性と並列性がそのシステム構造に従っていることに着目した上で，システム属性の処理については，オブジェクト指向による設計知識表現を異なる計算資源となるエージェントに分散させつつ，システム構造の調整やシステム要素の空間的配置に関わる処理については，カプセル化されたエージェントとして分散させることによる設計支援システムのアーキテクチャと実装のための基本的な考え方を示す．最後に，船舶基本設計に対するプロトタイプシステムを構築して，それらの有効性を検証する．

2 設計の分散化・並行化・協調化

エンジニアリングシステムの典型としては，船舶や航空機，プラントなどをあげることができる[1]．これらのシステムは様々な内容を含んでおり，複数の技術領域が関与していることもあって，設計処理はいわゆるコンカレント・エンジニアリングが目的としているような形態で行われている．コンカレント・エンジニ

*原稿受付 1998 年 4 月 13 日
†正員, 大阪大学大学院工学研究科 (〒 565-0871 吹田市山田丘 2-1).

アリングは一般には，製品開発における設計や製造に関わる諸内容を並行的に進めることにより，開発期間の短縮化や製品品質の向上をはかろうとするものである．そのような処理の並行化は設計対象に潜む様々な依存関係と対峙することから，自ずと並行化には限界があるはずであり，そのような限界のもとでコンカレント・エンジニアリングの展開を考える必要がある．

特に，コンピュータの利用を前提として設計処理のコンカレント化を考えるためには，並行化の限界を明示的に把握することが重要であり，そのような視点に立てば，コンカレント・エンジニアリングの考え方をコンピュータ援用設計支援システムとして展開していくための段階を以下の 3 つに分けることができる．

分散化 … 全体システムに対する一連の設計処理を潜在的な依存性や並行性に従って個別の設計処理に分割する．

並行化 … 分割された設計処理を異なる計算資源において並行に実行できるようにし，それらの間での形式的な同期を行えるようにする．

協調化 … 形式的に並行化された設計処理の間で内容レベルでの協調を行い，さらに有機的に統合化を進めて，高度な並行処理を実現する．

本研究は，いわゆるエンジニアリングシステムを対象とした上で，これらのうちの前 2 者に関して，エージェント方式による分散並行化のためのアーキテクチャを構成することを目的としている．

3　エンジニアリングシステムの設計処理

3.1　システム表現における階層的構造　エンジニアリングシステムは，その性質として，様々な粒度のサブシステムに分割することができる．それらのサブシステムはさらに小さなサブシステムに分割できる一方，各サブシステムはそれぞれに属性の集合として表すことができる．形状情報はこのような属性情報の典型的なものである．例えば，船舶は長さや幅，吃水などの主要目により代表される一方で，それらに船体形状，船倉配置，エネルギープラントの構成，個々の船倉における構造，個々の船倉内における機器などの配置などの表現が付随しているとみることができる．

3.2　設計処理の分散並行性　設計表現におけるサブシステム間の階層関係に着目した場合，ある上位システムの属性がその下位システムの属性を拘束しているとみることができる．船舶の例では，主要目は各種の性能項目の概算評価値に基づいて決定される一方，その結果として，主機関の選定が行えるようになり，また，同時にそのような主要目の範囲内で船体の

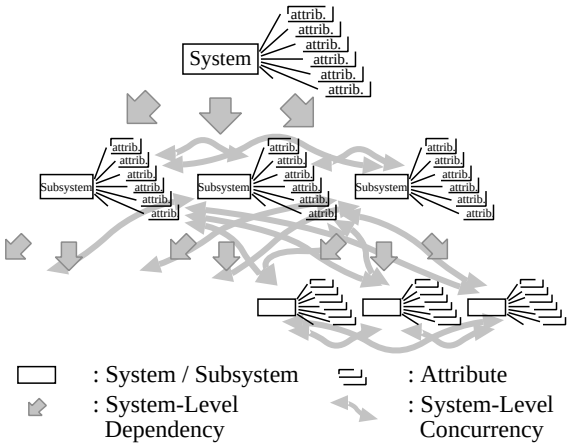

Fig. 1　System-level dependency and concurrency

形状を設計することができようになる．さらに，これらの結果を受けて，具体的な船体形状をもとに計算される推進抵抗が計算されると，選定された主機関の出力値がその値に対して確認される必要もある．これらに続いて，船倉の配置や機関室内の配置がそれぞれ並行的に行われていくことになる．

以上のような階層的なシステムの構造とそれらの間の独立性や依存関係は，設計処理を分散させ並行化する上での基盤となるものである．すなわち，図 1のように，システムの階層的な依存関係は設計処理における依存性に対応する一方で，そのような階層におけるあるレベルのサブシステム相互は設計処理の並行性に対応するものとみなすことができる．さらに，そのような同一レベルの並列性のもとでは，異なる上位システムを持つ下位システムどうしは，そのレベルに関わらず，同様の設計処理における並列性を有しているとみなすことができる．もちろん，システムが完全に独立的なサブシステムに分離できるわけではないため，以上のような並行性は設計処理の並行性を見い出すための荒い基準に過ぎないが，大局的にみれば，システム構造における関係に基づいて各設計処理を分散並行化する可能性が存在していると言える．

3.3　設計処理のタイプ　具体的な分散化や並列化を進めるためには，設計処理の内容についても考える必要がある．システムの内容は，図 1のように構造と属性として表現でき，さらに，構成要素の空間的な位置関係を定める配置が関わってくる．

船舶の基本設計における具体例に関連させれば，設計処理の内容は以下のように分類することができる．

構造のもとでの属性の決定 … システムの属性には，形状寸法などの諸条件に対して任意に決定す

るものと，既存の候補の中から条件に合うものを選択するものがある．以下では，これらの内容を‘パラメトリック設計’と呼ぶことにする．

船舶設計においては，主要目をはじめとして各種サブシステムの属性を定める必要がある．また，システムそのものの属性に加えて，それらの有する性能指標なども同様に算出する必要がある．また，主機関やエネルギープラントの構成機器を選定する必要がある．

システム構造の調整 … エンジニアリングシステムにおいては，多くの場合，基本的な構成は既に確立されており，何らかのテンプレートに基づいて構造が調整されるものとすることができる．

通常の船舶設計においても，システム構造の大部分は前もって定まっているが，船のサイズにより船倉の数を変更するなどのことが行われる．

システム要素の空間配置 … これについても，基本的な配置は標準化されているものの，例えば，主機関やエネルギープラントの構成機器を機関室に配置するなどの処理を行う必要がある．

これらのなかでも，設計が定型的である場合には処理量の点からはパラメトリック設計の重要度が大きい．

4 エージェント方式による設計の分散化と並行化

4.1 オブジェクト指向とエージェント方式 設計処理の分散化や並行化を進めるには，個別処理の独立性を高めることが前提となる．これに対して，パラメトリック設計において有効な表現・処理方法でもあるオブジェクト指向[(2)~(4)]は，個々の情報処理要素がオブジェクトとしてカプセル化されており，計算処理もメッセージ・パッシングによって行われることから，自律分散的な性質を備えている[(5)]．しかし，通常の実装においては必ずしも分散性に伴う並列性が具体化されているわけではない．そこで，本研究では「エージェント方式」を導入して，オブジェクトを異なる計算資源に分配し，処理の並列化をはかることにする．

図2はそのような分散並行化を模式的に示したものであり，設計処理を行う要素をオブジェクトとして自律分散的に表現する一方，それらを相互に関連性の強いものにまとめた上で，異なる計算資源に対応するエージェントに分散配置し，さらに，オブジェクトにおける設計処理を行うためのメッセージの送信を並列的に行うようにする．このとき，個別のエージェント内における処理はマルチタスク機能により疑似並列化されるのみであるが，複数のエージェントにまたがる内容に関しては，それらが異なる計算資源の上に存在

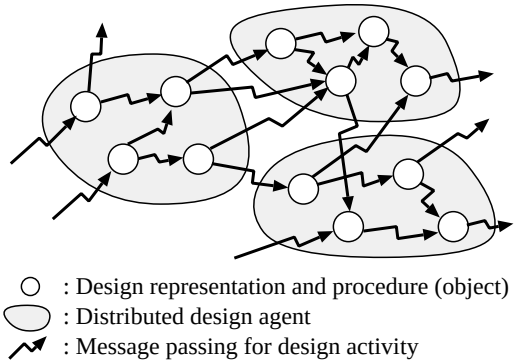

Fig. 2 Distributed object orientation

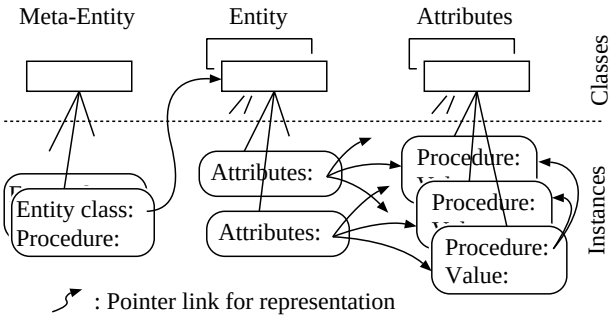

Fig. 3 Entity and attribute representation

していることから，エージェントの間で分散並行的な処理が実現できるようになる．

なお，エージェントの概念は様々に用いられているが，コンピュータ援用コンカレント・エンジニアリングの領域においても，単機能のエージェントを多数用いようとする立場[(6)]から， 比較的規模の大きい設計タスク毎のエージェントを導入しようとする立場[(7)]まで，様々である．本研究でのエージェントは，設計処理の異なる計算資源への分散化を目的とした比較的規模の大きいものである．いずれにしても，エージェントに対する期待は，表現や処理の分散化をはかった上でそれらの間での通信や協調によって，コンカレントな設計過程を実現しようとするものである．

4.2 パラメトリック設計のエージェント化 設計処理のエージェント化について，まず，根幹となるパラメトリック設計を分散並行化することを考える．

(1) オブジェクト指向による設計表現 パラメトリック設計については，オブジェクト指向による対象表現が有効である[(2)~(4)]．図3は，オブジェクト指向のもとで実体とその属性を表現するための枠組みを示したものである[1]．設計処理を行うためには，その結果を

[1] このような表現方法は文献[(2)~(4)]に基づいたものであるが，CLOSでの実装のために，それらのものからは変更が行われている．

保持するスロットと同様に，個々の項目についての手続きが必要となる．図においては，実体と属性のそれぞれがインスタンスオブジェクトとして表現され，後者は対応する前者によって保持されており，個々の属性オブジェクトはその値と手続きを保持するためのスロットを持っている．さらに，メタ実体のインスタンスは，設計要求に従って実体のインスタンスを生成するために導入するものであり，実体のインスタンスが属することになるクラスと具体的な生成手続きをそのスロットに記述している．

(2) 設計計算における依存性と並行性 以上の枠組みのもとでの主要な処理はオブジェクトの手続きに従ってその内容を決定することである．個々の内容はそのスロットに記述された手続きを通じて他のいくつかの内容に依存しているが，前節での依存性と並行性はこのような関係の再帰性によるものである．例えば，*A, B, C* という属性に関して，*A* の値が $B+2.0C$ という手続きによって決まるものと記述されている場合を考えれば，*A* の値の決定は *B* と *C* についての処理結果に依存している一方で，*B* と *C* についての設計処理そのものは並行的に行うことができることになる．

(3) メッセージの送信による設計処理 図3のように実体と属性として表現された設計対象の内容を操作して，設計を進めていくための手段として，以下のメソッドを起動するメッセージを用意する[(2)]．

decide ··· メッセージの送り元の値を決定するための必要性から，その (属性) オブジェクトの値の決定を依頼する．もし，そのオブジェクトが値を保持していない場合には，このメッセージは再帰的に送信されていくことになる．

cancel[2] ··· メッセージを受けたオブジェクトの保持している値を消去する．なお，その消去に先立ち，その値に依存した他の設計情報の内容も再帰的なメッセージ送信により消去しておく．

query ··· あるオブジェクトにその値を保持しているかどうかを問い合わせる．

以上のようなオブジェクトによる表現とメッセージによる処理をエージェント化するためには，オブジェクトをエージェント群に分散させた上で，上記のメッセージを送信する処理を並列化し，エージェント内でのメッセージ送信と同様に，異なるエージェントに存在するオブジェクトの間でもメッセージ送信ができるようにする必要がある．

[2] これは，以前[(2)]は *delete* と呼んでいたものに対応する．

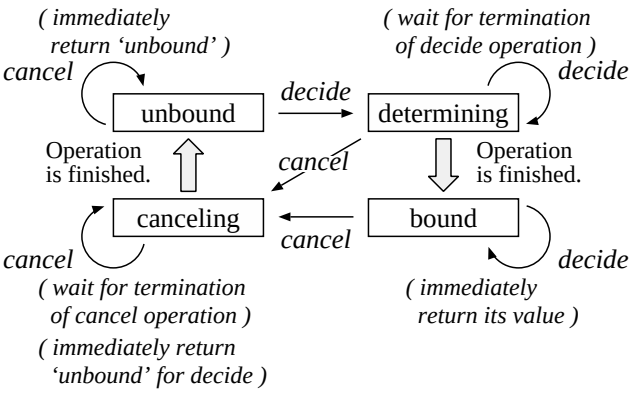

Fig. 4 State transition map for redesign mechanism

(4) 設計処理の同期 *decide* と *cancel* メッセージにより設計処理を並列化した場合，重複したり相互に競合するようなメッセージが同時に特定のオブジェクトに送信されることが起こる．例えば，その値を決定しようとしているオブジェクトに対して，さらに *decide* メッセージが送付されたり，処理中の *decide* メソッドの内容を直ちに中断すべきような *cancel* メッセージが送りつけられたりする状況が起こり得る．それらの状況に対応して設計処理を円滑に進めていくために，各オブジェクトの振舞いに関して以下の4つの状態を定義し，それらの状態に従って設計処理の内容を切り替えるようにする．

unbound 状態 ··· 値が決定されておらず，また，値を決定するための処理も行われていない状態．

determining 状態 ··· 値を決定するための処理が進行中の状態．

bound 状態 ··· 値が決定された状態．したがって，値を決定するための処理も完了し，何の処理も行われていない状態である．

canceling 状態 ··· 値とそれに関連する内容を消去するための処理が進行中である状態．

これらの状態がメッセージの受信に従って遷移する順序を図4に示す．このような状態遷移には，*determining* と *canceling* という他の設計処理との同期を取るための待ち状態に相当するものを含める必要がある．そのようなものは，逐次的な処理においては，処理の実行そのものがそのような状態を意味することから，明示的に導入する必要のなかったものである．

(5) 再設計のメカニズム 再設計は，パラメトリック設計に限らず，重要な設計局面である．上記のメソッドや状態のもとでは，再設計は，基本的には，まず *cancel* メッセージによって再設計に関連する旧設計の部分を消去した後で，再度 *decide* メッセージを送

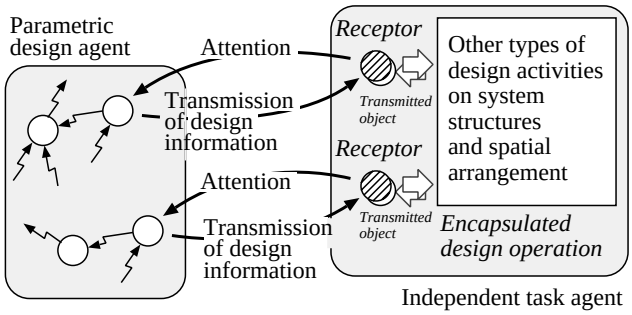

Fig. 5 Transmission of design information between agents through receptors

信することにより行うことができる．つまり，このような関係のもとで，設計者がある設計項目の内容を再設計しようとした場合には，対応するオブジェクトに *cancel*メッセージが送信され，手続きにおける依存関係に従って関連する項目が消去された後，そのオブジェクトに新たな設計内容を割り当てるようにする．これによって，後ほど，各オブジェクトに *decide*メッセージが送信された際に，その内容とそれを決定するために必要となる一連の項目が再帰的な *decide*メッセージによって再設計されることになる．

4.3 システム構造調整や空間配置のエージェント化 パラメトリック設計の分散化と同様に，システム構造や空間配置に関わる内容もエージェント化によって分散化させる．ここでは，それらの個別設計処理をそれぞれにエージェントとして独立させる (以下では，独立エージェントと称する) 一方，そのようなエージェントにパラメトリック設計を行うエージェントでの設計結果を取り込んで，両者の間の連係が行えるようにする．これによって，個別設計処理を独立的にコンピュータ化できるようにする．

図 5はこのような独立エージェントの考え方を示したものである．すなわち，独立エージェントが起動されると，まず，そのエージェントの受け持つ設計処理を始めるために必要となる設計情報を予約するという目的に対して， *receptor*というオブジェクトを生成する．このオブジェクトはいわば元来の設計情報を保持している実体や属性のオブジェクトのミラー媒体 (ミラーは *transmitted-object*のクラスのオブジェクトとして別途，生成される) である． *receptor*オブジェクトが生成されると，直ちに，対応するオブジェクトに予約を行うための *attention*メッセージを本来の実体や属性のオブジェクトに送信し，そのメッセージを受けたオブジェクトは設計情報に変更があったときに，そのことを通知すべきオブジェクトのリストに送信元のオブジェクト名を加える．これによって，パラメトリック設計における情報の変化は随時独立エージェントに移送されるようになり，独立エージェントは，以上のような *receptor*オブジェクト以外には，他のエージェントでの設計処理とは関係なく，自身の内容を記述して実行することができるようになる．

4.4 設計プロセスの管理 以上の内容は個別設計処理の分散化に関するものであったが，設計システムにおいては，それらの個別処理とは別に，設計処理を起動し，全体としての設計プロセスを形成・管理する機能が必要である．前述のように，エンジニアリングシステムは明確な構造を持っており，それに従って設計プロセスにおける順序や並行性が定まっている．これらの関係に従って，設計項目も大別されるはずであり，関連性の強い一連の代表的な設計項目をまとめて設計ステップと呼ぶことにする．設計プロセスをそのようなステップをもとに記述することができる[(2)]．このような記述のもとでは，ある設計ステップを構成する代表的な設計項目に対して *decide*メッセージを送信することによって，そのステップを起動することができるようになる．さらに，そのような操作を各ステップの間でどのような順序や同時性のもとで進めていくかを管理することによって，設計プロセスを管理することができるようになる．

4.5 設計診断の機能 設計は，本来，設計・検証・評価・修正 (再設計) を繰り返す過程であることから，検証を行うための項目をルール化した[(2)]上で，それらによる検証処理を他の処理と統合化する．すなわち，検証内容をルールとして項目化した場合，各ルールは属性項目間の不等式や等式として表現でき，関連する設計項目が定まるとそのルールを適用できるようになることから，検証項目を表現・処理するために，独立エージェントにおいて用いた *receptor*のメカニズム (図 5) を流用して，ルールに含まれる設計項目についての情報を移送するようにする．これによって，各ルールは適切な時点で評価できるようになる．実際のルールによる検証は，上記の設計プロセス管理と関連させて，各設計ステップの内容が確定した後に，一連の検証ルールを評価するようにして，設計者が設計処理についての判断が容易にできるようにする．

5 分散並行型設計支援システムのコンピュータ実装

5.1 基本環境 エージェント方式による設計の分散並行化をシステムとして実装するにあたっては，一連の UNIX ワークステーションをクラスター化し，

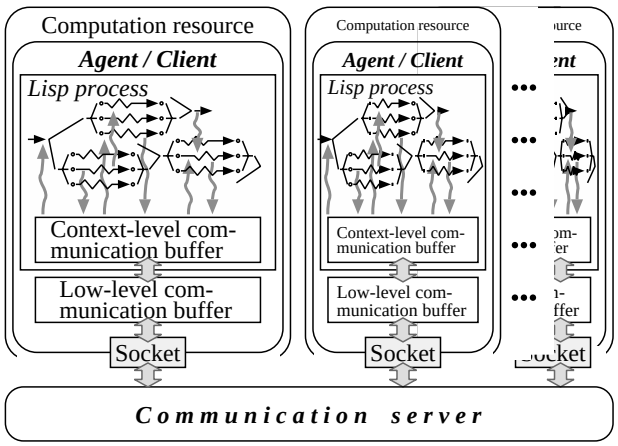

Fig. 6 Communication server based agent communication

Allegro Common Lisp (ACL)[8]をシステム構築基本言語として用いる一方，通信サーバを経由してそれらの間で TCP/IP ソケットによる通信を行なう形態をとる (図 6)．エージェント間のメッセージ通信を実現するためには，任意のタイミングで送信されてくるメーセッジに対して随時各エージェントが反応できる必要があり，また，エージェント内部での設計処理においても，他のエージェントに依頼した処理が終了するまでの間にエージェント内で可能な処理を並行的に実行していく必要からも， ACL の持つマルチタスク機能は必須であり，また，個別の設計処理を実装においても， Lisp の記号処理機能と CLOS[9]によるオブジェクト指向プログラミングが有用である．一方，通信サーバは C 言語で実装されており，ただ単に各エージェントから送信されてくるメッセージをタグに示されたエージェントに転送する機能のみを持っている．

5.2 エージェント間通信 前述の目的でオブジェクト指向をエージェント方式に展開するために，オブジェクト指向におけるメッセージ・パッシングを個別の計算資源内と同様に異なる計算資源の間でも行えるようにする．このようなメッセージ・パッシングの拡張のために，以下の 2 つのメッセージをエージェント間でのメッセージとして導入する．

Send 型メッセージ … あるオブジェクトから別のエージェントに存在するオブジェクトに (オブジェクト指向) メッセージを送信するための (エージェント) メッセージである．このメッセージの機能は，送信先と送信内容からなるメッセージをエージェント間の通信チャンネルに送り込み，それに対する返事が届くのを待つようにする一方，通信チャンネルの両側でエージェントメッセージとオブジェクト指向メッセージを相互に変換するためのしくみを用意することにより，実現することができる．

Broadcast 型メッセージ … すべてのエージェントにある送信内容を送信するためのメッセージであり，例えば，特定の名前のオブジェクトがどのエージェントに存在するかを調べるなどのために用いられる．なお，実現方法は送信先が明示的でない以外は *Send*型と同様である．

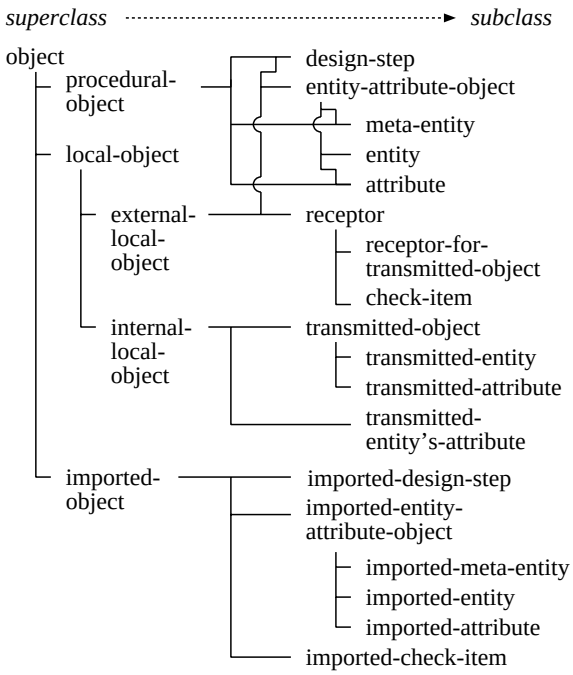

Fig. 7 Class hierarchy

エージェント方式により具体的な計算処理を行うために，さらに，以下の 2 つのものを導入する．

Agent setup 型メッセージ … 必要に応じて，新しいエージェントを起動するためのメッセージ．

Remote procedure call 型メッセージ … 別のエージェントにおいて特定のプログラミング言語で記述された一連の処理を実行するためのメッセージであり，例えば，あるエージェントにおける処理を初期化するためなどに用いる．

5.3 オブジェクト指向におけるクラス階層 図 7はオブジェクト指向プログラミングにおける基本的なクラス階層を示したものであり，主要なクラスの内容を以下に示す．

ローカル・オブジェクト … そのエージェントに存在するオブジェクトに対するクラス．

外部 (ローカル・) オブジェクト … 他のエージェントから参照することのできるローカル・オブジェクトに対するクラス.

内部 (ローカル・) オブジェクト … 他のエージェントから参照することのできないローカル・オブジェクトに対するクラス.

インポート・オブジェクト … 他のエージェントに存在するオブジェクトの参照媒体となるオブジェクトに対するクラス. このようなオブジェクトは, その名称と元々のオブジェクトが存在するエージェントの識別子のみを保持するものであり, あるエージェントが他のエージェントに存在するオブジェクトを見つけるために *Broadcast*型メッセージを送信した後で生成される.

以上のクラス階層のもとでは, インポート・オブジェクトが各外部オブジェクトについて必要である.

前節での内容については, パラメトリック設計のために meta-entity, entity, attribute の各クラスが, 独立エージェントとのインターフェースのために receptor と transmitted-object が, 設計プロセス管理のために design-step が, 設計評価のために check-item が, 定義されている. なお, meta-entity, attribute, design-step の各クラスにおいては, その内容を決定するためのメソッドが procedural-object クラスにより共通的に定義されている. 一方, 独立エージェントにおいてはこれらのものとは異なるクラスが別途必要となるが, それらについては図 5に示したしくみにより, 図 7の内容とは関係なく, 導入できるようになっている.

6 船舶基本設計におけるプロトタイプシステム

6.1 船舶の基本設計 船舶の基本設計では, 要求される船のサイズや速力を満たし, かつ安全性や運航コストなどの評価尺度を満足化するように, 主要目, 船体形状, 貨物倉の配置, 動力推進プラントの構成, 機関室配置などを決定することが求められる[(10)]. 船舶設計における設計手法と設計知識は, 過去の研究によって確立され系統立てられているが, 商船は大規模で複雑なシステムであり, 設計処理の内容も膨大であり多様であることから, 緒言でも述べた意味において, 分散並行型の設計支援システムの意義は大きい.

6.2 プロトタイプシステムの構成 船舶の基本設計では, 設計対象は, まず, 一連の主要目によって表現され, そのもとに船体形状, 貨物倉の配置, 動力推進プラントといった様々なレベルのサブシステムが存在する. このようなシステム構成に対する設計処理相互の関係は図 8のようになっている. これに対して, 上記のアーキテクチャに従った分散型の設計支援システムを構成するにあたり, 本研究では, 設計処理を, パラメトリック設計については基本設計・主機関選定・船体設計・安定性分析の各エージェントとして分散させる一方, プラント計画・機関室配置に関する独立タスクエージェントを連係させるようにする. なお, これらのエージェントにおける設計知識の内容や設計処理の方法は既に開発済みのもの[(2)~(4) (11) (12)]をエージェント方式によるアーキテクチャに向けて修正したものでもある.

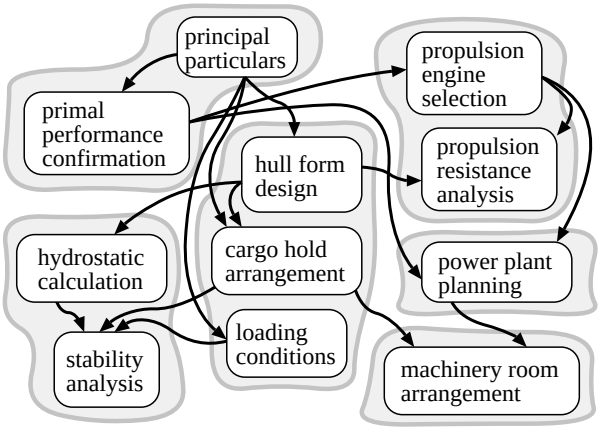

Fig. 8 Design process of basic ship design

6.3 システムによる設計処理例 構築した設計支援システムを船舶基本設計に適用した状況を図 9に模式的に示す. 本事例では, 載貨重量 38,000ton, 航海速力 14.9knot のばら積み貨物船の設計を行なっている. 図中①は, 主要目の決定に関わる設計と再設計の繰り返しを示している. ②は主機関の選定過程を示し, ③では船体の形状設計が正面線図と全体の投影図として示されている. ④では舶用プラントの設計が行われており, 総コストの点で優れたプラント候補が列挙されるとともに, 最終的に選択されたプラントの構成が示されている. ⑤では安定性解析が行われ, 静水力学曲線と GZ 曲線が示されている. ⑥では機関室の配置設計が行われており, 主機関の周囲に様々な補機が配置されていく様子が示されている.

このようなシステムの実行においては, 必要な情報が整った時点で個々の設計処理が開始されることや, 再設計が行なわれた際には, 再設計に関わっていた内容が消去されることによって他のエージェントにおいても設計処理が一時中断され, 再び内容が決定されてから中断された処理が自動的に再開されるなどのことが可能であることを確認できた. これによって, 本研究で提案したアーキテクチャとその実装方法が, 設計

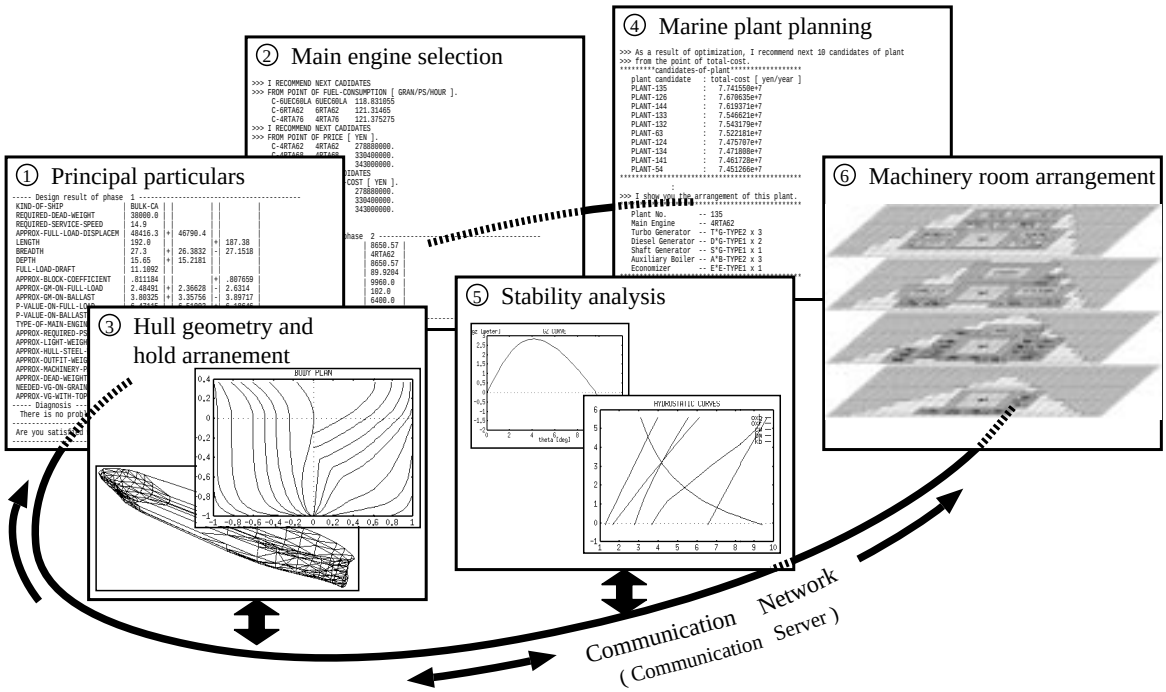

Fig. 9 Distributed basic ship design system

処理を複数のエージェントに分散させた上で，それらが並行的に実行されるようにするための基盤として有効であることを実証することができた．

7 結 言

本研究では，エンジニアリングシステムの次世代の設計支援システムに関して，システム構造に付随する依存性や並行性に従って，設計情報とその設計処理とを相互に関連付けたまま分散させるためのアーキテクチャを提案し，プロトタイプシステムの構築とその船舶の基本設計問題への適用事例を示した．このような考え方は，コンピュータ援用コンカレント・エンジニアリングシステムに向けての方法論として有効であると考えられ，今後は，単に情報とその処理を分散させるだけではなく，設計処理順序や依存関係を積極的に管理することによる協調化を進めて，より効率的な設計処理が行えるようなアーキテクチャへと拡張していくことが必要であると考えている．

なお，本研究の一部は，日本学術振興協会未来開拓学術研究推進事業 (96P00702)，ならびに，文部省科学研究費補助金 (08405016) の援助によるものである．

文 献

(1) 赤木, システム工学 — エンジニアリングシステムの解析と計画 —, (1992), 共立出版.
(2) 赤木・藤田, オブジェクト指向に基づく設計エキスパートシステムの研究, 日本機械学会論文集 C 編, Vol. 54, No. 500, (1988), pp. 1017-1025.
(3) 赤木・藤田・窪西, プラント設計におけるエキスパート CAD システムの研究, 日本機械学会論文集 C 編, Vol. 54, No. 497, (1988), pp. 228-233.
(4) 藤田・赤木, 機能設計における設計対象のモデリングと形状モデルとの融合, 日本機械学会論文集 C 編, Vol. 57, No. 535, (1991), pp. 1058-1065.
(5) Booch, G., *Object-Oriented Design with Applications*, (1991), Benjamin/Cummings Publishing.
(6) Brown, D. C., Using single function agents for design, *Knowledge Intensive CAD, Vol. 1, edited by Tomiyama, T., et al.*, (1996), pp, 15-20, Chapman & Hall.
(7) Park, H., Cutkosky, M. R., Conru, A. B. and Lee, S. H., Agent-based approach to concurrent cable harness design, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, Vol. 8, No. 1, (1994), pp. 45-61.
(8) *Allegro CL User Guide, version 4.3*, (1996), Frantz Inc.
(9) Steele, G. L. Jr., *Common Lisp — The Language (2nd edition)*, (1990), Digital Press.
(10) 冨田, 船舶基本設計論, (1982), 丸善出版サービスセンター.
(11) 藤田・赤木, AI 手法によるエネルギープラントの設計法 — 舶用プラントおよび陸用コー・ジェネレーションプラント設計への適用 —, 日本舶用機関学会誌, Vol. 27, No. 5, (1992), pp. 390-399.
(12) 赤木・藤田, 制約指向に基づく基本配置設計支援システムの研究 (第 1 報：制約指向による基本配置アルゴリズム), 日本機械学会論文集 C 編, Vol. 56, No. 528, (1990), pp. 2286-2293.