2005－BIO－2（3）
2005／10／7

in-Silico スクリーニングを支援するワークフロースキーマの設計と実装

木戸善之[†1,†4]　伊達進[†1]　前野隆志[†1]
長谷川一郎[†3]　下條真司[†2]　松田秀雄[†1]

近年，創薬過程における化合物を探索する工程，*in-Silico* スクリーニングはライフサイエンス分野において注目を集めている．しかしながら *in-Silico* スクリーニングでは，様々なツールを組み合わせた利用や，パラメータの調整といった試行錯誤的な作業を必要とし，その効率性が課題である．さらに煩雑な作業手順を複数の研究者らと共有する手段が未だ確立されていない．本論文では最適な手段およびパラメータの発見のために柔軟な変更・実行ができるワークフロー実行環境と，作業手順や知識の共有を目標としたワークフロースキーマの設計・実装について報告する．

Design of Workflow Schema for *in-Silico* Screening

Yoshiyuki Kido,[†1,†4] Susumu Date,[†1] Takashi Maeno,[†1]
Ichiro Hasegawa,[†3] Shinji Shimojo[†2] and Hideo Matsuda [†1]

Docking simulation, which is one of *in-Silico* screening methods, has an important role in the process of drug discovery. The docking simulation is applied to narrow compounds in database down to a few compounds that can become candidates of drugs. However, the docking simulation has too many difficult factors. Scientists have to coordinate whole parameter sets in trial and error processes. Sequences of the processes are interrupted by checking a coordinates of the compound on protein. In addition to that, it is difficult for inexperienced scientists to make up the sequencing processes, because building the processes requires much technical knowledge of docking simulation. In this paper, we describe a design and a development of workflow schema for *in-Silico* screening. This workflow schema promises to share technical knowledge, and to achieve more throughput by reducing interruptions of workflows.

1. 背　　景

近年，情報科学の発達によってバイオインフォマティックスが目覚ましく進展を遂げている．それにより医療，創薬，食料品などライフサイエンスにおける研究や開発が加速化，合理化を推し進められている．創薬プロセスにおけるバイオインフォマティックスによる加速化，合理化は主に予測とモデリングによってもたらされる期待されている．ゲノムやたんぱく質，化合物データベースの充実により，たんぱく質の分子構造や機能が解明されつつあり，未知のたんぱく質や化合物の機能についても予測が可能になりつつある．また機能や振る舞いを予測することにより，分子レベルでの可視化によって分子モデリングが可能となる．さらにリガンドになりうる化合物を絞り込む作業，スクリーニングにおいて，実際に試料を用いた実験を行う *in-Vitro* スクリーニングの前の段階で *in-Silico* スクリーニングを行うことにより，リガンド候補を予測することが可能となり，有用性が示されている[11]．またたんぱく質の変動を予測することによって主作用のみならず，同時に副作用となりうる遺伝子やたんぱく質の情報も得る事が可能になると考えられている．こうしたたんぱく質，化合物の特性や作用を *in-Vitro* スクリーニングの事前に予測する行為は，コストや効率の面で大きく寄与すると考えられ，創薬プロセスにおいて *in-Silico* スクリーニングは加速化，合理化において非常に重要であると言える．*in-Silico* スクリーニングにおいて計算手法やデータ転送効率など様々な手

†1 大阪大学情報科学研究科
Information Science and Technology, Osaka University
†2 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University
†3 情報通信研究機構
National Institute of Information and Communications Technology
†4 三井情報開発株式会社
Mitsui Knowledge Industry

法を用い，計算科学の領域にて効率化のアプローチが試みられている．その例としては，ドッキングシミュレーションのソフトウェア DOCK[1] を分散並列計算環境であるグリッドコンピューティング[10] の実装である Globus Toolkit[5] にて動作させる方法が試みられている[13]．数万のリガンド候補に対してスクリーニングを行う際には非常に有効な手段であると言えるが，依然問題点を残している．

上述した *in-Silico* スクリーニングに対する計算機科学のアプローチは，分散並列処理を行うことによるハイスループット *in-Silico* スクリーニングを実現しているが，問題点とは *in-Silico* スクリーニングの持つ複雑性，専門性にあると著者らは考えている．*in-Silico* スクリーニングに主に用いられる手法としてはたんぱく質と化合物のドッキングシミュレーションが挙げられる．ドッキングシミュレーションのソフトウェアには DOCK[1]，AutoDock[2]，GOLD[3]，presto X[4] などがあり，それぞれに特色の違いはあるがいずれも複雑な手順やパラメータ調整が必要となる．ドッキングシミュレーションまでの決定事項をパラメータなり対話形式なりでユーザが決定しなければならない．それらの決定にはソフトウェアに含まれる様々なツールにおけるパラメータを調整することによって，ユーザが試行錯誤を行い決定する．パラメータの調整には，受容体や酵素などのたんぱく質や核酸の活性部位における立体構造や，分子レベルでの電子的特徴を元に決定する必要があるため，専門知識が必要不可欠となる．そのためツールによっては 70 以上のパラメータが存在する．こうした複雑性は試行錯誤を繰り返すうちに経験則として知識に蓄えられていくわけだが，また自分自身で行った膨大な数のスクリーニング結果，成功例，失敗例などを参考にする仕組みが必要とされている．さらに，経験のあさい研究者らに知識を共有する手段としても考えられ，効率化，合理化に繋がると著者らは考えている．

本研究では，得られる結果や知識を蓄積，共有する *in-Silico* スクリーニング支援環境を構築し，ライフサイエンス研究の効率化，合理化を進めることを目指す．そのための準備段階としてドッキングシミュレーションの複雑性を考慮したワークフロースキーマの設計，開発を行った．本稿では，２章にて具体例を挙げたドッキングシミュレーションの複雑性について，３章にてワークフローの設計と実装について，４章にてまとめと今後の課題について，それぞれ述べる．

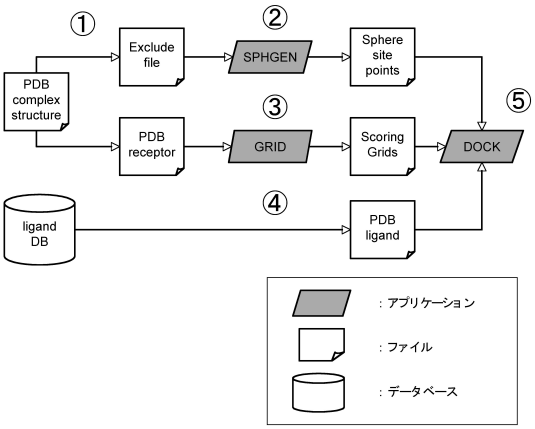

図 1 DOCK 全体のフロー
Fig. 1 DOCK flow overview

2. ドッキングシミュレーションの複雑性

本章ではドッキングシミュレーションの具体的な例を挙げ，ワークフロースキーマの要件を述べる．著者らは，ドッキングシミュレーションのアプリケーションとして代表的であり，また汎用性の高い DOCK を採用した．ドッキングシミュレーションや立体配座解析のソフトウェアは現在までに数多くリリースされているが，DOCK はその中で最も古い部類に相当する．DOCK ではたんぱく質活性部位の化合物結合可能部位にあらかじめ複数の球体を設定し，球体を誤差範囲として結合可能な化合物を網羅的に検索する．得られた結合可能な化合物に対しドッキングシミュレーションを行うソフトウェアツール群であり，近年でもなお有用性が示されている[12]．

DOCK の主たるフローを要約すると図 1 になる．図中の注釈にあるように，網掛けの菱形がアプリケーションとなっており，また DOCK のフローにはファイル，データベースが存在する．図 1 中にある番号は手順の順番を示している．その番号に沿って手順を説明する．

(1) ターゲットとなるたんぱく質の活性部位と計算外領域に分割する．
(2) SPEGEN を使用し，計算外領域から化合物の結合部位の表面を抽出し，その表面に沿う形で球を設定する．この球は化合物の配座位置を示している．
(3) 活性部位にあたる部分を格子状で切り分け，それぞれの格子に含まれる化合物の化学特性を踏まえスコアリングを行う．
(4) 化合物データベースからリガンドとなりうる化

合物を取得する．

(5)　(2)(3)(4)で得られた結果と併せ網羅的なドッキングシミュレーションを行う．

ここで注意すべき点は，このフローは概ねのフローであって，DOCKがドッキングシュミレーションを行うためには本来，図2に示すようにアプリケーションとデータファイルが複雑なフローを構成している．これらを大雑把に分割すると２つのフローで形成される．本章では以下の項目において，それらフローを元にモデル化を行い，ワークフロースキーマの対する要求を述べる．

2.1　配座位置の設定

まず，たんぱく質活性部位における化合物の配座位置の大まかな範囲を指定する必要がある．解析するたんぱく質や化合物が立体構造上自由度が少なく，かつ立体構造と配座位置における距離，角度，分子静電ポテンシャルなどが決定すれば比較的容易な計算でシミュレーションを行うことが可能だが，多くの場合，特に高分子化合物は立体構造の自由度が高いものが多い．自由度が高ければ，どこにどういう角度でドッキングをする可能性があるのかを洗い出す多次元問題に発展し，非常に困難になる可能性がある．それら計算範囲，結合部位の大まかな位置を決定するには，研究者らの経験則や知識に基づき場合が多い．例えば化合物の配座位置にたんぱく質の分子が重なってしまった場合や，配座位置が大きな誤差が生じるなどした場合，図3の(5)で示す可視化にて妥当性が確認できる．つまり図3で示した手順は(2)で設定したパラメータの結果を(5)で確認しながら試行錯誤を行うフローとなる．図3の手順を以下にまとめる．

(1)　活性発現状態にあるたんぱく質と薬物化合物の立体構造を分離する作業を行う．

(2)　計算範囲，活性部位を指定する．

(3)　活性部位になりうる箇所の表面を算出する．

(4)　活性部位表面から割り出した化合物の配座位置を球状で示す．

(5)　可視化で妥当性を確認する．

2.2　ドッキングシミュレーション

2.1節にて述べた配座位置を決定した結果を元にしたドッキングシミュレーションを行うフローを説明する．図4の図中の番号はその手順を示しており，以下に列挙する．

(1)　配座位置を決定したたんぱく質の立体構造が記載されたファイルを取得する．

(2)　配座位置を格子状に区切り，格子点に対し分子間相互作用場として静電分子ポテンシャルを算出する．それによりリガンド候補のドッキング可能性をスコアで示すことが可能となる．

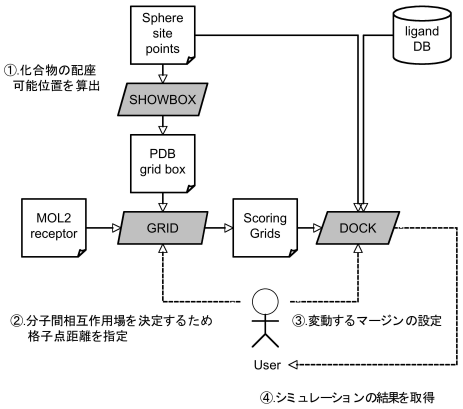

図 4　ドッキングシミュレーション
Fig. 4　DOCK Simulation

(3)　(1)の配座位置を示した立体構造，(2)で示した分子の化学特性やポテンシャルなどを示した数値，リガンド候補となる化合物の立体構造の３つを合わせドッキングシミュレーションが成立する．

またドッキングシミュレーションを行う際にパラメータとして，化合物の回転角度や力場の特性などをパラメータとして設定する事も可能となっている．

2.3　DOCK のモデル化

2.1節，2.2節で述べたそれぞれのフローは複雑性と単純性を示している．経験豊富な研究者であるなら，処理の手順はすぐに理解できまた組み立てる事も可能であるが，経験の少ない研究者であるならこれらフローの組み立てや，またツールが変更するたびに修得しなければならず，作業効率の面で著しく低下することとなる．図3の配座位置決定までのフローを簡潔に記述すると，図5となる．最初のAutoMSに対するオプションやパラメータとして計算領域を設定し（図5の”IN”），フローの末尾，SHOWSHPEREの出力であるファイル（図5の”OUT”）で配座位置の確認をするためのインタラクションが発生する．つまりフロー自体は，ツール間を入力と出力をつなぎ合わせる事で表現が可能である．こうしたフローを作成する作業に専門的な知識が必要であるが，フローを組み立てる要素としては

- 入力
- 出力
- ツール（アルゴリズム）

であると言える．

さらにDOCKにまつわるツールに関する入力と出

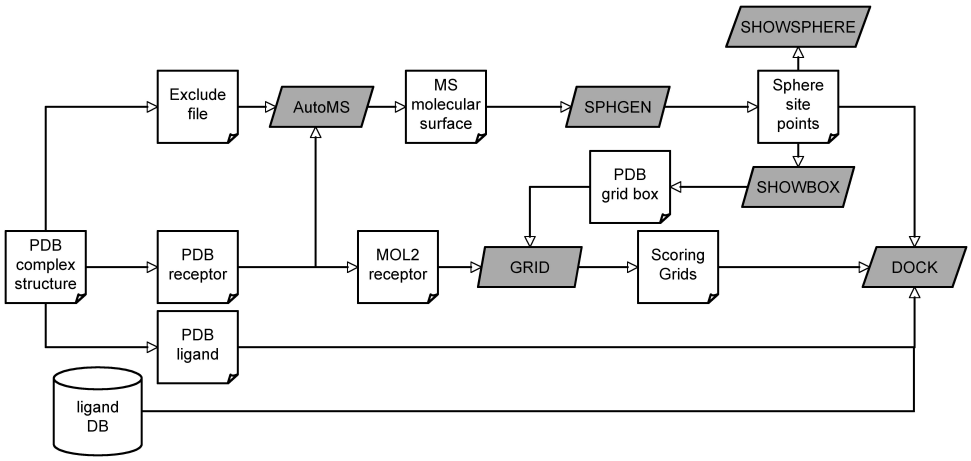

図 2 DOCK の詳細
Fig. 2 Detail of DOCK flow

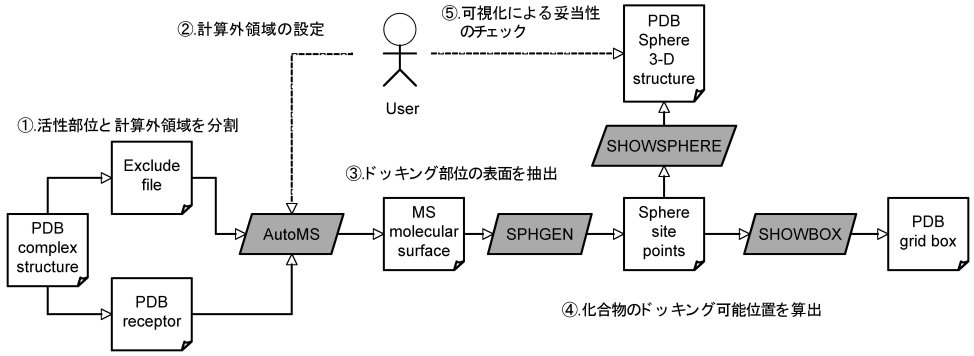

図 3 活性部位確認までの手順
Fig. 3 The flow of Viewing Active site

力の具体的な要素を整理すると図 6 のようになる．入力の実際の形態は

- ファイル，標準入力，オプション

などであり，出力の形態は

- ファイル，標準出力／標準エラー出力

である．汎用を高めることを考慮する場合，ネットワークによる入出力なども考えられる．ここで重要な点は，それらの形態が変容することを考慮し実装する必要があるという点である．つまり入出力には様々な形態，様式が存在し，それらをワークフローに記述する必要があり，ワークフローを解釈するパーサにその形態へ対応する実装が必要となる．入出力を構成する要素を繋げることによって，ワークフローの記述が可能となる．

2.4 ワークフローの要件

ドッキングシミュレーションにおいてワークフローに求められる点は，手順を知識として蓄積するだけでなく，効率化という点でも要求がある．ワークフローにて様式化された手順を複数回繰り返す必要があり，その点で効率化を求められている．*in-Silico* スクリーニングのメリットは，*in-Vitro* や *in-Vivo* の試験の前にリガンド候補を絞り込むことであるので，化合物

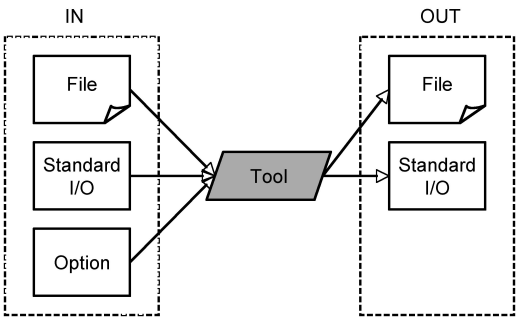

図 6 ツールのモデル化
Fig. 6 The model case of tool

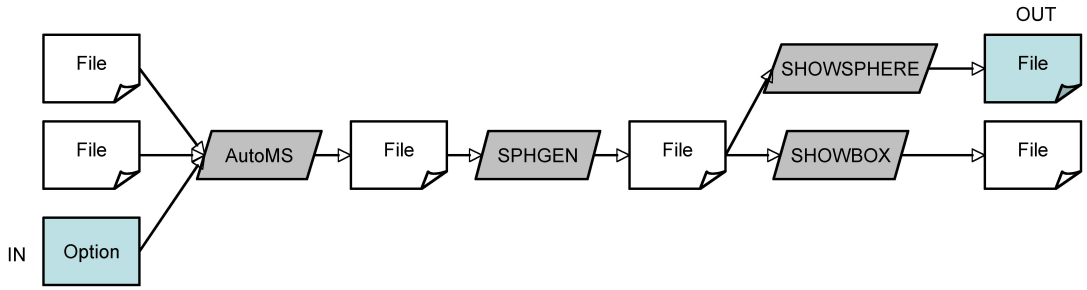

図 5 フローのモデル化
Fig. 5 The model case of flow

データベースより数万という化合物に対し *in-Silico* スクリーニングを行わなければならない．つまり図 4 で示した手順は，より多くの化合物や，さらに配座位置や GRID で求めた分子間相互作用場の特性などを調整しながら行う．つまり効率化を目指す場合，スクリーニングを行う化合物の候補の数，およびパラメータの数，パラメータの調整幅に比例して，並列度を高める必要が生じる．つまりドッキングシミュレーションのワークフロー化において要求される点は

- 知見の共有
- 作業の高効率化

と言える．

3. ワークフローの設計と実装

本章では，著者らが行ったワークフロースキーマの設計と実装について述べる．ワークフロースキーマの設計にあたっては 2.4 章で述べた要求を満たすと共に，ワークフローの実行結果を保存することも考慮する．ワークフロースキーマの全体概要を Entity-Relationship Diagram (ERD) で記述した図 7 に示す．以下の項目では図 7 に示した 3 層それぞれについて述べる．

3.1 フローテンプレート層

本節ではワークフローを記述するための定義群について述べる．ワークフローにて実行し結果を得る環境では，研究者がワークフローを組み立る段階と，既に組みあがったワークフローを実行する段階に分けて考えられる．実際にはワークフローを記述する行為と，ワークフローを実行する行為をこれらを繰り返しながら試行錯誤するケースもあるが，短い間隔で 2 つの段階を行き来するものと捉えることができる．そこでそれぞれの段階に併せて設計を行う．まず研究者がワークフローを記述，保存する行為に対し，ワークフローを定義するスキーマが必要となる．ワークフローを記述し，保存するためのスキーマ群をフローテンプレート層と定義した．フローテンプレートではワークフローを記述する項目，つまり図 6 が示すツール周辺の要素の定義が必要となる．さらにワークフローはツールの入力と出力をつなぎ合わせることで成立するため，入出力をつなぎ合わせる定義，ワークフローをひとまとめに管理する定義が求められる．図 7 に記述されたフローテンプレート層のテーブルについての説明を以下にまとめる．

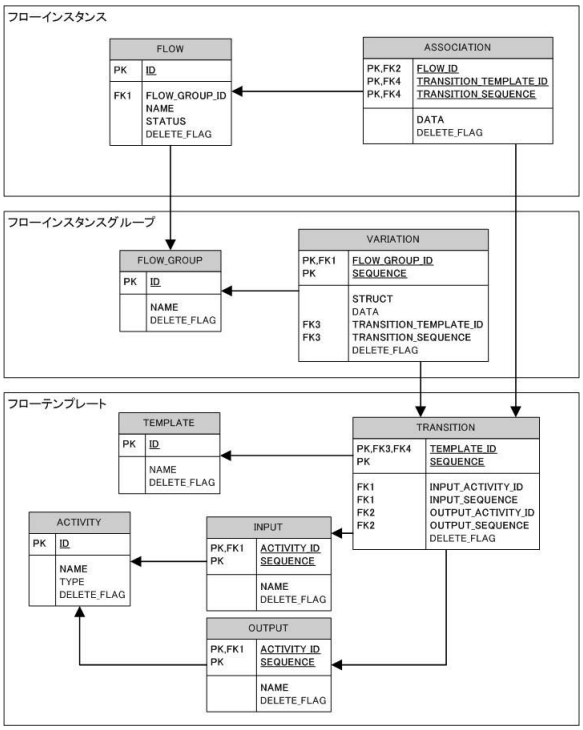

図 7 ワークフロースキーマの概要
Fig. 7 The overview of the workflow schema

- `ACTIVITY` ：アプリケーションの定義
- `INPUT` ：入力の定義
- `OUTPUT` ：出力の定義
- `TRANSITION` ：入力と出力をつなぎあわせる定義

- TEMPLATE　：ワークフローをひとまとめに管理する定義

ワークフローを構成する重要なカラムとして，TRANSITION テーブルにおける INPUT_ACTIVITY_ID, INPUT_SEQUENCE, OUTPUT_ACTIVITY_ID, OUTPUT_SEQUENCE が挙げられる．ツールの一意性は ACTIVITY テーブルの ID カラムを Public Key (PK) とすることでで保たれる．ツールの入力と出力を定義する INPUT テーブルと OUTPUT テーブルでは，複数の入出力を管理するため SEQUENCE カラムがあり，ACTIVITY_ID カラムと併せて PK としている．それら２つのカラムを INPUT, OUTPUT それぞれのテーブルに対し Foreign Key (FK) として TRANSITION テーブルに持たせる事でワークフローの接続を実現している．これらを用いることによって，ワークフローの組み立て，蓄積が可能となる．

3.2 フローインスタンス層

既に組みあがったワークフローを用いて実行する段階では，実行結果やワークフローの状態を蓄積するための定義が必要となる．本節ではワークフローを実行した結果を蓄積するためのスキーマについて述べる．図 7 では最上段にあたるこの層では，ワークフローの状態とユニークな ID で管理するための FLOW テーブルと実際の結果をデータとして保持するための ASSOCIATION テーブルの２つで構成されている．ASSOCIATION テーブルの DATA カラムに実際の結果が蓄積され，蓄積された結果がどの入出力に対応するかを指し示さなければならない．そのためにフローテンプレート層の TRANSITION テーブルに関連付けられている．また FLOW テーブルにおいて STATUS カラムでは，実行したワークフローが現在どのような状態にあるかを示すものである．STATUS カラムには実行中，正常終了，異常終了などの計算機が能動的にステータスを更新するほか，研究者が期待された結果が得られたか，失敗であるかを記述することもできる．これにより研究者はワークフローの結果を成功例，失敗例として分別することが可能となる．結果の分別において重要なのは，上述したステータスを記述することにある．調整したパラメータに対し，その結果が成功したのか失敗したのかが，経験則に基づく知識となる場合が多い．こうした試行錯誤をする過程を残す手段，特に失敗例を示すことによって，経験の浅い研究者らは最適解を導き出すことが容易になる．

3.3 フローインスタンスグループ層

図 7 で示した層は 3 層あり，その中間に位置するのがフローインスタンスグループ層である．この層では，複数のワークフローを実行するためにワークフローをグループ化するためのスキーマ群であり，2.4 節で述べた要件である高効率化を目指すものである．VARIATION テーブルは TRANSITION テーブルに対して関連づけられており，TRANSITION テーブルの入出力に対し変動幅を記述するためのテーブルである．具体的には STRUCT カラムにデータタイプを入力し，DATA カラムに変動幅を記述する．図 8 はパラメータの変動幅の記述例である．変動パラメータは大きく分けてステップ型と列挙型がある．ステップ型は変動するステップ数と開始と終端を記述する．開始から終端までをステップ数毎にパラメータとするものである．図 8 の param A では，0 から 1000 までの値を 10 ステップ毎にパラメータ入力を行う例であり．0,10,20...990 までの 100 のパラメータの値が発生する．図 8 の param B は文字列の列挙でのパラメータ入力を行う例である．これによりユーザは，パラメータの調整幅を事前に設定でき，ワークフローを繰り返し実行する手間を省くことが可能となる．

```
param A=1,1000,10
param B=AK095084,BC006195,U18197,X64330
```

図 8 サンプル変動パラメータ
Fig. 8 Sample of alignment parameters

3.4 実　　装

本章でこれまで述べてきたワークフロースキーマを解釈しワークフローを実行する環境を実装，構築した．ワークフロースキーマは Relational DataBase Management System (RDBMS) である MySQL を利用した．またワークフローを解釈するための実装については Web サービスの技術である SOAP を利用し，分散実行環境への対応を行った．システムアーキテクチャを図**??**に示す．実装は２つの層に分離して行った．Workflow Engine では，3.1 節，3.2 節で説明したフローテンプレート層とフローインスタンス層を解釈するものであり，ワークフロー記述とパラメータの解釈および実行結果の管理を行う．また Parameter Sweep Engine では，3.3 節で説明したフローインスタンスグループ層のパラメータの変動幅を解釈し，Workflow Engine へワークフローの実行依頼を行う．２層に分離して実装を行った理由としては，ワークフローの実行環境においてパラメータの調整，変動する必要性は付加機能であるという点である．分離することにより互いの汎用性が高められ，今後の課題に対応するアーキ

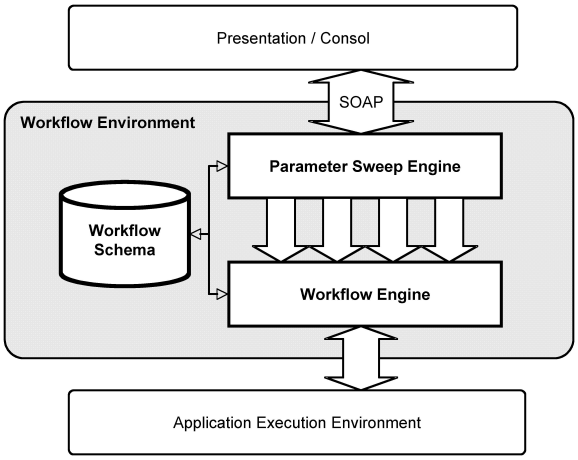

図 9 ワークフロー実行環境アーキテクチャ
Fig. 9 Architecture of workflow environment

テクチャの変更や開発に対し柔軟な対応が容易となる．

4. まとめと今後の課題

本研究ではドッキングシミュレーションをモデル化し，ワークフロースキーマの設計とワークフローを解釈するミドルウェアの実装を行った．ワークフローを記述，保存，実行する環境を構築し，結果を蓄積する環境を提供することによって，専門的な知識の共有だけでなく，試行錯誤の実験によって得られる知見の共有も実現可能であることを示した．またパラメータの変動を行う機能により，研究者らの試行錯誤する過程をワークフロー化し高効率化を目指した．

しかしながら問題点もまだ残されている．本実装においてユーザインターフェイスはコンソールベースであり，利便性については向上しているとは言えない．ワークフローを組み立てるにあたり，ユーザ利便性を考慮した Graphical Use Interface (GUI) が必要不可欠となる．さらに GUI で組み立てるワークフローの記号化なども課題として挙げられる．ワークフローについては様々な研究がなされており，ワークフローの標準化団体である Workflow Management Coalition (WfMC)[6] では，XML をベースにしたワークフロー記述言語 XML Process Definition Language(XPDL) の標準化を提案している．XPDL は Web サービスを想定したビジネスシーンでのワークフロー記述言語であるが，ワークフローによるプロジェクト／タスク管理など[7]，様々な利用想定がなされており汎用性を示している．XML 関連の標準化団体である Organization for the Advancement of Structured Information Standards (OASIS)[7] もワークフロー記述言語として Business Process Execution Language (BPEL)[8] を提案している．

またドッキングシミュレーションでは，実行するパラメータの調整幅に伴いワークフローの並列度の上昇し，計算量が要求されると予測できる．しかしながらワークフローの数に比例した計算量の増加であるので，分散並列処理に対応することで解決するものと考えている．近年，分散並列処理において注目されている技術としては，グリッドコンピューティングが挙げられる．グリッドコンピューティングのミドルウェア，Globus では，多くの計算機を資源として接続しひとつの巨大な計算機として見せるべく仮想化について研究が進められている．グリッドコンピューティングでは計算量の問題，データ量の問題などに着手し成果をあげている．さらにグリッドコンピューティングを用いたワークフローフレームワークの開発も行っている[14]．またハイスループットコンピューティングの実装である Condor[9] も注目されている．Condor は分散並列処理に対しジョブの投入，管理，スケジューリングを行うミドルウェアであり，グリッドコンピューティングと統合する動きもある[15]．こうした分散並列処理の実行環境に対応することも課題として挙げられる．

上述したように，ワークフローに関する研究は現在も様々な分野において進められている．つまり各分野において現在，ワークフローが求められているとも言える．これら関連研究の動向を踏まえ，ライフサイエンス研究の効率化，合理化を進めることに今後も取り組んでいきたい．

参 考 文 献

1) DOCK, http://dock.compbio.ucsf.edu
2) AutoDock, http://www.scripps.edu/mb/olson/doc/autodock/
3) GOLD, http://www.ccdc.cam.ac.uk/products/life_sciences/gold/
4) presto X, http://www.jbic.or.jp/presto_x/index_px.html
5) The Globus Alliance, http://www.globus.org
6) Workflow Management Coalition, http://www.wfmc.org
7) Organization for the Advancement of Structured Information Standards,

http://www.oasis-open.org/
8) Business Process Execution Language, http://dev2dev.bea.com/webservices/bpel/
9) Thain, D., Tannenbaum, T. and Livny, M.: Distributed Computing in Practice: The Condor Experience, *Concurrency and Computation: Practice and Experience*, (2004)
10) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *International Jounal of Supercomputer Applications*, Vol.11, No.2, pp. 115–128 (1997).
11) Bissantz, C., Folkers, G. and Rognan, D.: Protein-Based Virtual Screening of Chemical Databases. 1. Evaluation of Different Docking/Scoring Combinations, *Journal of Medicinal Chemistry* , 43, pp. 4759–4767 (2000).
12) Kakumoto, K., Yamanaka, S., Hamada, C. and Yoshimura, I.: A Statistical Analysis of an Effective method to Conduct In Silico Screening for Active Compounds, *Chem-Bio Informatics Journal* , Vol. 4, No. 4, pp. 121–132 (2004).
13) Buyya, R., Branson, K., Giddy, J. and Abramson, D.: The Virtual Laboratory: a toolset to enable distributed molecular modelling for drug design on the World-Wide Grid, *Journal of Concurrency and Computation: Practice and Experience (CCPE)* , 15, pp. 1–25 (2003).
14) Junwei Cao, Jarvis, S.A., Saini, S. and Nudd, G.R.:GridFlow: workflow management for grid computing , *Proceedings of Cluster Computing and the Grid (CCGrid)* , pp. 198–205 (2003).
15) Wolski, R., Brevik, J., Obertelli, G., Spring, N. and Su, A.: Writing programs that run EveryWare on the Computational Grid, *Parallel and Distributed Systems, IEEE Transactions* , Vol. 12, Issue. 10, pp. 1066–1080 (2001)