(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2020-155102
(P2020-155102A)
(43)公開日　令和2年9月24日(2020.9.24)

(51)Int.Cl.		F I			テーマコード（参考）
G05B 19/05	(2006.01)	G05B	19/05	A	5H220

審査請求 未請求　請求項の数 28　ＯＬ　外国語出願　（全 49 頁）

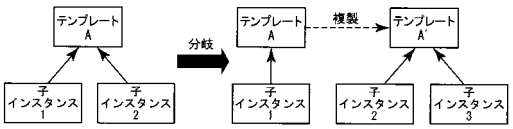
(21)出願番号　特願2020-12441 (P2020-12441)
(22)出願日　令和2年1月29日 (2020.1.29)
(31)優先権主張番号　16/359,014
(32)優先日　平成31年3月20日 (2019.3.20)
(33)優先権主張国・地域又は機関
米国 (US)

(71)出願人　000006507
横河電機株式会社
東京都武蔵野市中町２丁目９番３２号
(74)代理人　100106909
弁理士　棚井　澄雄
(74)代理人　100146835
弁理士　佐伯　義文
(74)代理人　100167553
弁理士　高橋　久典
(74)代理人　100181124
弁理士　沖田　壮男

最終頁に続く

(54)【発明の名称】工業制御システムについてのエンジニアリングデータを作成する方法およびシステム

(57)【要約】　（修正有）
【課題】工業プラントモジュールベースエンジニアリング方法を提供する。
【解決手段】プロセスは、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートＡを複製して、インスタンス１を複製せずに1つまたは複数の複製子テンプレート２、３を有する複製エンジニアリングテンプレートＡ’を作成するためのプロセスである。
【選択図】図3



【特許請求の範囲】
【請求項１】
工業プラントモジュールベースエンジニアリング方法であって、
1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートを複製して、前記ソースエンジニアリングテンプレートにおける前記1つまたは複数のソース子テンプレートからインスタンス化された1つまたは複数の子インスタンスを含むインスタンスを複製することなしに、それぞれ、1つまたは複数のソース子テンプレートに対応する1つまたは複数の複製子テンプレートを有する複製エンジニアリングテンプレートを作成するステップであって、前記複製エンジニアリングテンプレートがインスタンスを有さず、前記1つまたは複数の子インスタンスの各々が、前記ソースエンジニアリングテンプレートへの元のリンクを有する、ステップと、
前記ソースエンジニアリングテンプレートからインスタンス化された1つまたは複数の子インスタンスから少なくとも1つの子インスタンスを選択するステップであって、前記選択された少なくとも1つの子インスタンスが、前記ソースエンジニアリングテンプレートへの元のリンクを有する、ステップと、
前記元のリンクを前記選択された少なくとも1つの子インスタンスと前記複製エンジニアリングテンプレートとの間の新しいリンクに変更するステップであって、前記選択された少なくとも1つの子インスタンスが、前記複製エンジニアリングテンプレートへの前記新しいリンクを有し、前記ソースエンジニアリングテンプレートへの前記元のリンクを有さず、選択されない1つまたは複数の子インスタンスが、前記ソースエンジニアリングテンプレートへの前記元のリンクを保持する、ステップとを含む工業プラントモジュールベースエンジニアリング方法。
【請求項２】
前記1つまたは複数のソース子テンプレートを有する前記ソースエンジニアリングテンプレートをインスタンス化して、前記1つまたは複数のソース子テンプレートを有する前記ソースエンジニアリングテンプレートを複製する前に各インスタンスが1つまたは複数の子インスタンスを有する1つまたは複数のインスタンスを作成する、請求項1に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項３】
前記1つまたは複数のソース子テンプレートを有する前記ソースエンジニアリングテンプレートは、ソースクラスモジュールへのリンクを有する1つまたは複数のソース子クラスモジュールを有するソースグループクラスモジュールを備え、
前記1つまたは複数の子インスタンスは、前記ソースエンジニアリングテンプレートからインスタンス化されており、前記ソースクラスモジュールへのリンクを有する1つまたは複数の子アプリケーションモジュールを備える、請求項2に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項４】
前記ソースクラスモジュールをインスタンス化して、前記ソースクラスモジュールへのリンクを有する1つまたは複数のクラスベースアプリケーションモジュールを作成するステップをさらに含む、請求項3に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項５】
前記ソースクラスモジュールを複製して複製クラスモジュールを作成するステップをさらに含む、請求項4に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項６】
前記ソースクラスモジュールを複製して複製クラスモジュールを作成した後、前記1つまたは複数のクラスベースアプリケーションモジュールのうちの選択されたクラスベースアプリケーションモジュールと前記ソースクラスモジュールとの間の前記リンクを前記選択されたクラスベースアプリケーションモジュールと前記複製クラスモジュールとの間のリンクに変更するステップをさらに含む、請求項5に記載の工業プラントモジュールベー

スエンジニアリング方法。
【請求項７】
前記ソースクラスモジュールを複製して複製クラスモジュールを作成した後、前記1つまたは複数の子アプリケーションモジュールのうちの選択された子アプリケーションモジュールと前記ソースクラスモジュールとの間の前記リンクを前記選択された子アプリケーションモジュールと前記複製クラスモジュールとの間のリンクに変更するステップをさらに含む、請求項5に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項８】
ソースクラスモジュールへのリンクを有する1つまたは複数のソース子クラスモジュールを有する前記ソースグループクラスモジュールを複製して、前記ソースクラスモジュールへのリンクを有する前記1つまたは複数の複製子クラスモジュールを有する複製グループクラスモジュールを作成するステップをさらに含む、請求項5に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項９】
前記1つまたは複数のソース子クラスモジュールを有する前記ソースグループクラスモジュールをインスタンス化して、前記ソースグループクラスモジュールへのリンクを有する1つまたは複数のグループアプリケーションモジュールを作成するステップであって、前記1つまたは複数のグループアプリケーションモジュールが前記ソースクラスモジュールへの1つまたは複数の子アプリケーションモジュールを有する、ステップをさらに含む、請求項8に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１０】
前記ソースグループクラスモジュールをインスタンス化して前記1つまたは複数のグループアプリケーションモジュールを作成した後、前記1つまたは複数のグループアプリケーションモジュールのうちの選択されたグループアプリケーションモジュールと前記ソースグループクラスモジュールとの間の前記リンクを前記選択されたグループアプリケーションモジュールと前記複製グループクラスモジュールとの間のリンクに変更するステップをさらに含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１１】
前記選択されたグループアプリケーションモジュールと前記ソースグループクラスモジュールとの間の前記リンクを前記選択されたグループアプリケーションモジュールと前記複製グループクラスモジュールとの間の前記リンクに変更した後、前記選択されたグループアプリケーションモジュールの前記1つまたは複数の子アプリケーションモジュールのうちの選択された子アプリケーションモジュールと前記ソース子クラスモジュールとの間の前記リンクを前記選択された子アプリケーションモジュールと前記複製子クラスモジュールとの間のリンクに変更するステップをさらに含む、請求項10に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１２】
前記ソースクラスモジュールを複製して複製クラスモジュールを作成した後、前記1つまたは複数のクラスベースアプリケーションモジュールのうちの選択されたクラスベースアプリケーションモジュールと前記ソースクラスモジュールとの間の前記リンクを前記選択されたクラスベースアプリケーションモジュールと前記複製クラスモジュールとの間のリンクに変更するステップをさらに含む、請求項11に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１３】
前記ソースクラスモジュールを複製して前記複製クラスモジュールを作成した後、前記1つまたは複数の子アプリケーションモジュールのうちの選択された子アプリケーションモジュールと前記ソースクラスモジュールとの間の前記リンクを前記選択された子アプリケーションモジュールと前記複製クラスモジュールとの間のリンクに変更するステップをさらに含む、請求項12に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１４】

各グループクラスモジュールは、
i)複数のクラスモジュールのグループ化、および
ii)複数の論理制御モジュール図面の複合ループを定義するための複数の図面へ前記クラスモジュールの割当てを定義する少なくとも1つの割振りを含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１５】
前記ソースグループクラスモジュールをインスタンス化する前記ステップは、
前記ソースグループクラスモジュールの階層を反映することによって前記ソースグループクラスモジュールをインスタンス化して、アプリケーション構造ナビゲータにおいて、前記グループクラスモジュールからグループアプリケーションモジュールおよび前記グループアプリケーションモジュールの階層を生成するステップと、
子クラスモジュールをインスタンス化して、前記アプリケーション構造ナビゲータにおいて、前記子クラスモジュールから子アプリケーションモジュールを生成するステップとを含む、請求項14に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１６】
各グループクラスモジュールの前記割振りおよび各グループクラスモジュールの前記複合ループのそれぞれのトポロジーを更新するステップであって、グループモジュール更新エンジンが、各グループアプリケーションモジュールの前記割振りおよび各グループアプリケーションモジュールの前記複合ループのそれぞれのトポロジーを更新するように構成される、ステップをさらに含む、請求項15に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１７】
前記グループクラスモジュールの少なくとも1つの割振りに基づいて、前記グループアプリケーションモジュールおよびレガシーアプリケーションモジュール(レガシーAPM)を工業プラントにおけるフィールドコントローラの図面にバインドするステップであって、前記レガシーアプリケーションモジュールが、iii)ソースクラスモジュールを参照するクラスベースアプリケーションモジュール、およびiv)ソースクラスモジュールを参照しないクラスレスアプリケーションモジュールのうちの少なくとも一方である、ステップをさらに含む、請求項16に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１８】
各グループクラスモジュールは、複数の子クラスモジュールと、フォルダと、前記子クラスモジュールおよび前記フォルダの階層とを有する、請求項17に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項１９】
各グループクラスモジュールは、
モジュール設計の文書を生成するために使用される設計文書情報と、
前記クラスモジュールに関するアーチファクトを記憶するために使用されるアタッチメントと、
クラスモジュールのグループ化および複数の制御図面への前記クラスモジュールの割当てを定義してフィールド制御システムの図面についての複合ループを定義する前記割振りとを備える、請求項18に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項２０】
前記クラスモジュールは、
モジュール設計の文書を生成するために使用される設計文書情報と、
前記クラスモジュールに関するアーチファクトを記憶するために使用されるアタッチメントと、
プロセス制御の論理情報を含む制御論理と、
アラーム属性と、
チューニングパラメータとを備える、請求項19に記載の工業プラントモジュールベースエンジニアリング方法。

【請求項２１】
子クラスモジュールにおいてインスタンス化すべきでない部分を選択して、単一のグループクラスモジュールからそれぞれに異なるグループアプリケーションモジュールのセットを生成するステップをさらに含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項２２】
各グループクラスモジュールにおいて定義される複数の論理制御図面を複数の物理フィールド制御システムのそれぞれに割り当てるステップをさらに含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項２３】
グループモジュール更新エンジンが、グループクラスモジュールの複合ループのそれぞれのトポロジーを更新する場合に、前記グループクラスモジュールからインスタンス化された前記グループアプリケーションモジュールを参照する前記子アプリケーションモジュールの追加および削除の少なくとも一方を実行するステップをさらに含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項２４】
グループモジュール更新エンジンが、グループクラスモジュールの複合ループのそれぞれのトポロジーを更新する場合に、前記グループクラスモジュールからインスタンス化された前記グループアプリケーションモジュールを参照する前記子アプリケーションモジュールのコンテンツを更新するステップをさらに含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項２５】
グループモジュール更新エンジンが、各グループアプリケーションモジュールの複合ループのそれぞれのトポロジーを更新する場合に、グループクラスモジュールからインスタンス化された前記グループアプリケーションモジュールを参照する前記子アプリケーションモジュールの追加および削除の少なくとも一方を実行するステップをさらに含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項２６】
グループモジュール更新エンジンが、各グループアプリケーションモジュールの複合ループのそれぞれのトポロジーを更新する場合に、グループクラスモジュールからインスタンス化された前記グループアプリケーションモジュールを参照する前記子アプリケーションモジュールのコンテンツを更新するステップをさらに含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項２７】
各グループアプリケーションモジュールは、
iii)アプリケーションモジュールのグループ化、および
iv)複数の制御図面への前記アプリケーションモジュールの割当てを定義して複数の論理制御図面の複合ループを定義する少なくとも1つの割振りを含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【請求項２８】
グループモジュール更新エンジンが、各グループクラスモジュールの割振りおよび各グループクラスモジュールの複合ループのそれぞれのトポロジーを更新した場合に、各グループアプリケーションモジュールの前記割振りおよび各グループアプリケーションモジュールの前記複合ループの前記それぞれのトポロジーを更新するステップをさらに含む、請求項9に記載の工業プラントモジュールベースエンジニアリング方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明の実施形態は、概してプロセス制御システムを構成または設計するための方法およびシステムに関し、より詳細には、工業プラントのプロセス制御システムについてのエ

ンジニアリングデータを効率的に作成し修正するための方法およびシステムに関する。
【背景技術】
【０００２】
[モジュールベースエンジニアリングの概要]
工業プラントにおいて、モジュールベースエンジニアリングは、プラント器械、安全器械、および維持管理を含む、全体的なプラント制御システムを構成し維持することによるオートメーション設計に有用である。一般に、オートメーションエンジニアリングシステムのサーバは、エンジニアリングデータのデータベースを集中的に管理して、プラント制御システムを拡張、修正、または維持するために設計情報を利用可能にし、設計情報とプラント制御システムに記憶された実際の情報との間の不一致を修正するための不要な人的資源を節約する。モジュールベースエンジニアリングは、制御論理および設計情報をモジュールに変換し、次いでオートメーションエンジニアリングシステムのサーバにおいてモジュール同士を組み合わせることによって制御アプリケーションおよびアラームを設計するためのエンジニアリング方法を指す。モジュールは一般に、過去の設計パターン経験から収集された顧客情報およびノウハウなどの独立したソフトウェア構成要素からなることがあり、制御論理、アラーム属性、および設計情報を含むこともある。以前のプロジェクトにおいて構成されたモジュールを再使用すると、エンジニアリング品質を向上させ、エンジニアリング時間を短縮することができ、このことはプロジェクト期間を短縮することに寄与する。図1は、オートメーション設計についてのモジュールベースエンジニアリングの一般的な概念の概略図である。図1に示すように、モジュールは、オートメーションエンジニアリングシステムのエンジニアリングツールによって構成し、オートメーションエンジニアリングシステムのサーバに登録し、かつオートメーションエンジニアリングのサーバからダウンロードすることができる。エンジニアリング設計情報は、エンジニアリング結果の要約文書を作成するために保存される。さらに、モジュールベースエンジニアリングは、以下のエンジニアリングタスク、すなわち入出力設計、制御アプリケーション設計、およびシステム構成設計と同時に実行することができる。
【０００３】
モジュールベースエンジニアリングでは、制御論理、アラーム属性、設計情報、およびアタッチメントがモジュールと見なされる。モジュールは一般に、設計情報、制御論理、チューニングパラメータ、アラーム属性、およびアタッチメントを含むことがある。機能仕様などの設計情報は、モジュール構成要素として定義されることがある。設計情報は一般に、モジュールの詳細について説明するテキスト、画像、および表を含むことがある。制御論理は、制御図面、ならびに機能ブロック、スイッチ、およびメッセージの詳細な定義を含むことがある。制御論理は、クラスモジュールまたはアプリケーションモジュールにおいて定義されることがある。モジュールベースエンジニアリングは、フィールド制御システムの制御論理に定義された機能ブロックにおけるチューニングパラメータ設計値のバルク編集により、かつフィールド制御システムのチューニングパラメータ設計値および現在の値を比較し設定することによって、チューニングパラメータをモジュール構成要素として見なすのを可能にすることがある。フィールド制御システムは、ハードウェア入出力コントローラであってもよい。アラーム属性は、アラーム設定値およびアラーム優先順位であってもよい。任意のファイルをモジュール構成要素として付加することができる。アタッチメントのリストは単純な動作によって起動することができる。
【０００４】
モジュールベースエンジニアリングには2種類のモジュール、たとえばクラスモジュールおよびアプリケーションモジュールが利用可能である。クラスモジュールは、制御アプリケーションについてのテンプレートとして使用され、アプリケーションモジュールは実際の制御アプリケーションとして働く。クラスモジュールはテンプレートである。クラスモジュールに基づいて、実際の制御アプリケーションを実行するアプリケーションモジュールを作成することができる。アプリケーションモジュールは、テンプレートとして使用されるクラスモジュールとの関係を維持し、クラスモジュールに対する変更はアプリケー

ションモジュールに反映される。単一のモジュールが複数のアプリケーションモジュールについてのテンプレートとして使用される。アプリケーションモジュールは、入出力およびタグ名をアプリケーションモジュールに割り当てることによって制御アプリケーションを実行する。2種類のアプリケーションモジュール、たとえば、クラスモジュールに基づいて作成されるクラスベースアプリケーションモジュール、およびクラスモジュールを使用せずに作成されるクラスレスアプリケーションモジュールが利用可能である。

【０００５】

モジュールベースエンジニアリングは、入出力、制御アプリケーション、およびシステム構成を並行して設計するのを可能にすることがあり、それによってシステム構成を仕上げる前に制御アプリケーションおよび入出力の設計を開始するのを可能にする。制御アプリケーションを設計した後でも、入出力設計が柔軟に変更されてもよい。プラント情報は、制御システムエンジニアリングを実施するための様々な種類の情報を含む。モジュールベースエンジニアリングでは、入出力は、プラントの取得された入出力情報に基づいて設計されてもよい。一般に、オートメーションエンジニアリングシステムのエンジニアリングツールは、入出力を設計して、プラント入出力情報を表形式の入出力情報リストとして構成するために使用されてもよい。入出力情報リストは、入出力タグ名、入出力モジュールタイプ、および入出力モジュールが装備されるFCSステーション名、ならびに各入出力の特定の情報などの情報を定義する。入出力情報リストに対してエクスポートおよび/またはインポートを行うことができる。入出力情報リストの設定情報は、エクスポートされた外部ファイル上で編集されてもよく、設定情報は、オートメーションエンジニアリングシステムのエンジニアリングツールにインポートされてもよい。まず必要に応じてクラスモジュールが作成される。次いで、クラスモジュールを使用するかまたはクラスモジュールを使用せずにアプリケーションモジュールが作成される。オートメーションエンジニアリングシステムのエンジニアリングツールは、制御アプリケーション設計のエンジニアリングに使用される。アプリケーションモジュールの入出力端末に入出力タグ名が与えられる。アプリケーションモジュールの機能ブロックに実際のタグ名が適用される。入出力タグ名に基づいて制御アプリケーションが作成されるので、制御アプリケーションは、入出力モジュール割当て情報または各入出力の特定の情報などの入出力設計が完了する前に作成することができる。

【０００６】

入出力アプリケーションおよび制御アプリケーションを設計すること以外の項目のエンジニアリングは、フィールド制御ステーション(FCS)およびヒューマンインターフェースステーション(HIS)などのシステム構成ならびにプロジェクトに共通し各ステーションに関連する項目を設計することによって行われる。これらの項目は、システムビューによって構成される。オートメーションエンジニアリングシステムのエンジニアリングツールは、コモンスイッチ、グローバルスイッチ、アナンシエータ、信号イベント、オペレータ案内メッセージ、および印刷メッセージなどのスイッチを設定するために使用される。システム構成とは独立にオートメーションエンジニアリングシステムのエンジニアリングツールにおいて作成される入出力設計および制御アプリケーション設計のエンジニアリングデータは最終的に、フィールド制御ステーション(FCS)に割り当てられ、任意のプロジェクトデータとして生成される。プロセス入出力、シリアル通信イーサネット(登録商標)通信についてのエンジニアリングは、プロジェクトデータが完成するときに完了する。

【０００７】

プロジェクトデータを完成することによってモジュールベースエンジニアリングが終了すると、試験機能の定義済みのセットを使用することによって制御アプリケーション試験が行われる。モジュールベースエンジニアリングをサポートする機能は、文書生成機能、バルク編集機能、チューニングパラメータ管理機能を含む。文書生成機能は、モジュールの設計情報と様々なエンジニアリングデータを統合して単一の文書ファイルを生成する機能である。チューニングパラメータ管理機能は、制御アプリケーションを作成する際に設計される機能ブロックチューニングパラメータ値およびフィールド制御システムの現在の

チューニングパラメータ値を管理するための機能である。バルク編集機能は、制御アプリケーションを構成する間に設計されるモジュールの制御論理およびアラーム属性を集合的に編集するための機能である。

【発明の概要】

【課題を解決するための手段】

【０００８】

いくつかの態様において、いくつかの実施形態では、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースエンジニアリングテンプレートを複製するステップと、少なくとも1つの子インスタンスを選択するステップと、元のリンクを新しいリンクに変更するステップとを含んでもよい。ソースエンジニアリングテンプレートを複製するためのプロセスは、限定はしないが、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートを複製して、ソースエンジニアリングテンプレートにおける1つまたは複数のソース子テンプレートからインスタンス化された1つまたは複数の子インスタンスを含むインスタンスを複製することなしに、それぞれ、1つまたは複数のソース子テンプレートに対応する1つまたは複数の複製子テンプレートを有する複製エンジニアリングテンプレートを作成するステップであって、複製エンジニアリングテンプレートがインスタンスを有さず、1つまたは複数の子インスタンスの各々が、ソースエンジニアリングテンプレートへの元のリンクを有する、ステップを含んでもよい。少なくとも1つの子インスタンスを選択するためのプロセスは、限定はしないが、ソースエンジニアリングテンプレートからインスタンス化された1つまたは複数の子インスタンスから少なくとも1つの子インスタンスを選択するステップであって、選択された少なくとも1つの子インスタンスが、ソースエンジニアリングテンプレートへの元のリンクを有する、ステップを含んでもよい。元のリンクを新しいリンクに変更するためのプロセスは、限定はしないが、元のリンクを選択された少なくとも1つの子インスタンスと複製エンジニアリングテンプレートとの間の新しいリンクに変更するステップであって、選択された少なくとも1つの子インスタンスが、複製エンジニアリングテンプレートへの新しいリンクを有し、ソースエンジニアリングテンプレートへの元のリンクを有さず、選択されない1つまたは複数の子インスタンスが、ソースエンジニアリングテンプレートへの元のリンクを保持する、ステップを含んでもよい。

【図面の簡単な説明】

【０００９】

【図１】オートメーション設計のためのモジュールベースエンジニアリングの一般的な概念の概略図である。

【図２Ａ】関連技術における、ソーステンプレートモジュールとソーステンプレートモジュールを参照し/ソーステンプレートモジュールに関連付けられた複数の子インスタンスとの間の関係の一般的な概念の概略図である。

【図２Ｂ】ソーステンプレートモジュールとしてのクラスモジュールと、ソーステンプレートモジュールとしてのクラスモジュールに共通的にリンクするかまたはクラスモジュールを参照する複数のクラスベースアプリケーションモジュールとの間の関係の一般的な概念の概略図である。

【図３】ソースエンジニアリングテンプレートをインスタンス化した後にソースエンジニアリングテンプレートを分岐させるための分岐プロセスの全体的な概念の例示的な例の概略図である。

【図４】グループクラスモジュールを分岐させるためのプロセスを示す概略図である。

【図５】図4のグループクラスモジュールを分岐させるためのプロセスの後にレガシークラスモジュールを分岐させるためのプロセスを示す概略図である。

【図６】クラスモジュールを分岐させるためのプロセスを示す概略図である。

【図７】グループクラスモジュールベースエンジニアリングに基づいて階層化アーキテクチャとして編成されたプラント制御システムにおける様々な機器を参照する工業プラントの階層を示すブロック図である。

【図８】制御モジュールレベル、機器モジュールレベル、および/またはユニットレベルで再使用可能なグループクラスモジュールを使用するグループモジュールベースエンジニアリングシステムを含むプラントエンジニアリングシステムを示すブロック図である。
【図９】グループクラスモジュール、およびグループクラスモジュールから部分的にインスタンス化された2つのグループアプリケーションモジュールの例を示す図である。
【図１０】グループクラスモジュールおよびグループアプリケーションモジュールのための更新管理の例を示す図である。
【図１１】グループアプリケーションモジュールのためのモジュール更新のサンプルシナリオを示す図である。
【発明を実施するための形態】
【００１０】
いくつかの実施形態では、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースエンジニアリングテンプレートを複製するステップと、少なくとも1つの子インスタンスを選択するステップと、元のリンクを新しいリンクに変更するステップとを含んでもよい。ソースエンジニアリングテンプレートを複製するためのプロセスは、限定はしないが、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートを複製して、ソースエンジニアリングテンプレートにおける1つまたは複数のソース子テンプレートからインスタンス化された1つまたは複数の子インスタンスを含むインスタンスを複製するステップなしに、それぞれ、1つまたは複数のソース子テンプレートに対応する1つまたは複数の複製子テンプレートを有する複製エンジニアリングテンプレートを作成するステップを含み、複製エンジニアリングテンプレートはインスタンスを有さず、1つまたは複数の子インスタンスの各々は、ソースエンジニアリングテンプレートへの元のリンクを有する。少なくとも1つの子インスタンスを選択するためのプロセスは、限定はしないが、ソースエンジニアリングテンプレートからインスタンス化された1つまたは複数の子インスタンスから少なくとも1つの子インスタンスを選択するステップであって、選択された少なくとも1つの子インスタンスが、ソースエンジニアリングテンプレートへの元のリンクを有する、選択するステップを含んでもよい。元のリンクを新しいリンクに変更するためのプロセスは、限定はしないが、元のリンクを選択された少なくとも1つの子インスタンスと複製エンジニアリングテンプレートとの間の新しいリンクに変更するステップであって、選択された少なくとも1つの子インスタンスが、複製エンジニアリングテンプレートへの新しいリンクを有し、ソースエンジニアリングテンプレートへの元のリンクを有さず、選択されない1つまたは複数の子インスタンスが、ソースエンジニアリングテンプレートへの元のリンクを保持する、変更するステップを含んでもよい。
【００１１】
場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートをインスタンス化して、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートを複製する前に1つまたは複数の子インスタンスを有する1つまたは複数のインスタンスを作成するステップをさらに含んでもよい。
【００１２】
場合によっては、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートは、限定はしないが、ソースクラスモジュールへのリンクを有する1つまたは複数のソース子クラスモジュールを有するソースグループクラスモジュールを含んでもよく、1つまたは複数の子インスタンスは、ソースエンジニアリングテンプレートからインスタンス化されており、限定はしないが、ソースクラスモジュールへのリンクを有する1つまたは複数の子アプリケーションモジュールを含んでもよい。
【００１３】
場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースクラスモジュールをインスタンス化して、ソースクラスモジュールへのリン

クを有する1つまたは複数のクラスベースアプリケーションモジュールを作成するステップをさらに含んでもよい。

【００１４】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースクラスモジュールを複製して複製クラスモジュールを作成するステップをさらに含んでもよい。

【００１５】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースクラスモジュールを複製して複製クラスモジュールを作成した後、1つまたは複数のクラスベースアプリケーションモジュールのうちの選択されたクラスベースアプリケーションモジュールとソースクラスモジュールとの間のリンクを選択されたクラスベースアプリケーションモジュールと複製クラスモジュールとの間のリンクに変更するステップをさらに含んでもよい。

【００１６】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースクラスモジュールを複製して複製クラスモジュールを作成した後、1つまたは複数の子アプリケーションモジュールのうちの選択された子アプリケーションモジュールとソースクラスモジュールとの間のリンクを選択された子アプリケーションモジュールと複製クラスモジュールとの間のリンクに変更するステップをさらに含んでもよい。

【００１７】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、ソースクラスモジュールへのリンクを有する1つまたは複数のソース子クラスモジュールを有するソースグループクラスモジュールを複製して、ソースクラスモジュールへのリンクを有する1つまたは複数のソース子クラスモジュールを有する複製グループクラスモジュールを作成するステップをさらに含んでもよい。

【００１８】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、1つまたは複数のソース子クラスモジュールを有するソースグループクラスモジュールをインスタンス化して、ソースグループクラスモジュールへのリンクを有する1つまたは複数のグループアプリケーションモジュールを作成するステップであって、1つまたは複数のグループ子アプリケーションモジュールが、ソースクラスモジュールへの1つまたは複数の子アプリケーションモジュールリンクを有する、ステップをさらに含んでもよい。

【００１９】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースグループクラスモジュールをインスタンス化して1つまたは複数のグループアプリケーションモジュールを作成した後、1つまたは複数のグループアプリケーションモジュールのうちの選択されたグループアプリケーションモジュールとソースグループクラスモジュールとの間のリンクを選択されたグループアプリケーションモジュールと複製グループクラスモジュールとの間のリンクに変更するステップをさらに含んでもよい。

【００２０】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、選択されたグループアプリケーションモジュールとソースグループクラスモジュールとの間のリンクを選択されたグループアプリケーションモジュールと複製グループクラスモジュールとの間のリンクに変更した後、選択されたグループアプリケーションモジュールの1つまたは複数の子アプリケーションモジュールのうちの選択された子アプリケーションモジュールとソース子クラスモジュールとの間のリンクを選択された子アプリケーションモジュールと複製子クラスモジュールとの間のリンクに変更するステップをさらに含んでもよい。

【００２１】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースクラスモジュールを複製して複製クラスモジュールを作成した後、1つまたは複数のクラスベースアプリケーションモジュールのうちの選択されたクラスベースアプリケーションモジュールとソースクラスモジュールとの間のリンクを選択されたクラスベースアプリケーションモジュールと複製クラスモジュールとの間のリンクに変更するステップをさらに含んでもよい。

【００２２】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、ソースクラスモジュールを複製して複製クラスモジュールを作成した後、1つまたは複数の子アプリケーションモジュールのうちの選択された子アプリケーションモジュールとソースクラスモジュールとの間のリンクを選択された子アプリケーションモジュールと複製クラスモジュールとの間のリンクに変更するステップをさらに含んでもよい。

【００２３】

場合によっては、各グループクラスモジュールは、i)複数のクラスモジュールのグループ化、およびii)複数の図面へのクラスモジュールの割当てを定義して複数の論理制御モジュール図面の複合ループを定義する少なくとも1つの割振りを含む。

【００２４】

場合によっては、ソースグループクラスモジュールをインスタンス化するステップは、限定はしないが、ソースグループクラスモジュールの階層を反映することによってソースグループクラスモジュールをインスタンス化して、アプリケーション構造ナビゲータにおいて、グループクラスモジュールからグループアプリケーションモジュールおよびグループアプリケーションモジュールの階層を生成するステップと、子クラスモジュールをインスタンス化して、アプリケーション構造ナビゲータにおいて、子クラスモジュールから子アプリケーションモジュールを生成するステップとを含んでもよい。

【００２５】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、各グループクラスモジュールの割振りおよび各グループクラスモジュールの複合ループのそれぞれのトポロジーを更新するステップであって、グループモジュール更新エンジンが、各グループアプリケーションモジュールの割振りおよび各グループアプリケーションモジュールの複合ループのそれぞれのトポロジーを更新するように構成される、ステップをさらに含んでもよい。

【００２６】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、グループクラスモジュールの少なくとも1つの割振りに基づいて、グループアプリケーションモジュールおよびレガシーアプリケーションモジュール(レガシーAPM)を工業プラントにおけるフィールドコントローラの図面にバインドするステップであって、レガシーアプリケーションモジュールが、iii)ソースクラスモジュールを参照するクラスベースアプリケーションモジュール、およびiv)ソースクラスモジュールを参照しないクラスレスアプリケーションモジュールのうちの少なくとも一方である、バインドするステップをさらに含んでもよい。

【００２７】

場合によっては、各グループクラスモジュールは、複数の子クラスモジュールと、フォルダと、子クラスモジュールおよびフォルダの階層とを有する。

【００２８】

場合によっては、各グループクラスモジュールは、限定はしないが、モジュール設計の文書を生成するために使用される設計文書情報と、クラスモジュールに関するアーチファクトを記憶するために使用されるアタッチメントと、クラスモジュールのグループ化および複数の制御図面へのクラスモジュールの割当てを定義してフィールド制御システムの図面についての複合ループを定義する割振りとをさらに含んでもよい。

【００２９】

場合によっては、クラスモジュールは、限定はしないが、モジュール設計の文書を生成するために使用される設計文書情報と、クラスモジュールに関するアーチファクトを記憶するために使用されるアタッチメントと、プロセス制御の論理情報を含む制御論理と、アラーム属性と、チューニングパラメータとをさらに含んでもよい。

【００３０】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、子クラスモジュールにおいてインスタンス化すべきでない部分を選択して、単一のグループクラスモジュールからそれぞれに異なるグループアプリケーションモジュールのセットを生成するステップをさらに含んでもよい。

【００３１】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、各グループクラスモジュールにおいて定義される複数の論理制御図面を複数の物理フィールド制御システムのそれぞれに割り当てるステップをさらに含んでもよい。

【００３２】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、グループモジュール更新エンジンが、グループクラスモジュールの複合ループのそれぞれのトポロジーを更新する場合に、グループクラスモジュールからインスタンス化されたグループアプリケーションモジュールを参照する子アプリケーションモジュールの追加および削除の少なくとも一方を実行するステップをさらに含んでもよい。

【００３３】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、グループモジュール更新エンジンが、グループクラスモジュールの複合ループのそれぞれのトポロジーを更新する場合に、グループクラスモジュールからインスタンス化されたグループアプリケーションモジュールを参照する子アプリケーションモジュールのコンテンツを更新するステップをさらに含んでもよい。

【００３４】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、グループモジュール更新エンジンが、各グループアプリケーションモジュールの複合ループのそれぞれのトポロジーを更新する場合に、グループクラスモジュールからインスタンス化されたグループアプリケーションモジュールを参照する子アプリケーションモジュールの追加および削除の少なくとも一方を実行するステップをさらに含んでもよい。

【００３５】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、グループモジュール更新エンジンが、各グループアプリケーションモジュールの複合ループのそれぞれのトポロジーを更新する場合に、グループクラスモジュールからインスタンス化されたグループアプリケーションモジュールを参照する子アプリケーションモジュールのコンテンツを更新するステップをさらに含んでもよい。

【００３６】

場合によっては、各グループアプリケーションモジュールは、iii)アプリケーションモジュールのグループ化、およびiv)複数の制御図面へのアプリケーションモジュールの割当てを定義して複数の論理制御図面の複合ループを定義する少なくとも1つの割振りを含む。

【００３７】

場合によっては、工業プラントモジュールベースエンジニアリング方法は、限定はしないが、グループモジュール更新エンジンが、各グループクラスモジュールの割振りおよび各グループクラスモジュールの複合ループのそれぞれのトポロジーを更新した場合に、各グループアプリケーションモジュールの割振りおよび各グループアプリケーションモジュールの複合ループのそれぞれのトポロジーを更新するステップをさらに含んでもよい。

【００３８】

以下の開示では、一方が通常クラスモジュールを使用し、他方がグループクラスモジュ

ールを使用する、2種類のモジュールベースエンジニアリング方法について説明する。通常クラスモジュールベースエンジニアリングの適用範囲は、階層における最低レベルを包含する。たとえば、通常クラスモジュールベースエンジニアリングの適用範囲は、制御モジュール図面の規則的な単純ループのみを包含し、制御モジュール図面、機器モジュール図面、および/またはユニット図面の複合ループを含まない。インスタンス化指向のクラスモジュールベースオブジェクトの特性および振る舞いに関する制御モジュール図面についてのクラスモジュールのグループ化は、適用範囲を拡張し、グループクラスモジュールベースエンジニアリングの適用範囲に最低階層レベルにおける制御モジュール図面の規則的な単純ループだけでなくより高い階層レベルにおける図面の複合ループも包含することを可能にする。グループクラスモジュールベースエンジニアリングの適用範囲は、通常クラスモジュールベースエンジニアリングの適用範囲よりも広い。

【００３９】

インスタンス化は、クラスの特定のインスタンスを作成するためのプロセスである。

【００４０】

クラスのインスタンス化はオブジェクトを生成するためのプロセスである。

【００４１】

オブジェクトはクラスのインスタンスである。オブジェクトはクラスをインスタンス化することによって生成される。

【００４２】

グループアプリケーションモジュール(グループAPM)は、グループクラスモジュールのインスタンスである。

【００４３】

子アプリケーションモジュール(子APM)は、クラスモジュールのショートカットまたは参照である子クラスモジュールのインスタンスである。

【００４４】

クラスは、オブジェクトを作成するための拡張可能なプログラムコードテンプレートであり、振る舞いの状態メンバー変数および処理系についての初期値を示す。クラスモジュールは再使用可能なテンプレートである。クラスモジュールは、エンジニアリング論理データと、アラーム属性と、チューニングパラメータとを含み、パラメータおよびモジュール規則についてのクラスモジュールのデフォルトセットを定義してインスタンス化時にパラメータに値を動的に設定する。クラスは、命令および特性または他のオブジェクトの参照のセットを含めるためのテンプレートでもある。クラスモジュールは、別のクラスの参照を含むことのできるクラスとは異なり、別のクラスモジュールを参照することはできない。

【００４５】

クラスモジュールのインスタンス化は、クラスベースアプリケーションモジュールを生成するためのプロセスである。

【００４６】

クラスベースアプリケーションモジュールはクラスモジュールのインスタンスである。

【００４７】

オブジェクトは、クラスをインスタンス化することによって生成され、変数、関数、およびデータ構造の組合せ、または名前もしくはキーおよび値の集合から構成されるデータタイプから構成される。

【００４８】

制御ループは、制御ループ内の別の変数の値を操作することによってシステム変数の所望の値を実現して維持するためにシステムとして協働する構成要素のグループである。各制御ループは、少なくとも入力と出力とを有する。2種類の制御ループ、開ループと閉ループがある。

【００４９】

グループは、クラスモジュール間の関係を定義するために論理構造またはトポロジーを

作成または形成するクラスについてのコンテナである。しかし、クラスとは異なり、グループモジュールは、部分的な構造によってインスタンス化され、一方、クラスは常にそのオブジェクトについての構造全体によってインスタンス化される。

【００５０】

グループクラスモジュールは、モジュールエンジニアリングの利点を拡張するのを可能にして、複数の制御図面の複合ループをグループ化することによってグループクラスモジュールの再使用可能性および利便性を向上させる。

【００５１】

[エンジニアリングテンプレートを分岐させないモジュールベースエンジニアリング]

工業用コントローラは、工業プロセスまたは製造環境を制御するために設計される。工業用コントローラは一般に、制御プログラムまたは制御論理を特定の工業プロセスについて一意に設計する必要がある。ユーザ/エンジニアは、コントローラの制御論理を変更または構成する場合、自動化エンジニアリングシステムの自動化設計オーガナイザなどのエディタツールを使用して工業プラントのコントローラのエンジニアリングデータまたは設計データを作成し修正することができる。実際には、制御論理は、工業プラントの様々な設計ニーズおよび動作要件を満たすために随時変更する必要がある。

【００５２】

この目的のために、一例として、エンジニアはエンジニアリングテンプレート(ソース親)を作成して使用し、本明細書におけるエンジニアリングテンプレートは、クラスモジュールとして理解することができ、クラスモジュールは一般に、設計情報の汎用制御論理を含む。さらに、インスタンスとしての複数のアプリケーションモジュールを、クラスモジュールとしてのエンジニアリングテンプレートに基づいて作成することができる。

【００５３】

エンジニアリングテンプレートおよびエンジニアリングデータは、後で試験されシステムプロジェクトに展開される。システムプロジェクトは、エンジニアリングデータが記憶されるデータのグループを指す。試験時には、追加の顧客の要件をサポートするためにエンジニアリングデータを変更することが必要になる場合がある。そのような要件に応じて、アプリケーションモジュールとしてのインスタンスを変更する必要がある。

【００５４】

アプリケーションモジュールとしてのインスタンスが変更された後、エンジニアリングテンプレートと子インスタンスとの間の関係またはリンクが破壊され、エンジニアリングテンプレートの再使用可能性はもはや利用可能ではなくなる。既存の慣行では、エンジニアおよび/またはユーザは、インスタンスに対する修正を実行する必要がある場合、各インスタンスに対する修正を個々に実行する前に、エンジニアリングテンプレートの、そのインスタンスとのリンクを解除する必要がある。これらのプロセスは、各インスタンスに対して1つずつ修正を実行する必要があるので、時間がかかりエラーが生じやすい。さらに、修正すべき各インスタンスとソースエンジニアリングテンプレートとの間のリンクが破壊されているので、インスタンスは、ソースエンジニアリングテンプレートを参照しないクラスレスアプリケーションモジュールになる。

【００５５】

図2Aは、関連技術における、ソーステンプレートモジュールとソーステンプレートモジュールを参照し/ソーステンプレートモジュールに関連付けられた複数の子インスタンスとの間の関係の一般的な概念の概略図である。図2Bは、ソーステンプレートモジュールとしてのクラスモジュール1(CM1)と、ソーステンプレートモジュールとしてのクラスモジュール1(CM1)に共通的にリンクするかまたはクラスモジュール1(CM1)を参照する複数のクラスベースアプリケーションモジュール1、2、3、および4(クラスベースAPM1、APM2、APM3、およびAPM4)との間の関係の一般的な概念の概略図である。

【００５６】

クラスモジュールは、エンジニアリングテンプレートとして理解することができ、複数のクラスベースアプリケーションモジュール/インスタンス1、2、3、および4(APM1、APM2

、APM3、およびAPM4)はクラスモジュールに基づいて作成することができる。一例として、複数のクラスベースアプリケーションモジュール/インスタンス1、2、3、および4(APM1、APM2、APM3、およびAPM4)はクラスモジュール1(CM1)に基づいてインスタンス化され、エンジニアがインスタンス化後に各子インスタンスに対する修正を実行するには、クラスモジュール1(CM1)と複数のクラスベースアプリケーションモジュール1、2、3、および4(APM1、APM2、APM3、およびAPM4)との間の関係またはリンクを破壊する必要がある。

【００５７】

複数のクラスベースアプリケーションモジュール/インスタンス1、2、3、および4(クラスベースAPM1、APM2、APM3、およびAPM4)のうちの一部またはすべてが変更された後、エンジニアリングテンプレートとしてのクラスモジュール1(CM1)とクラスベースアプリケーションモジュール1、2、3、および4(APM1、APM2、APM3、およびAPM4)における子インスタンスとの間の関係またはリンクは破壊され、関係またはリンクはもはや利用可能ではなくなる。したがって、クラスモジュール1(CM1)の再使用可能性は利用できなくなる。既存の慣行では、エンジニアまたはユーザは、インスタンスとしての複数のクラスベースアプリケーションモジュール1、2、3、および4(クラスベースAPM1、APM2、APM3、およびAPM4)のうちの一部またはすべてに対する修正を実行する必要がある場合、インスタンスとしての複数のクラスベースアプリケーションモジュール1、2、3、および4(クラスベースAPM1、APM2、APM3、およびAPM4)のうちの一部またはすべてからエンジニアリングテンプレートとしてのクラスモジュール1(CM1)をリンク解除する必要があり、それによって、複数のクラスベースアプリケーションモジュール1、2、3、および4(クラスベースAPM1、APM2、APM3、およびAPM4)のうちの一部またはすべては、各クラスレスインスタンスに対して個々に修正を実行する前に、複数のクラスレスアプリケーションモジュール1、2、3、および4(クラスレスAPM1、APM2、APM3、およびAPM4)になる。これらのプロセスは複雑で時間がかかり、エラーが生じやすい。

【００５８】

要するに、エンジニアは、一般に設計情報の汎用制御論理を含むソースエンジニアリングテンプレート(ソース親)を作成し、次いで、ソースエンジニアリングテンプレート(ソース親)に基づいてエンジニアリングデータの子インスタンスとしての変更および/または構成を行う。

【００５９】

ソースエンジニアリングテンプレートおよび子インスタンスとしてのエンジニアリングデータは、後で試験されシステムプロジェクトに展開される。試験時に、エンジニアリングデータは、顧客の追加の要件をサポートするように変更する必要がある。エンジニアは、そのような要件に応じて、通常、子インスタンスに対する変更を更新する必要がある。

【００６０】

その結果、テンプレートと子インスタンスとの間の関係が破壊され、エンジニアリングテンプレートの再使用可能性が無効になる。リンクが破壊された後、インスタンスは、ソーステンプレートを有さないクラスレスアプリケーションモジュールになり、エンジニアは、更新を個々に実行する必要がある。さらに、リンケージが破壊されると、既存の論理の一部も影響を受ける。

【００６１】

[エンジニアリングテンプレートを分岐させるクラスベース/グループクラスベースモジュールエンジニアリング]

再使用可能性およびエンジニアリング効率を改善するには、既存のエンジニアリングテンプレートまたはソースエンジニアリングテンプレートから1つまたは複数の新しいエンジニアリングテンプレート(新しい親)を分岐させるための分岐プロセスが有効である。エンジニアは、既存のテンプレートから新しいエンジニアリングテンプレート(新しい親)を分岐させてもよい。エンジニアは、分岐グループクラスモジュール対話を介したユーザの選択による分岐時にどのエンジニアリングデータを子インスタンスとして含めるべきかを選択することができる。分岐が成功した後、それぞれの子インスタンスは、新しい親を参

照し、それによって、エンジニアは、子インスタンス上で直接更新するのではなく新しいエンジニアリングテンプレート上で追加のデータを構成することができる。

【００６２】

具体的には、モジュールベースエンジニアリングでは、グループクラスモジュールをエンジニアリングテンプレートとして理解することができ、エンジニアリングテンプレートは、設計情報と、アタッチメントリストと、割振りとを含む。各グループクラスモジュールは、複数のクラスモジュールと、フォルダと、それらの階層とからなる。

【００６３】

各グループクラスモジュールは、1つまたは複数のグループアプリケーションモジュール(グループAPM)にインスタンス化することができる。各グループアプリケーションモジュール(グループAPM)はまた、設計情報と、アタッチメントと、割振りとを含む。同様に、各クラスモジュールは、クラスベースアプリケーションモジュール(クラスベースAPM)としてインスタンス化され、クラスベースアプリケーションモジュールは、テンプレートからの制御論理、チューニングパラメータ、アラーム属性と、設計情報およびアタッチメントを含む。

【００６４】

設計情報は、モジュールの設計がワード文書に公開される場合がある文書生成機能において使用される。

【００６５】

アタッチメントは、クラスモジュールに関する関連アーチファクトを記憶するために使用される。アーチファクトは任意のバイナリファイルとすることができる。

【００６６】

制御論理は、プロセス制御の論理情報を含む。制御論理のコンテンツは、制御図面および機能ブロック詳細定義を設計するためのコンテンツである。

【００６７】

アラーム属性のコンテンツは、機能ブロックのアラーム属性および動作ステーション(HIS)用のアラームシステムにおいて取り扱われるアラーム属性を設計するためのコンテンツである。

【００６８】

チューニングパラメータのコンテンツは、制御関連チューニングパラメータの設計値を設計するためのコンテンツである。

【００６９】

各グループクラスモジュールは、1)設計情報と、2)アタッチメントと、3)割振りとで構成される。設計情報およびアタッチメントは、クラスモジュールの場合と同様の機能であり、ドキュメンテーションの対象とされ、一方、割振りは、複合ループを定義するためにクラスモジュールのグループ化および複数の制御図面へのクラスモジュールの割当てを定義する。

【００７０】

子クラスモジュールは、クラスモジュールへのショートカットを含む。子クラスモジュールは、グループモジュールにおける複合ループの論理を定義して形成する。

【００７１】

グループクラスモジュールは、それ自体のトポロジーを定義して制御および/または機器モジュールまたはユニットについての論理構造を表す。

【００７２】

[ソースエンジニアリングテンプレートをインスタンス化した後にソースエンジニアリングテンプレートを分岐させるための分岐プロセス]

分岐プロセスは、1)1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートを複製して、ソースエンジニアリングテンプレートにおける1つまたは複数のソース子テンプレートからインスタンス化された1つまたは複数の子インスタンスを含むインスタンスを複製することなしに、それぞれ、1つまたは複数のソース子テン

プレートに対応する1つまたは複数の複製子テンプレートを有する複製エンジニアリングテンプレートを作成することであって、複製エンジニアリングテンプレートがインスタンスを有さず、1つまたは複数の子インスタンスの各々が、ソースエンジニアリングテンプレートへの元のリンクを有する、複製して作成することと、2)ソースエンジニアリングテンプレートからインスタンス化された1つまたは複数の子インスタンスから少なくとも1つの子インスタンスを選択することであって、選択された少なくとも1つの子インスタンスが、ソースエンジニアリングテンプレートへの元のリンクを有する、選択することと、3)元のリンクを選択された少なくとも1つの子インスタンスと複製エンジニアリングテンプレートとの間の新しいリンクに変更することであって、選択された少なくとも1つの子インスタンスが、複製エンジニアリングテンプレートへの新しいリンクを有し、ソースエンジニアリングテンプレートへの元のリンクを有さず、選択されない1つまたは複数の子インスタンスが、ソースエンジニアリングテンプレートへの元のリンクを保持する、変更することとを行うためのプロセスを指す。複製プロセスは、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートをインスタンス化して、各々が1つまたは複数の子インスタンスを有する1つまたは複数のインスタンスを作成した後に実行される。場合によっては、テンプレートは、限定はしないが、クラスモジュールまたはグループクラスモジュールであってもよい。

【００７３】

さらに、複製エンジニアリングテンプレートをインスタンス化して、複製エンジニアリングテンプレートからの1つまたは複数の子インスタンスを有する追加のインスタンスを作成するための追加のインスタンス化プロセスであって、1つまたは複数の子インスタンスが複製エンジニアリングテンプレートへのリンクを有する、追加のインスタンス化プロセスが実行される。

【００７４】

図3は、ソースエンジニアリングテンプレートをインスタンス化した後にソースエンジニアリングテンプレートを分岐させるための分岐プロセスの全体的な概念の例示的な例の概略図である。

【００７５】

第1に、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレート「テンプレートA」を作成するためのエンジニアリングテンプレート作成プロセスであって、1つまたは複数のソース子テンプレートの各々がソースエンジニアリングテンプレートへのショートカットまたはソースエンジニアリングテンプレートの参照を有する、エンジニアリングテンプレート作成プロセスが実行される。

【００７６】

第2に、1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレート「テンプレートA」をインスタンス化して、ソースエンジニアリングテンプレート「テンプレートA」における1つまたは複数のソース子テンプレートから1つまたは複数の子インスタンス「子インスタンス1」および「子インスタンス2」を各々が有する1つまたは複数のソースインスタンスを作成するインスタンス化プロセスが実行される。

【００７７】

第3に、以下の3つのサブプロセスを含む分岐プロセスが実行される。

【００７８】

1つまたは複数のソース子テンプレートを有するソースエンジニアリングテンプレートを複製して、ソースエンジニアリングテンプレート「テンプレートA」における1つまたは複数のソース子テンプレートからインスタンス化された1つまたは複数の子インスタンス「子インスタンス1」および「子インスタンス2」を複製することなしに、それぞれ、1つまたは複数のソース子テンプレートに対応する1つまたは複数の複製子テンプレートを有する複製エンジニアリングテンプレート「テンプレートA'」を作成するための、第1の分岐サブプロセスとしての複製プロセスであって、複製エンジニアリングテンプレート「テンプレートA'」がインスタンスを有さず、1つまたは複数の子インスタンスの各々は、ソ

ースエンジニアリングテンプレートへの元のリンクを有する複製プロセスが実行される。すなわち、ソースエンジニアリングテンプレートを複製して複製エンジニアリングテンプレート「テンプレートA'」を作成した直後には、複製エンジニアリングテンプレート「テンプレートA'」はインスタンスを有さず、1つまたは複数の子インスタンスの各々は、ソースエンジニアリングテンプレートへの元のリンクを有する。

【００７９】

ソースエンジニアリングテンプレート「テンプレートA」における1つまたは複数のソース子テンプレートからインスタンス化された1つまたは複数の子インスタンス「子インスタンス1」および「子インスタンス2」から少なくとも1つの子インスタンスを選択するための、第2の分岐サブプロセスとしての選択プロセスであって、複製エンジニアリングテンプレート「テンプレートA'」がインスタンスを有さず、選択された少なくとも1つのインスタンスがソースエンジニアリングテンプレートへの元のリンクを有する選択プロセスが実行される。

【００８０】

元のリンクを選択された少なくとも1つの子インスタンスと複製エンジニアリングテンプレートとの間の新しいリンクに変更するための、第3の分岐サブプロセスとしてのリンク変更プロセスであって、選択された少なくとも1つの子インスタンスが、複製エンジニアリングテンプレートへの新しいリンクを有し、ソースエンジニアリングテンプレートへの元のリンクを有さず、選択されない1つまたは複数の子インスタンスがソースエンジニアリングテンプレートへの元のリンクを保持するリンク変更プロセスが実行される。リンク変更プロセスは、選択された少なくとも1つまたは複数の子インスタンスとソースエンジニアリングテンプレートとのリンクを解除するためのサブプロセスと、その後、選択された少なくとも1つまたは複数の子インスタンスを複製エンジニアリングテンプレートとリンクするためのサブプロセスの2つのサブプロセスを含んでもよい。

【００８１】

第4に、複製エンジニアリングテンプレート「テンプレートA'」をインスタンス化して、複製エンジニアリングテンプレート「テンプレートA'」から1つまたは複数の子インスタンスを有する追加のインスタンス「子インスタンス3」を作成するための追加のインスタンス化プロセスであって、1つまたは複数の子インスタンスが、複製エンジニアリングテンプレート「テンプレートA'」へのリンクを有する追加のインスタンス化プロセスが実行される。

【００８２】

上述のように、分析プロセスは、ソースエンジニアリングテンプレート(ソースエンジニアリング「テンプレートA」)を複製することだけでなく、エンジニアがインスタンスまたは子アプリケーションモジュール(子AMP)などのアプリケーションモジュールを選択するのを可能にすることと、複製エンジニアリングテンプレート(「テンプレートA'」)と選択されたインスタンスまたは子アプリケーションモジュール(子AMP)との間のリンケージ/関係を自動的に生成することと、子インスタンス(「子インスタンス1」および「子インスタンス2」)を破壊または分離するのを可能にすることも含む。具体的には、子インスタンス1(「子インスタンス1」)はソースエンジニアリングテンプレート(「テンプレートA」)とともにとどまり、一方、子インスタンス2(「子インスタンス2」)には、複製エンジニアリングテンプレート(「テンプレートA'」)にリンクすることが選択される。

【００８３】

その結果、それぞれソースエンジニアリングテンプレート「A」および新しい複製エンジニアリングテンプレート「A'」にリンクされた子インスタンス(「子インスタンス1」および「子インスタンス2」)に対する修正は、互いに影響を与えることなしに分離される。これに対して、現在の分岐プロセスを実行しない場合、ソースエンジニアリングテンプレート「A」にリンクされた子インスタンス(「子インスタンス1」および「子インスタンス2」)に対する修正は互いに関係付けられ、互いに影響し合う。したがって、分岐プロセスが利用可能でない場合、第1のステップとして、ソースエンジニアリングテンプレート(テ

ンプレートA)と修正目標子インスタンス(「子インスタンス2」)との間のリンケージを、修正目標子インスタンス(「子インスタンス2」)に修正を施す前に破壊しておく必要がある。これに対して、分岐プロセスが利用可能である場合、ソースエンジニアリングテンプレート「A」が複製されて新しい複製エンジニアリングテンプレート「A'」が作成され、子インスタンス(「子インスタンス2」)とソースエンジニアリングテンプレート「A」とのリンクが、リンケージを破壊する代わりに、子インスタンス(「子インスタンス2」)と新しい複製エンジニアリングテンプレート「A'」との新しいリンクに変更され、それによって、修正目標子インスタンス(「子インスタンス2」)は依然としてクラスベースであり、依然として複製エンジニアリングテンプレートA'とのリンケージを有し、一方、他の修正目標子インスタンスすなわち非修正目標子インスタンスはクラスベースであり、依然としてソースエンジニアリングテンプレート「A」とのリンケージを有する。複製エンジニアリングテンプレートA'にリンクされた修正目標子インスタンス(「子インスタンス2」)はインスタンス化されなくてもよい。したがって、エンジニアリング作業を著しく低減させることができる。しかし、新しい子インスタンスまたは追加の子インスタンスが必要である場合、複製エンジニアリングテンプレートA'を任意にインスタンス化して、ソースエンジニアリングテンプレート「A」に影響を与えずに複製エンジニアリングテンプレートA'にリンクされた新しい追加の子インスタンスを作成することができる。

【００８４】

[グループクラスモジュールの分岐]

拡張するグループクラスモジュール(グループCM)のうちで、1つのグループクラスモジュールがソースグループクラスモジュール(ソースグループCM)として選択され、このソースグループクラスモジュールが複製されて、1つまたは複数の新しい複製グループクラスモジュールが作成される。選択されたグループクラスモジュールは、以下ではソースグループクラスモジュール(ソースグループCM)と呼ばれる。ソースグループクラスモジュール(ソースグループCM)から1つまたは複数の新しい複製グループクラスモジュールが作成された後、エンジニアまたはユーザは、リンク生成ツールを使用することができ、ソースグループクラスモジュール(ソースグループCM)からインスタンス化されソースグループクラスモジュール(ソースグループCM)にリンクされたグループアプリケーションモジュール(グループAPM)から、新しい複製グループクラスモジュール(複製グループCM)とリンクすべき1つまたは複数のグループアプリケーションモジュール(グループAPM)を選択することができる。エンジニアまたはユーザはリンク生成ツールを使用することができ、エンジニアまたはユーザが1つまたは複数のグループアプリケーションモジュール(グループAPM)を選択するのを可能にするように構成される。リンク生成ツールは、選択された1つまたは複数のグループアプリケーションモジュール(グループAPM)をソースグループクラスモジュール(ソースグループCM)からリンク解除し、選択された1つまたは複数のグループアプリケーションモジュール(グループAPM)を、ソースグループクラスモジュール(ソースグループCM)から複製された新しい複製グループクラスモジュール(複製グループCM)にリンクするように構成される。リンク生成ツールは、エンジニアまたはユーザによる選択時に、選択された1つまたは複数のグループアプリケーションモジュール(グループAPM)のリンク解除およびリンクを自動的に実行するように構成される。リンク生成ツールは、選択された1つまたは複数のグループアプリケーションモジュール(グループAPM)と新しい複製グループクラスモジュール(複製グループCM)との間に新しいリンクを自動的に生成するように構成される。

【００８５】

複製および選択後に、元はソースグループクラスモジュール(ソースグループCM)にリンクされていたインスタンス化されたグループアプリケーションモジュール(グループAPM)は、ソースグループクラスモジュールではなく、複製グループクラスモジュール(複製グループCM)を参照するかまたは複製グループクラスモジュールにリンクすることができるようになる。具体的には、複製グループクラスモジュール(複製グループCM)と対応する選択されたグループアプリケーションモジュール(グループAPM)との間のリンケージ/リンク

が自動的に生成され、オートメーションエンジニアリングシステムのサーバであるデータベースに記憶される。

【００８６】

エンジニアは、新しいエンジニアリングテンプレート(新しい親)と呼ばれる複製グループクラスモジュール(複製グループCM)上で直接エンジニアリングデータを変更し構成してもよい。新しいエンジニアリングテンプレートに基づいて作成されるインスタンスに変更を施すにはモジュール更新プロセスを実行する必要がある。モジュール更新の詳細については後で説明する。上記のようにして、新しい複製テンプレート上に変更を保存することができ、グループクラスモジュール(グループCM)などのエンジニアリングテンプレートの再使用可能性が向上する。同様のプロジェクトがある場合、エンジニアは、様々な設計要件に基づいて、ソースグループクラスモジュール(ソースグループCM)などの既存のテンプレートまたは複製グループクラスモジュール(複製グループCM)などの新しい複製テンプレートを再使用してもよい。

【００８７】

上述の分岐プロセスは、以下の利点の組を実現する。

【００８８】

分岐プロセスは、同じプロジェクト上の複数のチームによる分離同時作業を可能にする。

【００８９】

分岐プロセスは、リンク解除後に各インスタンスを個々に更新する際にヒューマンエラーが生じる危険性を低減させる。これらのエラーは、子インスタンスなどのエンジニアリングデータに様々なセットの変更が直接施されることによって生じる。

【００９０】

分岐プロセスは、顧客のいくつかの要件に基づいて様々なエンジニアリングテンプレート上での以後の分離された変更を効果的にサポートし、既存の論理のうちのいくつかを分岐させることができ、したがって、工場受入試験(FAT)などのいくつかの必須の試験時またはエンジニアリングフェーズの間にいくつかの変更が必要になる場合にエンジニアリング作業負荷を低減させる。

【００９１】

分岐プロセスは、データリリースを可能にし、顧客から直ちにフィードバックを得る。

【００９２】

様々なエンジニアリングテンプレート上でエンジニアリング作業を分析させ実行することによって、既存の論理のいくつかは影響を受けない。

【００９３】

上記の説明はグループクラスモジュールの分岐の場合についての説明であるが、分岐プロセスはクラスモジュールに適用することができる

【００９４】

[クラスモジュールの分岐]

分岐プロセスはクラスモジュールに適用することができる。ユーザによってクラスモジュールを複製することができ、このことは、1つのソースクラスモジュールに基づいて複数のクラスモジュールを作成できることを意味する。ユーザは、任意の利用可能なユーザインターフェースを使用して、たとえば、どのクラスベースアプリケーションモジュール(クラスベースAPM)、どの子クラスモジュール(子CM)および/または子アプリケーションモジュール(子APM)をソースクラスモジュールの代わりに複製クラスモジュールとリンクすべきかを選択することができる。

【００９５】

複製および選択の後で、元はソースクラスモジュール(ソースCM)にリンクされていた選択されたクラスベースアプリケーションモジュール(クラスベースAPM)、子クラスモジュール、および/または子アプリケーションモジュール(子APM)は、ソースクラスモジュールではなく、新しい複製クラスモジュールを参照することができ、または新しい複製クラス

モジュールとリンクすることができる。具体的には、「複製クラスモジュール」と対応する選択されたクラスベースアプリケーションモジュールAPM、子クラスモジュール、および/または子アプリケーションモジュール(子APM)との間のリンケージ/リンクが自動的に生成され、オートメーションエンジニアリングシステムのサーバなどのデータベースに記憶される。リンク生成構成要素はまた、複製クラスモジュールと選択されたクラスベースアプリケーションモジュール(クラスベースAPM)、子クラスモジュール、および/または子アプリケーションモジュール(子APM)との間にリンクを生成するように構成される。

【００９６】

エンジニアは、複製クラスモジュール上で直接エンジニアリングデータを変更し構成することができる。そのようにして、加えられた変更は新しい複製エンジニアリングテンプレート上に保存され、クラスモジュール(CM)などのエンジニアリングテンプレートの再使用可能性が向上する。同様のプロジェクトがある場合、エンジニアは、設計要件に基づいて、ソースクラスモジュール(ソースCM)などの既存のテンプレートまたは新しい複製テンプレート(複製CM)を再使用することができる。

【００９７】

分岐プロセスに関する以下の例示的な例について、図4および図5を参照して説明する。図4は、グループクラスモジュールを分岐させるためのプロセスを示す概略図である。図5は、図4のグループクラスモジュールを分岐させるためのプロセスの後でレガシークラスモジュールを分岐させるためのプロセスを示す概略図である。

【００９８】

図4に示すように、グループクラスモジュールを分岐させるための分岐グループクラスモジュール対話が起動され、ディスプレイスクリーンにウィンドウ「グループクラスモジュール(GCM1)の分岐」が示される。ユーザは、分岐グループクラスモジュール対話を起動する前に、クラスモジュールライブラリナビゲータにおいてクラスモジュール(CM1、CM2)を作成しておく。さらに、ユーザは、グループクラスモジュールライブラリナビゲータにおいてグループクラスモジュール(GCM1)を作成しており、グループクラスモジュール(GCM1)は、それぞれクラスモジュール(CM1、CM2)へのショートカットを有する子クラスモジュール(CCM1、CCM2)を有する。さらに、ユーザは、グループクラスモジュール(GCM1)をインスタンス化して2つのグループアプリケーションモジュール(GAPM1、GAMP2)をアプリケーション構造ナビゲータにおけるインスタンスとして作成している。グループアプリケーションモジュール(GAPM1、GAPM2)の各々は、それぞれクラスモジュール(CM1、CM2)にリンクされた2つの子アプリケーションモジュール(APM1、APM2)を有する。すなわち、子アプリケーションモジュール(APM1、APM2)は、それぞれクラスモジュール(CM1、CM2)にリンクされる。

【００９９】

分岐グループクラスモジュール対話の第1のステップでは、ユーザは、新しいグループクラスモジュールの分岐グループクラスモジュール名を入力する。新しいグループクラスモジュールは、ディスプレイスクリーン上のウィンドウ内のテキストボックス「新しいグループクラスモジュール名」(301)においてソースグループクラスモジュール(GCM1)を複製することによって作成される。この例では、分岐グループクラスモジュール名は、グループクラスモジュール2を意味する「GCM2」である。このテキストボックスは、分岐グループクラスモジュールの名前、または分岐に使用される作成された複製グループクラスモジュールの名前を入力するために使用され、この場合、デフォルト値は空である。

【０１００】

この状態では、グループクラスモジュール(GCM1)を分岐させるためのプロセスが実行され、グループクラスモジュール(GCM1)の複製である新しい複製グループクラスモジュール(GCM2)が作成される。新しい複製グループクラスモジュール(GCM2)は、クラスモジュール(CM1、CM2)へのショートカットを有する子クラスモジュール(CCM1、CCM2)を有する。新しい複製グループクラスモジュール(GCM2)の子クラスモジュール(CCM1、CCM2)は、グループクラスモジュール(GCM1)の子クラスモジュール(CCM1、CCM2)と同じである。

【０１０１】
分岐グループクラスモジュール対話の第2のステップでは、すべてのリンクされたグループアプリケーションモジュール(グループAPM)、たとえば、選択されたグループクラスモジュールの(GAPM1、GAPM2)が、グリッドコントロール(302)における行として示される。グリッドコントロール(302)は、リンクされたグループアプリケーションモジュール(グループAPM)の情報を含む。グリッドコントロール(302)は、ユーザがリンクされたグループアプリケーションモジュール(グループAPM)のうちのどれを分岐プロセスに含めるべきかを選択するのを可能にし、ユーザは、チェックボックスコラムヘッダを使用してグリッドコントロール(302)内の行を選択し選択解除することができる。
【０１０２】
リンクされたグループアプリケーションモジュール(グループAPM)のうちのどれを分岐プロセスに含めるべきかについてのユーザの選択に基づいて、選択されたグループアプリケーションモジュール(GAPM2)が新しい複製グループクラスモジュール(GCM2)に自動的にリンクされ、新しい複製グループクラスモジュール(GCM2)はもはやグループクラスモジュール(GCM1)にリンクされない。グループアプリケーションモジュール(GAPM)の「パス」パス/位置をアプリケーション構造ナビゲータにおいて参照することができる。
【０１０３】
図5に示すように、レガシークラスモジュールを分岐させるための分岐レガシークラスモジュール対話が起動され、ウィンドウ「レガシーグループクラスモジュール(CM3)の分岐」がディスプレイスクリーンに示される。
【０１０４】
分岐レガシークラスモジュール対話の第1のステップでは、ユーザは、新しいクラスモジュールの分岐レガシークラスモジュール名を入力する。新しいクラスモジュールは、ディスプレイスクリーン上のウィンドウ内のテキストボックス「新しいクラスモジュール名」(401)においてレガシークラスモジュール(CM2)を複製することによって作成される。この例では、分岐クラスモジュール名は、クラスモジュール3を意味する"CM3"である。このテキストボックスは、分岐クラスモジュールの名前、または分岐に使用される作成された複製クラスモジュールの名前を入力するために使用され、この場合、デフォルト値は空である。
【０１０５】
この状態では、クラスモジュール(CM2)を分岐させるためのプロセスが実行され、クラスモジュール(CM2)の複製である新しい複製クラスモジュール(CM3)が作成される。
【０１０６】
分岐クラスモジュール対話の第2のステップでは、すべてのリンクされた子クラスモジュール(GCM1のCCM2、GCM2のCCM2)、および子クラスモジュール(CCM2)についてリンクされた子クラスモジュール(GCM1のCCM2、GCM2のCCM2)を参照する子アプリケーションモジュール、すなわち、GAPM1の子AMP2(CM2)およびGAPM2の子APM2(CM2)が、グリッドコントロール(402)における行として示される。グリッドコントロール(402)は、クラスベースアプリケーションモジュール(クラスベースAPM)および/または子クラスモジュール(CCM2)ならびに子アプリケーションモジュール、すなわち、子クラスモジュールにリンクされた子APM(403)の情報を含む。子アプリケーションモジュール、すなわち、子APM(403)は、別個の列における補足情報として示される。
【０１０７】
グリッドコントロール(402)は、ユーザがクラスベースアプリケーションモジュール(クラスベースAPM)および/または子クラスモジュールならびにそれらのモジュールを参照する子アプリケーションモジュール(子APM)のうちのどれを分岐プロセスに含めるべきかを選択するのを可能にし、具体的には、このことは、チェックボックスを選択してグリッドコントロール(402)における行を選択/選択解除することによって完了することができる。
【０１０８】
どの子クラスモジュール(たとえば、GCM2のCCM2)と子クラスモジュールについてのどの

子アプリケーションモジュール(子APM2)とを一緒に分岐プロセスに含めるべきかについてのユーザの選択に基づいて、子クラスモジュール(たとえば、GCM2のCCM2)と子クラスモジュールの子アプリケーションモジュール(子APM2)が自動的にリンクされて新しい複製クラスモジュール(CM3)が得られ、新しい複製クラスモジュール(CM3)はもはやクラスモジュール(CM2)にはリンクされない。子アプリケーションモジュール(子APM)の「パス」パス/位置は、アプリケーション構造ナビゲータにおいて参照することができる。言い換えれば、選択された子クラスモジュールおよび選択された子クラスモジュールを参照するかまたは選択された子クラスモジュールに対応する子アプリケーションモジュールAPMは同時に、分岐できるように構成される。

【０１０９】

上述の分岐プロセッサは、グループクラスモジュールだけでなくクラスモジュールにも適用することができる。

【０１１０】

図6は、クラスモジュールを分岐させるためのプロセスを示す概略図である。図6に示すように、クラスモジュールを分岐させるための分岐クラスモジュール対話が起動され、ディスプレイスクリーンにウィンドウ「クラスモジュールの分岐」が示される。ユーザは、分岐クラスモジュール対話を起動する前に、クラスモジュールライブラリナビゲータにおいてクラスモジュール(CMA)を作成している。さらに、ユーザは、グループクラスモジュールライブラリナビゲータにおいてグループクラスモジュール(GCM1)を作成しており、グループクラスモジュール(GCM1)は、クラスモジュール(CMA)へのショートカットを有する子クラスモジュール(AAA)を有する。さらに、ユーザは、クラスモジュール(CMA)をインスタンス化して、アプリケーション構造ナビゲータにおいて2つのクラスベースアプリケーションモジュール(CBAPM1、CBAPM2)をインスタンスとして作成している。さらに、ユーザは、アプリケーション構造ナビゲータにおいて2つのグループアプリケーションモジュール(GAPM1、GAPM2)をインスタンスとして作成しており、グループアプリケーションモジュール(GAPM1)は、クラスモジュール(CMA)へのショートカットを有する子アプリケーションモジュール(子APMAAA)を有し、ならびにグループアプリケーションモジュール(GAPM2)も、クラスモジュール(CMA)へのショートカットを有する子アプリケーションモジュール(子APMAAA)を有する。さらに、ユーザは、グループクラスモジュール(GCM1)を複製してグループクラスモジュールライブラリナビゲータにおいて新しい複製グループクラスモジュール(GCM2)を作成しており、新しい複製グループクラスモジュール(GCM2)は、クラスモジュール(CMA)へのショートカットを有する2つの子クラスモジュール(BBB、CCC)を有する。次いで、ユーザは、新しい複製グループクラスモジュール(GCM2)をインスタンス化してアプリケーション構造ナビゲータにおいてグループアプリケーションモジュール(GAPM3)をインスタンスとして作成しており、グループアプリケーションモジュール(GAPM3)は、クラスモジュール(CMA)へのショートカットを有する2つの子アプリケーションモジュール(子APMBBB、子APMCCC)を有する。

【０１１１】

分岐クラスモジュール対話の第1のステップにおいて、ユーザは、新しいクラスモジュールの分岐クラスモジュール名を入力する。新しいクラスモジュールは、ディスプレイスクリーン上のウィンドウ内のテキストボックス「新しいクラスモジュール名」においてソースクラスモジュール(CMA)を複製することによって作成される。この例では、新しいクラスモジュール名は"CMB"である。このテキストボックスは、分岐クラスモジュールの名前、または分岐に使用される作成された複製クラスモジュールの名前を入力するために使用され、デフォルト値は空である。

【０１１２】

この状態では、クラスモジュール(CMA)を分岐させるためのプロセスが実行され、新しい複製クラスモジュール(CMB)が作成される。新しい複製クラスモジュール(CMB)は、図6には示されておらず、クラスモジュール(CMA)の複製である。

【０１１３】

分岐グループクラスモジュール対話の第2のステップでは、クラスベースアプリケーションモジュール(CBAPM)、たとえば、クラスベースアプリケーションモジュール(CBAPM1)、クラスベースアプリケーションモジュール(CBAPM2)、子クラスモジュール、すなわち、GCM1の子クラスモジュール(AAA)、GCM2の子クラスモジュール(BBB)および子クラスモジュール(CCC)、ならびに子クラスモジュールを参照する子アプリケーションモジュールが、グリッドコントロールにおける行として示される。グリッドコントロールは、リンクされたクラスベースアプリケーションモジュール(CBAPM)、グループクラスモジュールの子クラスモジュール(グループCM)、および子クラスモジュールを参照する子APMの情報を含む。グリッドコントロールは、ユーザが、リンクされたクラスベースアプリケーションモジュール(CBAPM)、グループクラスモジュールの子クラスモジュール(グループCM)、および/または子クラスモジュールを参照する子APMのうちのどれを分岐プロセスに含めるべきかを選択するのを可能にし、ユーザは、チェックボックスコラムヘッダを使用してグリッドコントロールにおける行を選択し選択解除することができる。

【０１１４】

リンクされたクラスベースアプリケーションモジュール、グループクラスモジュール(グループCM)の子クラスモジュール、および子クラスモジュールを参照する子APMのうちのどれを分岐プロセスに含めるべきかについてのユーザの選択に基づいて、選択されたクラスベースアプリケーションモジュール(すなわち、(1)CBAPM1)、子クラスモジュール(すなわち、(3)AAA、(8)CCC)、およびAAAの子APM、すなわち、(4)GAPM1の子APMAAAおよび(5)GAPM2の子APMAAA、ならびにCCCの子APM、すなわち、(9)GAPM3の子APMCCCが自動的にリンクされて新しい複製クラスモジュール(CMB)が得られ、新しい複製クラスモジュール(CMB)は、もはやクラスモジュール(CMA)にはリンクされない。クラスベースアプリケーションモジュール(CBAPM1)およびグループアプリケーションモジュール(GAPM)の「パス」パス/位置は、アプリケーション構造ナビゲータにおいて参照することができる。

【０１１５】

クラスモジュールおよび/またはグループクラスモジュールを分岐させるための上述の分岐プロセスは、プラントエンジニアリングシステムにおけるグループモジュールベースエンジニアリングシステムにおいて実施することができる。上記の説明は、グループクラスモジュールおよびクラスモジュールなどのエンジニアリングテンプレートを分岐させるための分岐プロセスを対象としている。以下の説明は、上述の分岐プロセスと組み合わせて同じくエンジニアリングプロセスのために実行される他のプロセスを対象とする。上述の分岐プロセス以外のプロセスは、インスタンス化プロセス、更新プロセス、およびバインディングプロセスを含んでもよい。上記で簡単に説明したように、上述の分岐プロセスは、更新プロセスについての実際の作業を低減させ、元のテンプレートの再使用可能性が向上する。

【０１１６】

[クラスモジュールおよびグループクラスモジュールを使用したモジュールベースエンジニアリング]

工業プラントのオートメーション設計についてのクラスモジュールベースエンジニアリングにおけるモジュール分岐プロセスの最も重要な機能のうちの1つは、再使用可能性の向上である。試験済みのクラスモジュールおよび/またはグループクラスモジュールを再使用すると、エンジニアリング品質を向上させ、エンジニアリング時間および試験時間を短縮することができ、試験済みのクラスモジュールおよび/またはグループクラスモジュールは、一部が修正され、一部が再使用される。

【０１１７】

図面についてのクラスモジュールをインスタンス化指向のクラスモジュールベースオブジェクトの特性および振る舞いに関してグループ化することによって、モジュールベースエンジニアリングの適用範囲が制御モジュール図面、機器モジュール図面、および/またはユニット図面の複合ループに拡張される。モジュールベースエンジニアリングの再使用可能性に関する値は大幅に増大する。複合ループを有する制御モジュール図面については

エクスポートおよびインポートを行うことができる。複合ループを有する制御モジュール図面は、インスタンス化してフィールド制御システムなどの物理デバイスに割り当てることができる。図7は、グループクラスモジュールベースエンジニアリングに基づいて階層アーキテクチャに編成されたプラント制御システムにおける様々な機器を参照する工業プラントの階層を示すブロック図である。グループモジュールベースエンジニアリングの場合、グループクラスモジュールは、制御モジュールレベルまたは最低レベルで再使用可能であるだけでなく、機器モジュールレベルおよびユニットモジュールレベルなどのより高いレベルでも再使用可能である。グループクラスモジュールを使用することによるグループクラスモジュールエンジニアリングの適用範囲は、最低レベルすなわち制御モジュールレベルだけでなく、機器モジュールレベルおよびユニットレベルも包含する。グループクラスモジュールは、複数の図面の範囲を有する。工業用途では、典型的なループは、制御モジュール、機器モジュール、および/またはユニットの複数の図面を有することがある。グループクラスモジュールを使用することによるモジュールエンジニアリングの適用範囲は、制御モジュール、機器モジュール、および/またはユニットの複数の図面の複合ループを包含する。

【０１１８】

グループクラスモジュールは、柔軟に全体的または部分的にインスタンス化されてもよく、それによって、ユーザは図面の高度に複合構造を作成することができるだけでなく、制御モジュール図面のより単純な構造を柔軟にインスタンス化することができる。グループクラスモジュールのインスタンス化は、トップダウンもしくはボトムアップ手法、または制御モジュール図面、機器モジュール図面、および/もしくはユニット図面の任意の複合構造からの単なる抽出であってもよい。トップダウン手法は、たとえば、限定はしないが、ユニット、機器モジュール、および制御モジュールの順番のモジュール開発であってもよい。ボトムアップ手法は、たとえば、限定はしないが、制御モジュール、機器モジュール、およびユニットの順番のモジュール開発であってもよい。

【０１１９】

グループクラスモジュールは、インスタンス化することができるクラスベースでオブジェクト指向のような特性および振る舞いの一部を固守するように設計されてもよい。各グループクラスモジュールは、クラスが別のクラスをその特性および/またはフィールドとして含むかまたは参照するのと同様に、クラスモジュールのショートカットまたは参照である1つまたは複数の子クラスモジュールを含んでもよい。グループクラスモジュールは、クラス継承などのオブジェクト指向のクラスのすべての特性に適用できるとは限らない。このことは、クラスモジュールを別のクラスモジュールまたはグループクラスモジュール、ポリモーフィズムなどから継承するかまたはバイアスさせることは不可能であることを意味する。

【０１２０】

図8は、制御モジュールレベル、機器モジュールレベル、および/またはユニットレベルで再使用可能なグループクラスモジュールを使用するグループモジュールベースエンジニアリングシステムを含むプラントエンジニアリングシステムを示すブロック図である。プラントエンジニアリングシステム50000は、グループモジュールエンジニアリングシステム51000と、制御ネットワーク52000と、複数の物理フィールドコントローラ53000と、フィールドネットワーク54000と、複数のフィールドデバイス55000と、制御システム56000とを含む。制御システム56000は、システムエンジニアリングデータベース56100を有する。制御システム56000は、イーサネット(登録商標)などの任意の利用可能なネットワークを介してグループモジュールエンジニアリングシステム51000に接続される。制御システム56000は、制御ネットワーク52000を介して複数の物理フィールドコントローラ53000に接続される。複数のフィールドデバイス55000はフィールドネットワーク54000に接続され、フィールドネットワーク54000は複数の物理フィールドコントローラ53000にさらに接続され、それによって、複数の物理フィールドコントローラ53000は複数のフィールドデバイス55000を制御する。

【０１２１】
グループモジュールエンジニアリングシステム50000は、グループモジュールエンジニアリングエディタ51000を含む。グループモジュールエンジニアリングエディタ51000は、ライブラリナビゲータ51100と、アプリケーション構造ナビゲータ51200と、システム構造51300と、グループモジュールインスタンス化エンジン51400と、グループモジュール更新エンジン51500と、グループモジュールバインディングエンジン51600とを含む。ライブラリナビゲータ51100は2つの異なる種類のナビゲーショナルライブラリ、たとえば、クラスモジュールライブラリ51110およびグループクラスモジュールライブラリ51120を有する。クラスモジュールライブラリ51110は、再使用可能クラスモジュールをクラスベースエンジニアリングテンプレートとして含む。各クラスモジュールは、設計情報と、アタッチメントと、制御論理と、アラーム属性と、チューニングパラメータとを含む。グループクラスモジュールライブラリ51120は、再使用可能グループクラスモジュールをグループクラスベースエンジニアリングテンプレートとして含む。

【０１２２】
アプリケーション構造ナビゲータの階層を制御システムのプラント階層に反映してモジュールエンジニアリングの適用範囲を拡張可能にすべきである。

【０１２３】
クラスモジュールライブラリ51110は、再使用可能モジュールエンジニアリングデータ(制御論理、アラームパラメータ、およびチューニングパラメータ)ならびにドキュメンテーション(設計情報およびアタッチメント)を含むクラスモジュールの定義を含む。各クラスモジュールは、1)説明情報、2)アタッチメント、3)制御論理、4)アラーム属性、および5)チューニングパラメータから構成される。

【０１２４】
グループクラスモジュールライブラリ51120は、複数の制御図面にバインドされたより複合ループを形成するためのクラスモジュールのグループ化を含むグループクラスモジュールの定義を含む。

【０１２５】
設計情報は、モジュールの設計がワード文書に公開される場合がある文書生成機能において使用される。

【０１２６】
アタッチメントは、クラスモジュールに関する関連するアーチファクトを記憶するために使用される。アーチファクトは、任意のバイナリファイルとすることができる。

【０１２７】
制御論理はプロセス制御の論理情報を含む。制御論理のコンテンツは、制御図面および機能ブロック詳細定義を設計するためのコンテンツである。

【０１２８】
アラーム属性のコンテンツは、機能ブロックのアラーム属性および動作ステーション(HIS)用のアラームシステムにおいて取り扱われるアラーム属性を設計するためのコンテンツである。

【０１２９】
チューニングパラメータのコンテンツは、制御関連チューニングパラメータの設計値を設計するためのコンテンツである。

【０１３０】
各グループクラスモジュールは、1)設計情報と、2)アタッチメントと、3)割振りとで構成される。設計情報およびアタッチメントは、クラスモジュールの場合と同様の機能であり、ドキュメンテーションの対象とされ、一方、割振りは、複合ループを定義するためにクラスモジュールのグループ化および複数の制御図面へのクラスモジュールの割当てを定義する。

【０１３１】
子クラスモジュールは、クラスモジュールへのショートカットを含む。子クラスモジュ

ールは、グループモジュールにおける複合ループの論理を定義して形成する。
【０１３２】
グループクラスモジュールは、それ自体のトポロジーを定義して制御および/または機器モジュールまたはユニットについての論理構造を表す。
【０１３３】
(グループモジュールライブラリ)
各グループクラスモジュールは、複合構造を有する。各グループクラスモジュールは、設計情報と、アタッチメントと、割振りとを含む。各グループクラスモジュールは1つまたは複数のクラスフォルダを有してもよい。クラスフォルダは、グループクラスモジュールまたはグループクラスモジュールにおける別のクラスフォルダの下位に作成することができ、それによって、クラスフォルダは、1つもしくは複数の子クラスモジュールまたは1つもしくは複数の他のクラスフォルダを含んでもよい。各グループクラスモジュールはまた、1つまたは複数の子クラスモジュールを有してもよい。図8は、グループクラスモジュールの複合構造の一例を示す。ここで、フォルダ1などのクラスフォルダはグループクラスモジュール1に属する。グループクラスモジュール1は、それぞれ複数のクラスモジュール1、2、---nに関連付けられた複数の子クラスモジュール1、2、---nを含む。この例では、子クラスモジュール1はクラスフォルダ1に属する。他の残りの子クラスモジュール2、3、---nは共通的にグループクラスモジュール1に属する。グループクラスモジュールと通常クラスモジュールとの違いの1つは、グループクラスモジュールが割振りと、それぞれクラスモジュールに関連付けられた複数の子クラスモジュールとを含むことである。グループクラスモジュールおよび複数の子クラスモジュールについての割振りによって、適用範囲が拡張され、グループクラスモジュールベースエンジニアリングの適用範囲を最低階層レベルの制御モジュール図面の規則的な単純ループだけでなく、より高い階層レベル、たとえば機器レベルおよびユニットレベルにおける図面の複合ループも包含することができる。
【０１３４】
第1に、ユーザはグループクラスモジュールライブラリ51120におけるグループクラスモジュールを作成する。ユーザは、次いで、グループクラスモジュールの下位にクラスフォルダと子クラスモジュールを作成する。子クラスモジュールは、レガシークラスモジュールへのショートカットファイルである。
【０１３５】
エンジニアリングテンプレートとしてのグループクラスモジュールにおける割振りは、制御モジュール図面についてのモジュールグループ化情報を含む。割振りは、少なくとも2つ、すなわち、1)クラスモジュールのグループ化、および2)制御モジュール、機器モジュール、および/またはユニットの複数の図面へのクラスモジュールの割当てを定義して、複数の論理図面の複合ループ構造を定義する。グループアプリケーションモジュールについての別の種類の割振りがあり、これについては後述する。ここでは、複数の論理図面の複合ループ構造についてのエンジニアリングテンプレートとしてのグループクラスモジュールにおける割振りについて説明する。
【０１３６】
それぞれの複数の子クラスモジュールは、複数のグループクラスモジュールの関連するグループクラスモジュールに属する。各子クラスモジュールは、複数のクラスモジュールの関連するクラスモジュールを参照するそれぞれのショートカットファイルを含む。それぞれの複数の子クラスモジュールは、関連するグループクラスモジュールに属し、関連するグループクラスモジュールにおける複合ループの論理を定義して形成する。
【０１３７】
エンジニアリングテンプレートとしてのグループクラスモジュールに属する子クラスモジュールは、クラスモジュールライブラリ51110における複数のクラスモジュールのクラスモジュールを参照するそれぞれのショートカットファイルを含む。子クラスモジュールが参照するクラスモジュールは、子クラスモジュールについてのレガシークラスモジュー

ルである。エンジニアリングテンプレートとしてのグループクラスモジュールは、ユーザが各子クラスモジュールのインスタンス化または非インスタンス化を選択するのを可能にする。言い換えれば、エンジニアリングテンプレートとしてのグループクラスモジュールは、ユーザが1つまたは複数の子クラスモジュールをインスタンス化しないモジュールとして選択し、残りの子クラスモジュールをインスタンス化するモジュールとして選択するのを可能にする。レガシークラスモジュールに対する修正は、レガシークラスモジュールを参照しレガシークラスモジュールからインスタンス化される1つまたは複数の子クラスモジュールのすべてに直接反映される。これによって、モジュール保全性がある程度向上する。1つまたは複数の子クラスモジュールが参照するレガシークラスモジュールはエクスポート可能でありかつインポート可能である。グループクラスモジュールは、グループクラスモジュールごとにチェックアウト/チェックインされる。各子クラスモジュールを独立にチェックアウト/チェックインすることはできない。各子クラスモジュールはレガシークラスモジュールから削除することができる。レガシークラスモジュールに追加の子クラスモジュールを追加することができ、追加の子クラスモジュールは、レガシークラスモジュールを参照するショートカットファイルを有する。各子クラスモジュールのショートカットファイルは、クラスがその特性および/またはフィールドとして別のクラスを含むかまたは参照するのと同様にレガシークラスモジュールを参照することができる。グループクラスモジュールベースエンジニアリングの場合、レガシークラスモジュールを参照する複数の子クラスモジュールがグループクラスモジュールの複合ループの論理を定義し形成するので、子クラスモジュールが必要である。

【０１３８】

グループクラスモジュールは、それ自体のトポロジーを制御モジュール図面、機器モジュール図面、および/またはユニット図面についての論理構造を表すように定義することができる。

【０１３９】

機能仕様などの設計情報は、モジュール構成要素として定義されてもよい。設計情報は一般に、モジュールの詳細について説明するテキスト、画像、テーブルを含んでもよい。制御論理は、制御図面と、機能ブロック、スイッチ、およびメッセージの詳細な定義とを含んでもよい。制御論理は、クラスモジュールまたはアプリケーションモジュールにおいて定義されてもよい。モジュールベースエンジニアリングは、フィールド制御システムの制御論理において定義された機能ブロックにおけるチューニングパラメータ設計値のバルク編集により、かつチューニングパラメータ設計値およびフィールド制御システムの現在の値を比較し設定することによって、チューニングパラメータをモジュール構成要素と見なすことを可能にする。アラーム属性は、アラーム設定値およびアラーム優先順位であってもよい。任意のファイルをモジュール構成要素として付加することができる。アタッチメントのリストは、単純な動作によって起動することができる。

【０１４０】

(グループクラスモジュールのインスタンス化)

図8に示すように、モジュールエンジニアリングエディタ51000はグループモジュールインスタンス化エンジン51400を含む。グループモジュールインスタンス化エンジン51400は、選択された子クラスモジュールのユーザ構成に基づいてグループアプリケーションモジュール(グループAPM)を作成する。子クラスモジュールのうちのいくつかを、導出されたグループアプリケーションモジュール(グループAPM)から除外することが可能である。これによって、エンジニアまたはユーザは、グループモジュールについてのモジュールエンジニアリングデータをフェーズごとに構成することができる。

【０１４１】

グループモジュールインスタンス化エンジン51400は、グループクラスモジュールおよびグループクラスモジュールライブラリ51120内の子クラスモジュール1、2、---nをインスタンス化して、アプリケーション構造ナビゲータ51200においてインスタンスおよびインスタンスに属する子インスタンスを生成するように構成される。グループモジュールイ

ンスタンス化エンジン51400によってグループクラスモジュール1から作成されるかまたはインスタンス化されるインスタンスは、インスタンスとしてのグループアプリケーションモジュール1および2(グループAPM2、グループAPM2)およびアプリケーションフォルダ1を含む。アプリケーションフォルダ1はグループアプリケーションモジュール1(グループAPM1)に属する。ここで、部分的なインスタンス化の結果として、グループアプリケーションモジュール2(グループAPM2)には、インスタンス化されていないグループクラスモジュールライブラリ51120内の子クラスモジュール1を含むアプリケーションフォルダ1をクラスフォルダ1として含めないことが可能である。

【０１４２】

グループアプリケーションモジュール1および2(グループAPM1、グループAPM2)はあるプロジェクト(PJT)に属する。グループクラスモジュールライブラリ51120内の子クラスモジュール1、2、---nからグループモジュールインスタンス化エンジン51400によって作成されるかまたはインスタンス化される子インスタンスは、子アプリケーションモジュール1、2、---nの第1のセットならびに子アプリケーションモジュール2、---nの第2のセットを含む。子アプリケーションモジュール1、2、---nの第1のセットはグループアプリケーションモジュール1(グループAPM1)に属し、子アプリケーションモジュール1は、グループアプリケーションモジュール1(グループAPM1)に属するアプリケーションフォルダ1に属する。他の子アプリケーションモジュール2、---nは、直接グループアプリケーションモジュール1(グループAPM1)に属する。子アプリケーションモジュール1、2、---nの第1のセットはそれぞれクラスモジュール1、2、---nに関連付けられる。子アプリケーションモジュール2、---nの第2のセットは、直接グループアプリケーションモジュール2(グループAPM2)に属する。子アプリケーションモジュール2、---nの第2のセットはそれぞれクラスモジュール2、---nに関連付けられる。

【０１４３】

グループアプリケーションモジュール1および2(グループAPM1、グループAPM2)の各々は、設計情報と、アタッチメントと、割振りとを含む。グループアプリケーションモジュール1(グループAPM1)の割振りは、グループクラスモジュールの割振りからの単なるサブセットであってもよい。グループアプリケーションモジュール1および2(グループAPM1、グループAPM2)の各々における割振りは、制御モジュール図面、機器モジュール図面、および/またはユニット図面についてのモジュールグループ化情報を含んでもよい。グループアプリケーションモジュール1および2(グループAPM1、グループAPM2)の各々における割振りは、少なくとも2つ、すなわち、1)グループアプリケーションモジュール1および2のグループ化、ならびに2)複数の論理図面の複合ループ構造を定義するための制御モジュール、機器モジュール、および/またはユニットの複数の図面へのグループアプリケーションモジュール1および2の割当てを定義する。

【０１４４】

さらに、グループモジュールインスタンス化エンジン51400は、クラスモジュールライブラリ51110内のクラスモジュール1、2、---nをインスタンス化して、それぞれクラスモジュール1、2、---nに関連付けられたクラスベースアプリケーションモジュール1、2、---nを生成するように構成される。クラスベースアプリケーションモジュールを生成した後、クラスベースアプリケーションモジュールをグループモジュール更新エンジン51500によって更新することができ、それによって、クラスベースアプリケーションモジュールとクラスモジュールとの間の関連付けが破壊され、クラスベースアプリケーションモジュールがクラスモジュールから独立し、その結果、クラスベースアプリケーションモジュールはクラスレスアプリケーションモジュールになる。

【０１４５】

図8に示すように、子アプリケーションモジュール1、2、---n、クラスベースアプリケーションモジュールn、およびクラスレスアプリケーションモジュールの各々は、設計情報と、制御論理と、チューニングパラメータと、アラーム属性と、アタッチメントとを含む。機能仕様などの設計情報は、モジュール構成要素として定義されてもよい。設計情報

は一般に、モジュールの詳細について説明するテキスト、画像、および表を含んでもよい。制御論理は、制御図面、ならびに機能ブロック、スイッチ、およびメッセージの詳細な定義を含んでもよい。制御論理は、クラスモジュールまたはアプリケーションモジュールにおいて定義されてもよい。モジュールベースエンジニアリングは、フィールド制御システムの制御論理において定義された機能ブロックにおけるチューニングパラメータ設計値のバルク編集により、かつチューニングパラメータ設計値およびフィールド制御システムの現在の値を比較し設定することによって、チューニングパラメータをモジュール構成要素と見なすことを可能にする。アラーム属性は、アラーム設定値およびアラーム優先順位であってもよい。任意のファイルをモジュール構成要素として付加することができる。アタッチメントのリストは、単純な動作によって起動することができる。

【０１４６】

モジュールエンジニアリングエディタ51000は、グループモジュール更新エンジン51500をさらに含む。グループモジュール更新エンジン51500は、グループアプリケーションモジュール1および2(グループAPM1、グループAPM2)ならびにクラスベースアプリケーションモジュール(クラスベースAPM)を更新するように構成される。モジュールエンジニアリングエディタ51000は、モジュールバインディングエンジン51600を含む。モジュールバインディングエンジン51600は、グループアプリケーションモジュール1(グループAPM1)に属するアプリケーションモジュール1および2(子APM1、子APM2)を制御図面"DRXXXX"にバインドし、また、別のインスタンス、子アプリケーションモジュール(子APM3)を別の制御図面"DRYYYY"にバインドするように構成される。同様に、モジュールバインディングエンジン51600は、クラスベースアプリケーションモジュール(クラスベースAPM)とクラスレスアプリケーションモジュール(クラスレスAPM)の両方をさらに別の制御図面"DRZZZZ"にバインドするように構成される。システム構造51300における制御図面"DRXXXX"、"DRYYYY"、および"DRZZZZ"は、エンジニアリングデータに変換され、エンジニアリングデータはシステムエンジニアリングデータベース56100に記憶される。図5に示すように、グループクラスモジュールおよびクラスモジュールからインスタンス化されたアプリケーションモジュールは、モジュールバインディングエンジン51600によって複数の制御図面にバインドされ、制御モジュール図面、機器モジュール図面、および/またはユニット図面の複合ループを定義する。エンジニアリング適用範囲を機器モジュールおよび/またはユニットレベルに拡張した結果、工業制御についてのエンジニアリング効率は著しく改善されている。

【０１４７】

上述のように、グループクラスモジュールライブラリ51120内のグループクラスモジュールおよび子クラスモジュールの階層は、グループモジュールインスタンス化エンジン51400によるグループクラスモジュールライブラリ51120におけるグループクラスモジュールおよび子クラスモジュールのインスタンス化時にアプリケーション構造ナビゲータ51200に反映される。ユーザは、モジュールライブラリ51110内のクラスモジュールのグループ化を構成してグループクラスモジュールライブラリ51120においてグループクラスモジュールを生成する。ユーザがクラスモジュールのグループ化を構成する理由は、このグループ化を後のプロセス、すなわち、インスタンス化後のモジュールバインディングにおいて使用するからである。ユーザは、クラスモジュールのグループ化を構成し、グループクラスモジュールライブラリ51120においてグループクラスモジュールを生成した後、インスタンス化を実行し、グループアプリケーションモジュール(グループAPM)およびその階層が、ユーザがアプリケーション構造ナビゲータ51200において指定した位置に作成される。グループクラスモジュールからインスタンス化されたインスタンスはグループアプリケーションモジュール(グループAPM)と呼ばれる。子クラスモジュールからインスタンス化されたインスタンスは子アプリケーションモジュール(子APM)と呼ばれる。ユーザは、インスタンス化を実行する際、インスタンス化すべきでない1つまたは複数の子クラスモジュールを選択するか、または、言い換えれば、インスタンス化すべき子クラスモジュールを選択する。ユーザは、同じグループクラスモジュールをそれぞれに異なる選択肢によって1つまたは複数の子アプリケーションモジュール(子APM)にインスタンス化することがで

きる。この部分的なインスタンス化によって、たとえば、1つまたは複数の子アプリケーションモジュール(子APM)の第1のセットを有する第1のグループアプリケーションモジュールと、1つまたは複数の子アプリケーションモジュール(子APM)の第2のセットを有する第2のグループアプリケーションモジュールが生成されてもよく、1つまたは複数の子アプリケーションモジュール(子APM)の第1のセットは、1つまたは複数の子アプリケーションモジュール(子APM)の第2のセットとは異なる。グループモジュールインスタンス化エンジン51400は、選択された子クラスモジュールのユーザ構成に基づいてグループアプリケーションモジュール(グループAPM)を作成するように構成される。子クラスモジュールのうちのいくつかを、導出されたグループアプリケーションモジュール(グループAPM)から除外することが可能である。これによって、エンジニアは、グループモジュールについてのモジュールエンジニアリングデータをフェーズごとに構成することができる。モジュールエンジニアリングエディタ51000は、グループクラスモジュールの適用可能なインスタンスを作成してユーザがテンプレートグループクラスモジュールを再使用するのを可能にするための処理系を含む。

【０１４８】

設計情報は、グループアプリケーションモジュール(グループAPM)によってコピーされることもまたは継承されることもない。設計情報は、その現実世界の用途、設計、および目的または具体的な用途、設計、および目的について説明するための設計情報自体のドキュメンテーションを有してもよい。アタッチメントは、グループアプリケーションモジュール(グループAPM)によってコピーされることもまたは継承されることもない。アタッチメントは、その現実世界の用途、設計、および目的または具体的な用途、設計、および目的について説明するためのアタッチメント自体のドキュメンテーションを有してもよい。グループアプリケーションモジュール(グループAPM)の割振りは、どの子クラスモジュールがインスタンス化されるかに応じてグループクラスモジュールの割振りからの単なるサブセットであってもよい。

【０１４９】

ユーザは、グループクラスモジュールのインスタンス化にどの子クラスモジュールを含めるべきかを選択する。

【０１５０】

それぞれの子アプリケーションモジュール(子APM)1、2、および3は通常、1)設計情報と、2)アタッチメントと、3)制御論理と、4)アラーム属性と、5)チューニングパラメータとを有する。

【０１５１】

アプリケーション構造ナビゲータ51200内の子アプリケーションモジュール1(子APM1)における設計情報とクラスモジュールライブラリ51110内のクラスモジュール1における設計情報の間には違いがある。

【０１５２】

アプリケーション構造ナビゲータ51200内の子アプリケーションモジュール1(子APM1)におけるアタッチメントとクラスモジュールライブラリ51110内のクラスモジュール1におけるアタッチメントの間には違いがある。

【０１５３】

いくつかの編集可能なフィールドにおいて、アプリケーション構造ナビゲータ51200内の子アプリケーションモジュール(子APM1)における制御論理とクラスモジュールライブラリ51110内のクラスモジュール1における制御論理の間には違いがある。

【０１５４】

いくつかの編集可能なフィールドにおいて、アプリケーション構造ナビゲータ51200内の子アプリケーションモジュール1(子APM1)におけるアラーム属性とクラスモジュールライブラリ51110内のクラスモジュール1におけるアラーム属性の間には違いがある。

【０１５５】

いくつかの編集可能なフィールドにおいて、アプリケーション構造ナビゲータ51200内

の子アプリケーションモジュール1(子APM1)におけるチューニングパラメータとクラスモジュールライブラリ51110内のクラスモジュール1におけるチューニングパラメータの間には違いがある。

【０１５６】

単にソースグループクラスモジュールからのインスタンス化によってモジュールエンジニアリングデータをトポロジーおよびコンテンツに関して再使用し、さらにエンジニアリングの労力を最小限に抑えるために、グループアプリケーションモジュール1(グループAPM1)およびグループアプリケーションモジュール2(グループAPM2)はグループクラスモジュール1から作成される。依然としていくつかの編集可能なパラメータを構成する必要がある。クラスベースアプリケーションモジュール(クラスベースAPM)はソースクラスモジュールを有し、クラスレスアプリケーションモジュール(クラスレスAPM)は、ソースクラスモジュールを参照しない。

【０１５７】

図9は、グループクラスモジュール、およびグループクラスモジュールから部分的にインスタンス化された2つのグループアプリケーションモジュールの一例を示す。割振りは、クラスモジュールのグループ化、および複合ループを定義するための複数の制御図面へのクラスモジュールの割当てを定義する。一例として、ライブラリ内のグループクラスモジュール「ABCD_UNIT」は、グループAPM「UNIT_0001(ABCD_UNIT)」またはグループAPM「UNIT_0002(ABCD_UNIT)」としてインスタンス化されている。ユーザは、部分的なインスタンス化を実行するための設計ニーズおよび/または設計フェーズに応じてインスタンス化すべきでないグループクラスモジュールの下位の子クラスモジュールを選択することができる。たとえば、図6に示すように、グループクラスモジュール「ABCD_UNIT」の子クラスモジュール「BBB(CM_2)」はグループAPM「UNIT_0001(ABCD_UNIT)」ではインスタンス化されていない。

【０１５８】

最後に、論理制御図面は、グループクラスモジュールにおいて構成されたが、物理フィールド制御システム33000の制御図面に割り当てられる。場合によっては、同じグループクラスモジュールの下位のグループアプリケーションモジュール(グループAPM)は、同じ物理フィールド制御システム33000に割り当てられるべきである。複数の子クラスモジュールからレガシークラスモジュールが参照されるとき、レガシークラスモジュールについての修正は複数の子クラスモジュールに直接反映され、それによってモジュール保全性が向上する。

【０１５９】

(制御図面用のグループアプリケーションモジュールのバインド)

図8に示すように、モジュールエンジニアリングエディタ51000はモジュールバインディングエンジン51600を含む。モジュールバインディングエンジン51600は、グループアプリケーションモジュール1(グループAPM1)に属する子アプリケーションモジュール1および2(子APM1、子APM2)を制御図面「DRXXXX」にバインドし、また、別のインスタンス、子アプリケーションモジュール(子APM3)を別の制御図面「DRYYYY」にバインドするように構成される。同様に、モジュールバインディングエンジン51600は、子APM、すなわち、グループAPM2の子APM2、クラスベースアプリケーションモジュール(クラスベースAPM)とクラスレスアプリケーションモジュール(クラスレスAPM)の両方をさらに別の制御図面「DRZZZZ」にバインドするように構成される。モジュールバインディングエンジン51600の機能は、子アプリケーションモジュール(子APM)間の関係を維持することである。レガシーアプリケーションモジュール(レガシーAPM)は任意の図面に独立にバインドすることができ、レガシーアプリケーションモジュール(レガシーAPM)にはグループ化も制限も適用されない。グループアプリケーションモジュール(グループAPM)Group APMでは、割振り情報は、子アプリケーションモジュール(子APM)にグループ化をあらかじめ割り当てて、子アプリケーションモジュール(子APM)を同じ制御図面に割り当てさせる。

【０１６０】

(インスタンス化後のグループクラスモジュールの更新)
グループモジュール更新エンジン51500は、グループクラスモジュールからのトポロジーおよび割振りデータの変更を、グループアプリケーションモジュール(グループAPM)についてのグループクラスモジュールの適用可能なインスタンスに同期させるために使用される。グループモジュール更新エンジン51500は、グループアプリケーションモジュール1および2のループ構造におけるトポロジーおよび1つまたは複数の目標グループアプリケーションモジュール1および/または2のトポロジーを更新して、グループアプリケーションモジュール1および2のトポロジーの更新を、1つまたは複数の目標グループアプリケーションモジュール1および/または2のトポロジーに同期させるかまたは反映するように構成される。グループモジュール更新エンジン51500は、各グループクラスモジュールの割振りおよびグループクラスモジュールライブラリ51120内の各グループクラスモジュール1の複合ループのそれぞれのトポロジーを更新するように構成される。グループモジュール更新エンジン51500は、目標グループアプリケーションモジュール1または2(目標グループAPM1またはグループAPM2)の割振りおよび目標グループアプリケーションモジュール1または2(目標グループAPM1またはグループAPM2)の複合ループのそれぞれのトポロジーを更新するように構成される。グループモジュール更新エンジン51500は、どのグループアプリケーションモジュールおよびループ構造におけるグループアプリケーションモジュールのトポロジーを目標グループアプリケーションモジュール(目標グループAPM)に反映しなければならないかを特定するか、またはユーザが特定するのを可能にするように構成される。言い換えれば、グループモジュール更新エンジン51500は、グループクラスモジュールのループ構造におけるトポロジーの更新による変更または修正を目標グループアプリケーションモジュール(目標グループAPM)のループ構造におけるトポロジーに反映するために使用される。

【０１６１】
上述のように、各グループクラスモジュールは複合ループ構造を有する。グループクラスモジュールには複合更新管理が必要である。上述のように、各グループクラスモジュールは設計情報と、アタッチメントリストと、割振りとを有する。同じく上述のように、各グループアプリケーションモジュール(グループAPM)は、設計情報と、アタッチメントリストと、割振りとを有する。設計情報とアタッチメントは別個の定義であり、インスタンス化またはモジュール更新時には更新されない。「更新」という用語は、変更、修正、追加、および削除のためのプロセスを指す。グループクラスモジュールのループ構造におけるトポロジーの更新による変更または修正を目標グループアプリケーションモジュール(目標グループAPM)のループ構造におけるトポロジーに反映するためのプロセスは、1)目標グループアプリケーションモジュール(目標グループAPM)のループ構造におけるトポロジーを更新すること、2)1つまたは複数の子アプリケーションモジュールが属する目標グループアプリケーションモジュール(目標グループAPM)に1つまたは複数の子アプリケーションモジュールを追加するかまたは目標グループアプリケーションモジュール(目標グループAPM)から1つまたは複数の子アプリケーションモジュールを削除すること、および3)1つまたは複数の子アプリケーションモジュールのコンテンツの3つのステップを含んでもよい。

【０１６２】
グループモジュール更新エンジン51500は、グループアプリケーションモジュール(グループAPM)内に子アプリケーションモジュール(子APM)が存在することおよびソース子クラスモジュールの位置に対する変更に基づいて、グループクラスモジュールおよびグループアプリケーションモジュール(グループAPM)のトポロジー間の更新を実行するように構成される。グループモジュール更新エンジン51500は、グループクラスモジュールのすでに作成された適用可能なインスタンスをその新しいバージョンによって更新するための処理系を含む。

【０１６３】
インスタンス化後のグループクラスモジュールの更新についてより詳細に説明し、すな

わち、再使用可能性、チェックアウト/イン、コンテンツ、更新、およびグループ化解除に関して説明する。

【０１６４】

図10は、グループクラスモジュールおよびグループアプリケーションモジュールについての更新管理の一例を示す。インスタンス化後に、グループクラスモジュールの下位においてクラスモジュール「AAA」が削除された場合、モジュール更新がトリガされならびに/または実行されているときには、グループクラスモジュールのインスタンスAPM_1がアプリケーション構造の下位の対応するグループAPMから削除される。グループモジュールの下位においてクラスモジュールEEE(CM_1)が追加された場合、クラスモジュールEEE(CM_1)は別個のインスタンスウィンドウ対話を使用してインスタンス化され、APM_5(CM_1)としてインスタンス化される。エンジニアまたはユーザは、アプリケーション構造の下位のクラスAPMを削除し、たとえば、APM_4を削除することができるが、この削除(DDD(CM_2)によってAPM_4のソースクラスモジュールが削除されることはない。

【０１６５】

再使用可能性:

グループクラスモジュールは2段階でエクスポートおよび/またはインポートを行うことができる。第1に、子クラスモジュールによって参照されるレガシークラスモジュールがエクスポートならびに/またはインポートされる。第2に、グループクラスモジュールがエクスポートならびに/またはインポートされる。再使用可能なモジュールエンジニアリングデータの、他のシステムプロジェクトのエンジニアリングデータベースへのエクスポートおよびインポートを行うことができ、それによって、他のシステムプロジェクトはこのエンジニアリングデータを再使用することができる。

【０１６６】

チェックアウトおよび/またはチェックイン

チェックアウトおよび/またはチェックインのためのプロセスは、グループクラスモジュールに属するすべての子クラスモジュールを含むグループクラスモジュール全体ごとに実行することができる。すなわち、各子クラスモジュールの独立したチェックアウトおよび/またはチェックインを行うことはできない。これに対して、グループアプリケーションモジュール(グループAPM)については、グループアプリケーションモジュール(グループAPM)のみまたは子アプリケーションモジュール(子APM)との組合せごとにチェックアウトおよび/またはチェックインのプロセスを実行することができる。すなわち、各子アプリケーションモジュール(子APM)の独立したチェックアウトおよび/またはチェックインを行うことができる。

【０１６７】

コンテンツ

上述のように、グループクラスモジュールは、そのグループクラスモジュールについての設計情報と、アタッチメントリストと、割振りとを有する。同じく上述のように、グループアプリケーションモジュール(グループAPM)は設計情報と、アタッチメントと、割振りとを有する。設計情報とアタッチメントは別個の定義であり、インスタンス化またはモジュール更新プロセス時には更新されない。この振る舞いはレガシーモジュールの振る舞いと同じである。

【０１６８】

更新

上述のように、インスタンス化後にグループクラスモジュールが修正される場合、以下の3段階で修正をグループアプリケーションモジュール(グループAPM)に反映することができる。

1)グループAPMトポロジーの更新

2)子APMの追加および/または削除

3)子APMコンテンツの更新

【０１６９】

第1に、グループクラスモジュールについてモジュール更新マネージャが起動される。次いで、グループアプリケーションモジュール(グループAPM)のトポロジーおよび割振りが、モジュール更新マネージャによって更新される。グループアプリケーションモジュール(グループAPM)は、グループクラスモジュールが修正される場合、ユーザの選択および/またはユーザによるトリガによって更新される。グループクラスモジュールを更新することによって子クラスモジュールが削除される場合、対応する子アプリケーションモジュール(子APM)が削除される。この段階では、すべての子アプリケーションモジュール(子APM)のチェックアウトおよび/またはチェックインを行わなければならない。

【０１７０】

第2に、1つまたは複数の子クラスモジュールが追加される場合、追加された子クラスモジュールはインスタンスリストウィンドウにおいてインスタンス化される。1つまたは複数の子アプリケーションモジュール(子APM)を削除する必要がある場合、1つまたは複数の子アプリケーションモジュール(子APM)は、アプリケーション構造ナビゲータにおいて削除される。

【０１７１】

最後に、子クラスモジュールによって参照されるレガシークラスモジュールが更新され、それによって、子アプリケーションモジュール(子APM)のコンテンツが更新される。

【０１７２】

グループAPMのグループ解除

ユーザは、グループアプリケーションモジュール(グループAPM)の「グループ解除」を実行することができる。グループ解除は、グループ化のプロセスとは逆のプロセスを指す。グループアプリケーションモジュール(グループAPM)をグループ解除すると、ループ構造が破壊され、各アプリケーションモジュールがアプリケーション構造におけるプロジェクトフォルダとしての最上位のフォルダの直下に配置される。「グループ解除」のためのプロセスの後、グループアプリケーションモジュール(グループAPM)の最上位のフォルダがアプリケーションフォルダになり、最上位のフォルダの直下の設計情報およびアタッチメントリストが削除され、子アプリケーションモジュール(子APM)が、それぞれのクラスモジュールを指すクラスベースアプリケーションモジュール(クラスベースAPM)になる。

【０１７３】

図11は、グループアプリケーションモジュール(グループAPM)についてのモジュール更新のサンプルシナリオを示す。この節ではグループAPMについてのモジュール更新の具体的な例について説明する。図11は、グループAPM EQUIP1を有するグループクラスモジュールGM0001の一例を示す。このグループアプリケーションモジュール(グループAPM)は、グループクラスモジュールに属する3つの子クラスモジュールのうちの2つによって部分的に作成される。

【０１７４】

アプリケーション設定

以下に、グループクラスモジュールおよびグループAPMからの割振り設定のための構成を示す。

【０１７５】

【表１】

グループクラスモジュールにおける割振り設定

モジュール	モジュールパス	グループ番号
CM0001	フォルダ1¥	1
PUMP_LOGIC1	フォルダ1¥	1
PUMP_LOGIC2	フォルダ1¥	2

【表２】

グループAPMにおける割振り設定

モジュール	モジュールパス	グループ番号
PUMP_LOGIC1	フォルダ1¥	1
RECIPE1	フォルダ1¥	1

【０１７６】
モジュールバインディング
これは、グループ番号1がFCS0101→DR0001にバインドされることを表すグループAPMのモジュールバインディング状態を示す。
【０１７７】
【表３】

APM	グループ番号	VPプロジェクト	FCS	番号
EQUIP1	1	VPPJT1	FCS0101	1

【０１７８】
以下の表に、ユーザシナリオおよびモジュール更新を実行した結果が与えられた場合のモジュール更新対話のステータスを表す。
【０１７９】

【表４Ａ】

-1 モジュール更新対話のステータスの判定および更新結果

シナリオ	グループクラスモジュールの変更	ステータス	理由	モジュール更新の結果
子クラスフォルダおよび/または子クラスモジュールの追加	GM0001が新しいクラスフォルダ“Folder2”とともに追加された。	最新	GM0001の変更はEQUIP1における子APMに影響を与えない。グループAPMを更新する必要はない。	
	GM0001が新しいクラスフォルダ“Folder2”およびいくつかの他の子クラスモジュールとともに追加された。	最新	上記と同じ。	
	GM0001が新しいクラスフォルダ“Folder2”とともに追加されており、CM0001が“Folder2”の下位に移動される。	旧式	子クラスモジュール(CM0001)がRECIPE1によって参照されているので構造の変更が検出される。	子アプリケーションフォルダ“Folder2”が作成され、RECIPE1が“Folder2”の下位に移動される。

【表４Ｂ】

子クラスフォルダおよび/または子クラスモジュールの削除	PUMP_LOGIC1が削除される。	旧式	子APM PUMP_LOGIC1が子クラスモジュールを参照している。	グループ番号1に割り当てられたPUMP_LOGIC1はDR0001にバインドされるのでモジュール更新は失敗する。ユーザは最初にバインド解除しなければならない。
	PUMP_LOGIC2が削除される。	最新	EQUIP1は、PUMP_LOGIC2を参照する子APMを有さない。	
名前変更	GM0001→フォルダ1の名前がフォルダAに変更された。	最新	クラスフォルダの名前の変更は構造の変更とは見なされない。	
	MIXTURE1 子アプリケーションフォルダの名前が“MIX_A”に変更される。	最新	子アプリケーションフォルダの名前の変更は構造の変更とは見なされない。	

【表４Ｃ】

	GM0001の名前がA_GM0001に変更される。	最新	これはクラスモジュールおよびクラスベースAPMと同じ振る舞いである。	
モジュールコンテンツ変更	CM0001コンテンツが更新され、チェックインされた。	最新	ソースクラスモジュールのコンテンツ更新はグループモジュール更新の範囲外である。これらの更新は、クラスモジュールからのモジュール更新対話を使用して反映することができる。	
	PUMP_LOGIC2のグループ番号が変更される。	最新	EQUIP1は、PUMP_LOGIC2を参照する子APMを有さない。	
	グループ番号が変更される。 CM0001 = 1 PUMP_LOGIC1 = 2	旧式	上記と同じ。	グループ番号がDR0001にバインドされるのでモジュール更新は失敗する。ユーザは最初にバインド解除しなければならない。

【０１８０】

要するに、上述の実施形態から、工業プラントのオートメーション設計についてのグループクラスモジュールベースエンジニアリングの最も重要な機能の1つが再使用可能性の向上であることを諒解することができる。試験済みのグループクラスモジュールを再使用するとエンジニアリング品質を向上させ、エンジニアリング時間および試験時間を短縮することができ、試験済みのグループクラスモジュールは部分的に修正され部分的に再使用される。

【０１８１】

上述のゲーム装置についての各要素またはデバイスは、ハードウェアとソフトウェアによって実装することも、またはソフトウェアなしでハードウェアによって実装することもできる。場合によっては、ゲーム装置は、1つまたは複数のハードウェアプロセッサおよび1つまたは複数のソフトウェア構成要素によって実装されてもよく、その場合、1つまたは複数のソフトウェア構成要素は、ゲーム装置についての各要素またはデバイスを実装するように1つまたは複数のハードウェアプロセッサによって実行される。いくつかの他の場合には、ゲーム装置は、ゲーム装置についての各要素またはデバイスの各動作を実行するように構成された回路のシステムまたは回路によって実装されてもよい。

【０１８２】

上述の実施形態におけるシステムおよび方法は、コンピュータソフトウェア、ソフトウェア構成要素、プログラムコード、および/または1つもしくは複数のプロセッサに対する命令を実行するマシンまたは回路によって部分的または全体的に展開されてもよい。1つまたは複数のプロセッサは、汎用コンピュータ、サーバ、クラウドサーバ、クライアント、ネットワークインフラストラクチャ、モバイルコンピューティングプラットフォーム、固定コンピューティングプラットフォーム、または他のコンピューティングプラットフォームの一部であってもよい。1つまたは複数のプロセッサは、プログラム命令、コード、バイナリ命令などを実行することができる任意の種類の計算または処理デバイスであってもよい。1つまたは複数のプロセッサは、信号プロセッサ、デジタルプロセッサ、組込みプロセッサ、マイクロプロセッサ、または記憶されたプログラムコードもしくはプログラム命令の実行を直接的もしくは間接的に容易にすることがあるコプロセッサ、たとえば数値演算コプロセッサ、グラフィックコプロセッサ、通信コプロセッサなどの任意の変形例であってもよく、またはそれらを含んでもよい。さらに、1つまたは複数のプロセッサは、複数のプログラム、スレッド、およびコードの実行を可能にしてもよい。各スレッドは、1つまたは複数のプロセッサの実行を向上させ、アプリケーションの同時動作を容易にするように同時に実行されてもよい。本明細書で説明するプログラムコード、プログラム命令などは、1つまたは複数のスレッドにおいて実施されてもよい。1つまたは複数のプロセッサは、本明細書で説明するようにコード、命令、およびプログラムを記憶するメモリを含んでもよい。プロセッサは、本明細書および他の部分で説明するようにコード、命令、およびプログラムを記憶する場合がある非一時的プロセッサ可読記憶媒体にインターフェースによってアクセスしてもよい。プログラム、コード、プログラム命令、またはコンピューティングデバイスもしくは処理デバイスによって実行することができる他の種類の命令を記憶するためにプロセッサに関連付けられた非一時的プロセッサ可読記憶媒体は、限定はしないが、メモリ、ハードディスク、フラッシュドライブ、RAM、ROM、CD-ROM、DVD、キャッシュなどのうちの1つまたは複数を含んでもよい。

【０１８３】

プロセッサは、マルチプロセッサの速度および性能を向上させる場合がある1つまたは複数のコアを含んでもよい。いくつかの実施形態では、プロセッサは、デュアルコアプロセッサ、クアッドコアプロセッサ、2つ以上の独立したコアを組み合わせる他のチップレベルマルチプロセッサなどであってもよい。

【０１８４】

本明細書で説明する方法およびシステムは、サーバ、クライアント、ファイアウォール、ゲートウェイ、ハブ、ルータ、または他のコンピュータおよび/もしくはネットワーキングハードウェア上でコンピュータソフトウェアを実行するマシンによって部分的または

全体的に展開されてもよい。

【０１８５】

ソフトウェアプログラムは、ファイルクライアント、印刷クライアント、ドメインクライアント、インターネットクライアント、イントラネットクライアント、および二次クライアント、ホストクライアント、分散クライアントなどの他の変形例を含んでもよい1つまたは複数のクライアントに関連付けられてもよい。クライアントは、メモリ、プロセッサ、コンピュータ可読媒体、記憶媒体、物理および仮想ポート、通信デバイス、ならびに有線または無線媒体によって他のクライアント、サーバ、マシン、およびデバイスにアクセスすることのできるインターフェースなどのうちの1つまたは複数を含んでもよい。本明細書で説明するようなプログラムまたはコードは、クライアントによって実行されてもよい。さらに、この適用例において説明するような方法を実行するのに必要な他のデバイスは、クライアントに関連するインフラストラクチャの一部と見なされることがある。クライアントは、サーバ、他のクライアント、プリンタ、データベースサーバ、プリントサーバ、ファイルサーバ、通信サーバ、分散サーバなどを含む他のデバイスとのインターフェースを構成してもよい。この結合および/または接続は、ネットワークを介したプログラムのリモート実行を容易にすることがある。これらのデバイスのうちのいくつかまたはすべてのネットワーク化は、1つまたは複数の位置におけるプログラムまたは方法の並列実行を容易にすることがある。さらに、インターフェースによってクライアントに付加されたデバイスのうちの任意のデバイスは、方法、プログラム、アプリケーション、コード、および/または命令を記憶することができる少なくとも1つの記憶媒体を含んでもよい。中央リポジトリは、様々なデバイス上で実行すべきプログラム命令を提供してもよい。この実装形態では、リモートリポジトリは、プログラムコード、命令、およびプログラム用の記憶媒体として働いてもよい。

【０１８６】

ソフトウェアプログラムは、ファイルサーバ、プリントサーバ、ドメインサーバ、インターネットサーバ、イントラネットサーバ、および二次サーバ、ホストサーバ、分散サーバなどの他の変形例を含んでもよい1つまたは複数のサーバに関連付けられてもよい。サーバは、メモリ、プロセッサ、コンピュータ可読媒体、記憶媒体、物理および仮想ポート、通信デバイス、ならびに有線または無線媒体によって他のサーバ、クライアント、マシン、およびデバイスにアクセスすることのできるインターフェースなどのうちの1つまたは複数を含んでもよい。本明細書で説明するような方法、プログラム、またはコードはサーバによって実行されてもよい。さらに、この適用例において説明するような方法を実行するのに必要な他のデバイスは、サーバに関連するインフラストラクチャの一部と見なされることがある。サーバは、クライアント、他のサーバ、プリンタ、データベースサーバ、プリントサーバ、ファイルサーバ、通信サーバ、分散サーバ、ソーシャルネットワークなどを含む他のデバイスとのインターフェースを構成してもよい。この結合および/または接続は、ネットワークを介したプログラムのリモート実行を容易にすることがある。これらのデバイスのうちのいくつかまたはすべてのネットワーク化は、1つまたは複数の位置におけるプログラムまたは方法の並列実行を容易にすることがある。インターフェースによってサーバに付加されたデバイスのうちの任意のデバイスは、プログラム、コード、および/または命令を記憶することができる少なくとも1つの記憶媒体を含んでもよい。中央リポジトリは、様々なデバイス上で実行すべきプログラム命令を提供してもよい。この実装形態では、リモートリポジトリは、プログラムコード、命令、およびプログラム用の記憶媒体として働いてもよい。

【０１８７】

本明細書で説明する方法およびシステムは、ネットワークインフラストラクチャによって部分的または全体的に展開されてもよい。当技術分野で知られているように、ネットワークインフラストラクチャは、コンピューティングデバイス、サーバ、ルータ、ハブ、ファイアウォール、クライアント、パーソナルコンピュータ、通信デバイス、ルーティングデバイス、ならびにその他の能動および受動デバイス、モジュール、ならびに/または構

成要素などの要素を含んでもよい。ネットワークインフラストラクチャに関連するコンピューティングおよび/または非コンピューティングデバイスは、他の構成要素は別として、フラッシュメモリ、バッファ、スタック、RAM、ROMなどの記憶媒体を含んでもよい。本明細書および他の部分で説明するプロセス、方法、プログラムコード、命令は、ネットワークインフラストラクチャ要素のうちの1つまたは複数によって実行されてもよい。

【０１８８】

本明細書および他の部分で説明する方法、プログラムコード、および命令は、モバイルデバイス上でまたはモバイルデバイスを介して実施されてもよい。モバイルデバイスは、ナビゲーションデバイス、携帯電話、モバイルフォン、モバイル携帯情報端末、ラップトップ、パームトップ、ネットブック、ページャ、電子ブックリーダ、音楽プレーヤなどを含んでもよい。これらのデバイスは、他の構成要素は別として、フラッシュメモリ、バッファ、RAM、ROM、および1つまたは複数のコンピューティングデバイスなどの記憶媒体を含んでもよい。モバイルデバイスに関連付けられたコンピューティングデバイスは、記憶されたプログラムコード、方法、および命令を実行するのを可能にされてもよい。代替として、モバイルデバイスは、他のデバイスと協働して命令を実行するように構成されてもよい。モバイルデバイスは、サーバとインターフェース接続されプログラムコードを実行するように構成された基地局と通信してもよい。モバイルデバイスは、ピアツーピアネットワーク、メッシュネットワーク、または他の通信ネットワーク上で通信してもよい。プログラムコードは、サーバに関連付けられた記憶媒体上に記憶され、サーバ内に組み込まれたコンピューティングデバイスによって実行されてもよい。基地局は、コンピューティングデバイスと記憶媒体とを含んでもよい。記憶デバイスは、基地局に関連付けられたコンピューティングデバイスによって実行されるプログラムコードおよび命令を記憶してもよい。

【０１８９】

コンピュータソフトウェア、プログラムコード、および/または命令については、ある時間間隔の間コンピューティングに使用されるデジタルデータを保持するコンピュータ構成要素、デバイス、および記録媒体、ランダムアクセスメモリ(RAM)と呼ばれる半導体ストレージ、光学ディスク、ハードディスク、テープ、ドラム、カード、およびその他の種類のような磁気ストレージ形式などの通常より永久的な記憶用のマスストレージ、プロセッサレジスタ、キャッシュメモリ、揮発性メモリ、不揮発性メモリ、CD、DVDなどの光学ストレージ、フラッシュメモリ、たとえば、USBスティックまたはキー、フロッピーディスク、磁気テープ、紙テープ、パンチカード、スタンドアロンRAMディスク、ジップドライブ、リムーバブルマスストレージ、オフラインなどの取外し可能媒体、ダイナミックメモリ、スタチックメモリ、読取り/書込みストレージ、ミュータブルストレージ、読取り専用、ランダムアクセス、シーケンシャルアクセス、ロケーションアドレス可能、ファイルアドレス可能、コンテンツアドレス可能、ネットワーク接続ストレージ、ストレージエリアネットワーク、バーコード、磁気インクなどの他のコンピュータメモリを含んでもよい機械可読媒体上での記憶および/またはアクセスが行われてもよい。

【０１９０】

本明細書で説明する方法、デバイス、装置、およびシステムは、物理的な品目および/または有形の品目を1つの状態から別の状態に変換してもよい。本明細書で説明する方法およびシステムはまた、物理的な品目および/または有形の品目を表すデータを1つの状態から別の状態に変換してもよい。

【０１９１】

本明細書に記載され、図全体におけるフローチャートおよびブロック図に含まれるモジュール、エンジン、構成要素、および要素は、モジュール、エンジン、構成要素、および要素間の論理的境界を暗示する。しかし、ソフトウェアまたはハードウェアエンジニアリングの慣習に従って、モジュール、エンジン、構成要素、および要素ならびにそれらの機能は、モノリシックソフトウェア構造、スタンドアロンソフトウェアモジュール、または外部ルーチン、コード、サービス、もしくはこれらの任意の組合せを使用するモジュール

として記憶されたプログラム命令を実行することができるコンピュータ実行可能媒体を介して1つまたは複数のプロセッサ、コンピュータ、マシン上に実装されてもよく、すべてのそのような実装形態は本開示の範囲内であってもよい。そのようなマシンの例には、限定はしないが、携帯情報端末、ラップトップ、パーソナルコンピュータ、モバイルフォン、他のハンドヘルドコンピューティングデバイス、医療機器、有線または無線通信デバイス、トランスジューサ、チップ、計算器、衛星、タブレットPC、電子ブック、ガジェット、電子デバイス、人工知能を有するデバイス、コンピューティングデバイス、ネットワーク化機器、サーバ、ルータ、プロセッサ組込みアイウェアなどが含まれてもよい。さらに、フローチャートおよびブロック図におけるモジュール、エンジン、構成要素、および要素または任意の他の論理構成要素は、プログラム命令を実行できる1つまたは複数のマシン、コンピュータ、またはプロセッサ上に実装されてもよい。上記の説明および説明において参照された図面には、開示されたシステムのいくつかの機能態様が記載されているが、明示的に指摘されるかまたは他の点で文脈から明らかでない限り、これらの機能態様を実施するためのソフトウェアの特定の構成をこれらの説明から推定すべきではない。上記で特定し説明した様々なステップが変形されてもよく、ステップの順序が本明細書で開示された技法の特定の適用例に適合されてもよいことが諒解されよう。すべてのそのような変形実施形態および修正実施形態は本開示の範囲内であることが意図される。様々なステップの順序の説明は、特定の適用例によって必要とされるか、または明示的に指摘されるかもしくは他の点で文脈から明らかでない限り、それらのステップを実行する特定の順序を必要とすると理解すべきではない。

【０１９２】

上述の方法および/またはプロセスならびにそれらのステップは、特定の適用例に適したハードウェア、ソフトウェア、またはハードウェアとソフトウェアの任意の組合せにおいて実現されてもよい。ハードウェアは、汎用コンピュータならびに/あるいは専用コンピューティングデバイスまたは特定のコンピューティングデバイスまたは特定のコンピューティングデバイスの特定の態様もしくは構成要素を含んでもよい。プロセスは、1つもしくは複数のマイクロプロセッサ、マイクロコントローラ、組込みマイクロコントローラ、プログラム可能デジタル信号プロセッサ、または他のプログラム可能デバイスにおいて、内部および/または外部メモリと協働して実現されてもよい。プロセスは、追加または代替として、特定用途向け集積回路、プログラム可能ゲートアレイ、プログラム可能アレイロジック、または電子信号を処理するように構成されてもよい任意の他のデバイスもしくはデバイスの組合せにおいて具現化されてもよい。プロセスのうちの1つまたは複数が、機械可読媒体上で実行することができるコンピュータ実行可能コードとして実現されてもよいことがさらに諒解されよう。

【０１９３】

コンピュータ実行可能コードは、上記のデバイスのうちの1つ上に記憶され、かつそのデバイス上で動作するようにコンパイルまたは解釈されてもよいオブジェクト指向プログラミング言語、ならびにプロセッサ、プロセッサアーキテクチャの異種組合せ、または異なるハードウェアおよびソフトウェアの組合せ、またはプログラム命令を実行することができる任意の他のマシンを使用して作成されてもよい。

【０１９４】

したがって、上述の各方法およびそれらの組合せは、1つまたは複数のコンピューティングデバイス上で実行されたときに方法のステップを実行するコンピュータ実行可能コードにおいて具現化されてもよい。別の態様では、方法は、そのステップを実行するシステムにおいて具現化されてもよく、いくつかの方法で各デバイスに分散されてもよく、または機能のすべてが専用スタンドアロンデバイスまたは他のハードウェアに組み込まれてもよい。別の態様では、上述のプロセスに関連するステップを実行するための手段は、上述のハードウェアおよび/またはソフトウェアのうちのいずれを含んでもよい。すべてのそのような変形および組合せは、本開示の範囲内であることが意図される。

【０１９５】

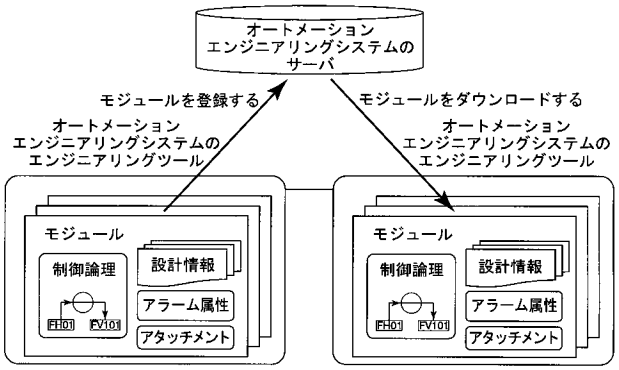
本発明のいくつかの実施形態について説明したが、これらの実施形態は一例としてのみ提示されており、本発明の範囲を制限するものではない。すなわち、本明細書で説明する新規の実施形態は、様々な他の形態で具現化されてもよく、さらに、本明細書で説明した実施形態の形の様々な省略、置換、および変更は、本発明の趣旨から逸脱せずに施されてもよい。添付の特許請求の範囲およびその均等物は、本発明の範囲および趣旨の範囲内であるような形態または修正を対象とするものである。

【符号の説明】

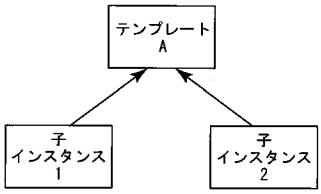
【０１９６】

1、2、3、4　クラスベースアプリケーションモジュール/インスタンス
302、402　グリッドコントロール
401　新しいクラスモジュール名
403　子APM
APM　アプリケーションモジュール
CBAPM　クラスベースアプリケーションモジュール
CCM　子クラスモジュール
CM　クラスモジュール
CMB　新しい複製クラスモジュール
GAPM　グループアプリケーションモジュール
GCM　グループクラスモジュール
33000　物理フィールド制御システム
50000　プラントエンジニアリングシステム
51000　グループモジュールエンジニアリングシステム、モジュールエンジニアリングエディタ
51100　ライブラリナビゲータ
51110　クラスモジュールライブラリ
51120　グループクラスモジュールライブラリ
51200　アプリケーション構造ナビゲータ
51300　システム構造
51400　グループモジュールインスタンス化エンジン
51500　グループモジュール更新エンジン
51600　モジュールバインディングエンジン
52000　制御ネットワーク
53000　物理フィールドコントローラ
54000　フィールドネットワーク
55000　フィールドデバイス
56000　制御システム
56100　システムエンジニアリングデータベース

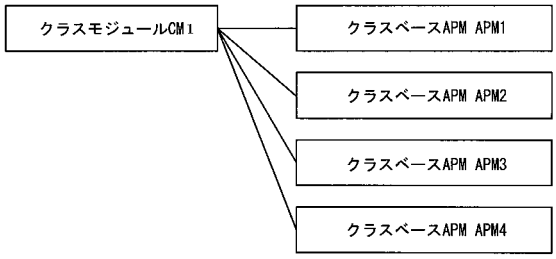
【図１】



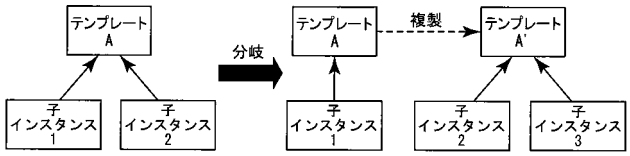
【図２Ａ】



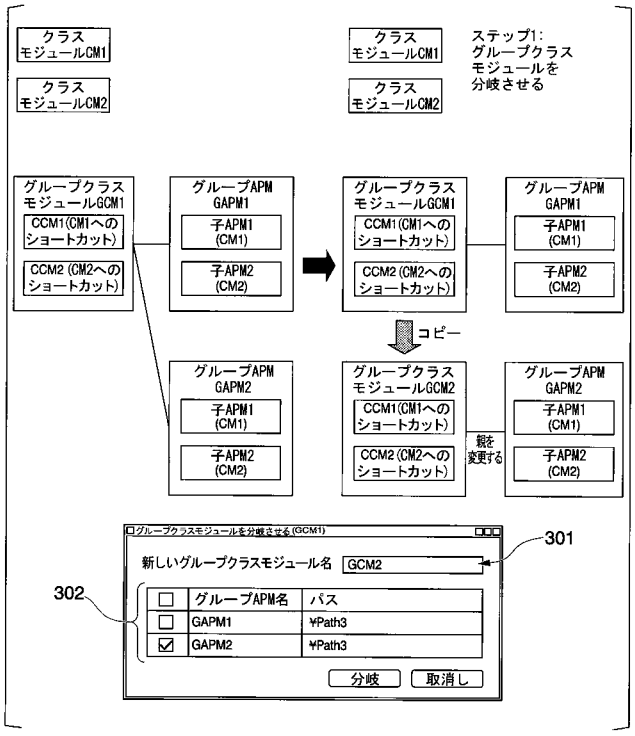
【図２Ｂ】



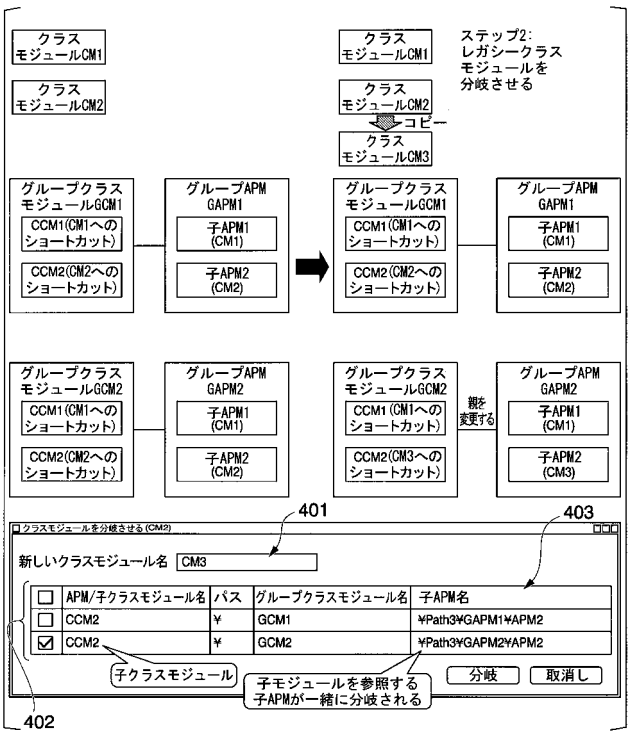
【図３】



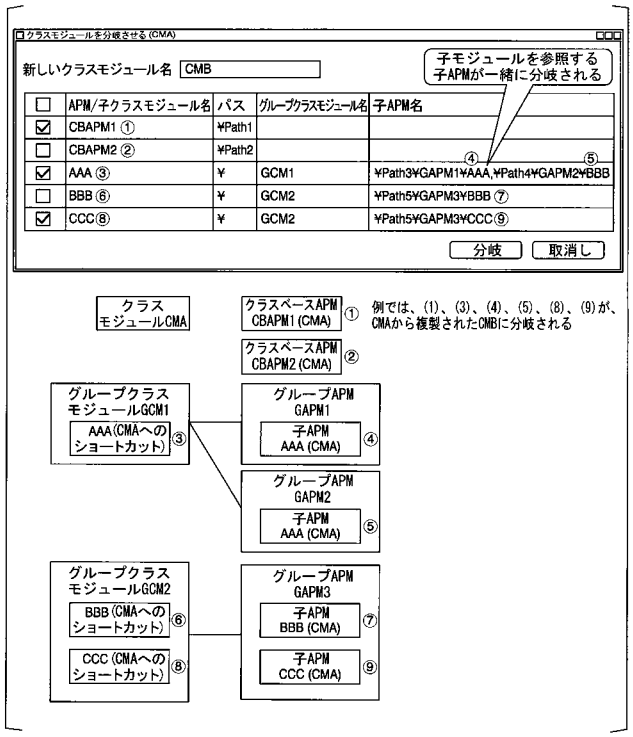
【図４】



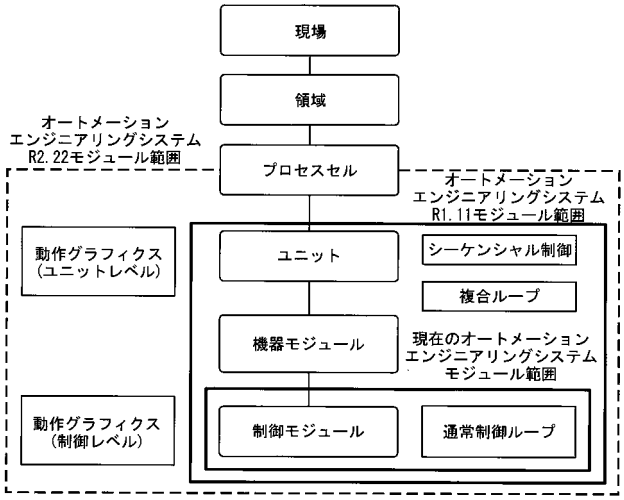
【図５】



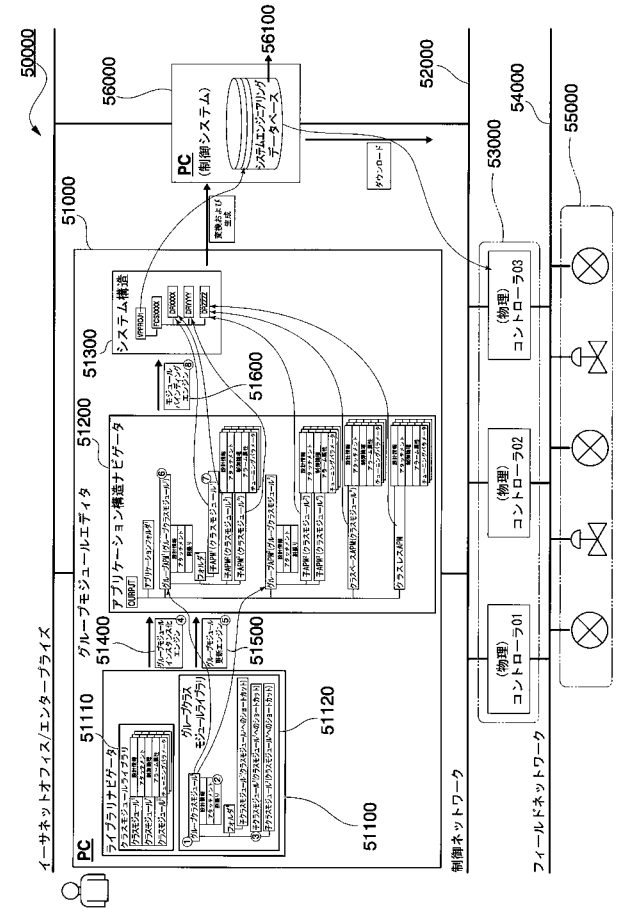
【図６】


クラスモジュールを分岐させる (CMA)
新しいクラスモジュール名 CMB
子モジュールを参照する
子APMが一緒に分岐される
APM/子クラスモジュール名
パス
グループクラスモジュール名
子APM名
CBAPM1 ①
¥Path1
CBAPM2 ②
¥Path2
AAA ③
¥
GCM1
¥Path3¥GAPM1¥AAA,¥Path4¥GAPM2¥BBB
BBB ⑥
¥
GCM2
¥Path5¥GAPM3¥BBB ⑦
CCC⑧
¥
GCM2
¥Path5¥GAPM3¥CCC⑨
分岐
取消し
クラス
モジュールCMA
クラスベースAPM
CBAPM1 (CMA) ①
例では、(1)、(3)、(4)、(5)、(8)、(9)が、
CMAから複製されたCMBに分岐される
クラスベースAPM
CBAPM2 (CMA) ②
グループクラス
モジュールGCM1
AAA(CMAへの
ショートカット) ③
グループAPM
GAPM1
子APM
AAA (CMA) ④
グループAPM
GAPM2
子APM
AAA (CMA) ⑤
グループクラス
モジュールGCM2
BBB (CMAへの
ショートカット) ⑥
CCC (CMAへの
ショートカット) ⑧
グループAPM
GAPM3
子APM
BBB (CMA) ⑦
子APM
CCC (CMA) ⑨

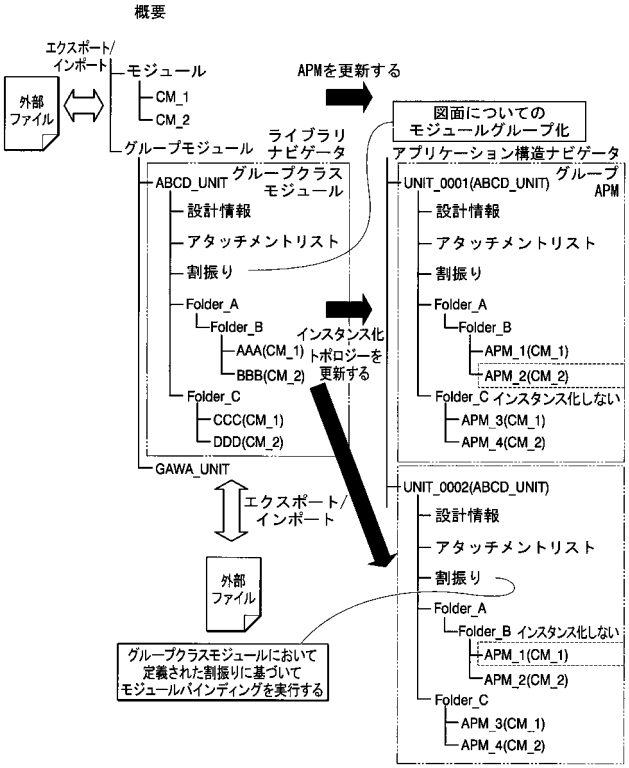
【図７】


現場
領域
オートメーション
エンジニアリングシステム
R2.22モジュール範囲
プロセスセル
オートメーション
エンジニアリングシステム
R1.11モジュール範囲
動作グラフィクス
(ユニットレベル)
ユニット
シーケンシャル制御
複合ループ
機器モジュール
現在のオートメーション
エンジニアリングシステム
モジュール範囲
動作グラフィクス
(制御レベル)
制御モジュール
通常制御ループ

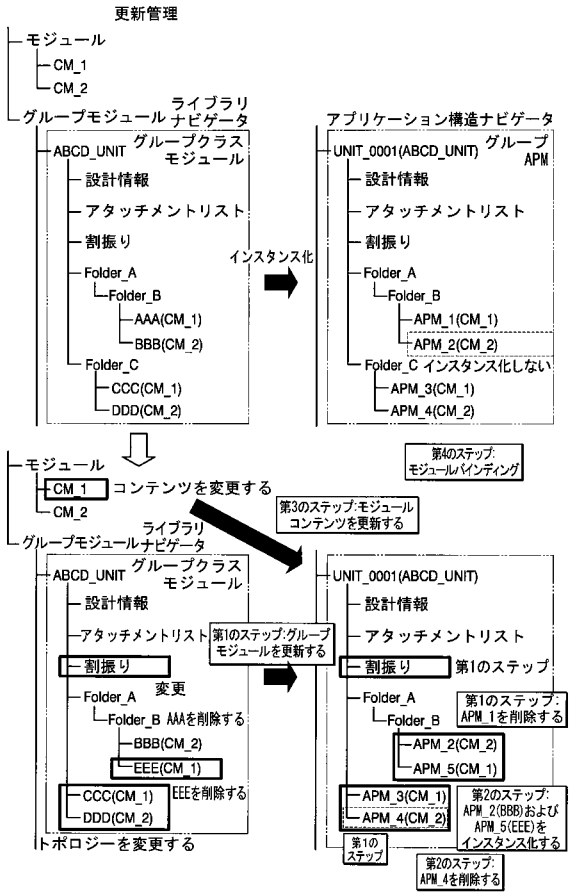
【図８】


50000
56000
56100
52000
54000
55000
53000
PC
(制御システム)
システムエンジニアリング
データベース
ダウンロード
51000
システム構造
51300
51600
51200
アプリケーション構造ナビゲータ
グループモジュールエディタ
51400
51500
51110
51120
51100
ライブラリナビゲータ
グループクラス
モジュールライブラリ
(物理)
コントローラ03
(物理)
コントローラ02
(物理)
コントローラ01
イーサネットオフィス/エンタープライズ
制御ネットワーク
フィールドネットワーク

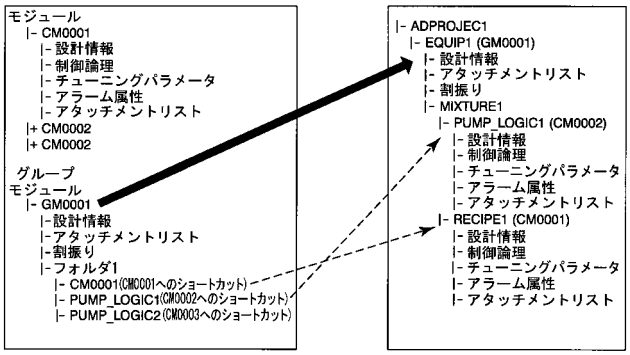
【図９】


概要
エクスポート/
インポート
外部
ファイル
モジュール
CM_1
CM_2
グループモジュール
APMを更新する
図面についての
モジュールグループ化
ライブラリ
ナビゲータ
グループクラス
モジュール
アプリケーション構造ナビゲータ
ABCD_UNIT
設計情報
アタッチメントリスト
割振り
Folder_A
Folder_B
AAA(CM_1)
BBB(CM_2)
Folder_C
CCC(CM_1)
DDD(CM_2)
インスタンス化
トポロジーを
更新する
GAWA_UNIT
エクスポート/
インポート
外部
ファイル
グループクラスモジュールにおいて
定義された割振りに基づいて
モジュールバインディングを実行する
UNIT_0001(ABCD_UNIT)
グループ
APM
APM_1(CM_1)
APM_2(CM_2)
Folder_C インスタンス化しない
APM_3(CM_1)
APM_4(CM_2)
UNIT_0002(ABCD_UNIT)
Folder_B インスタンス化しない

【図１０】


更新管理
モジュール
CM_1
CM_2
ライブラリ
グループモジュール ナビゲータ
ABCD_UNIT
グループクラス
モジュール
設計情報
アタッチメントリスト
割振り
Folder_A
Folder_B
AAA(CM_1)
BBB(CM_2)
Folder_C
CCC(CM_1)
DDD(CM_2)
インスタンス化
アプリケーション構造ナビゲータ
UNIT_0001(ABCD_UNIT)
グループ
APM
設計情報
アタッチメントリスト
割振り
Folder_A
Folder_B
APM_1(CM_1)
APM_2(CM_2)
Folder_C インスタンス化しない
APM_3(CM_1)
APM_4(CM_2)
第4のステップ:
モジュールバインディング
モジュール
CM_1
コンテンツを変更する
CM_2
第3のステップ:モジュール
コンテンツを更新する
ライブラリ
グループモジュールナビゲータ
ABCD_UNIT
グループクラス
モジュール
設計情報
アタッチメントリスト
第1のステップ:グループ
モジュールを更新する
割振り
変更
Folder_A
Folder_B AAAを削除する
BBB(CM_2)
EEE(CM_1)
CCC(CM_1)
EEEを削除する
DDD(CM_2)
トポロジーを変更する
UNIT_0001(ABCD_UNIT)
設計情報
アタッチメントリスト
割振り
第1のステップ
Folder_A
Folder_B
第1のステップ:
APM_1を削除する
APM_2(CM_2)
APM_5(CM_1)
APM_3(CM_1)
APM_4(CM_2)
第2のステップ:
APM_2(BBB)および
APM_5(EEE)を
インスタンス化する
第1の
ステップ
第2のステップ:
APM_4を削除する

【図１１】


モジュール
|- CM0001
|- 設計情報
|- 制御論理
|- チューニングパラメータ
|- アラーム属性
|- アタッチメントリスト
|+ CM0002
|+ CM0002
グループ
モジュール
|- GM0001
|-設計情報
|-アタッチメントリスト
|-割振り
|-フォルダ1
|- CM0001(CM0001へのショートカット)
|- PUMP_LOGIC1(CM0002へのショートカット)
|- PUMP_LOGIC2(CM0003へのショートカット)
|- ADPROJEC1
|- EQUIP1 (GM0001)
|- 設計情報
|- アタッチメントリスト
|- 割振り
|- MIXTURE1
|- PUMP_LOGIC1 (CM0002)
|- 設計情報
|- 制御論理
|- チューニングパラメータ
|- アラーム属性
|- アタッチメントリスト
|- RECIPE1 (CM0001)
|- 設計情報
|- 制御論理
|- チューニングパラメータ
|- アラーム属性
|- アタッチメントリスト

フロントページの続き

(72)発明者　村田　秀樹
シンガポール・４６９２７０・シンガポール・ベドク・サウス・ロード・５・ヨコガワ・エンジニアリング・アジア・ピーティーイー・リミテッド内
(72)発明者　マーク・アンソニー・デ・カストロ・チューアンジエン
シンガポール・４６９２７０・シンガポール・ベドク・サウス・ロード・５・ヨコガワ・エンジニアリング・アジア・ピーティーイー・リミテッド内
(72)発明者　ナイン・オオ・リン
シンガポール・４６９２７０・シンガポール・ベドク・サウス・ロード・５・ヨコガワ・エンジニアリング・アジア・ピーティーイー・リミテッド内
(72)発明者　アーチー・サンビタン・オリド
シンガポール・４６９２７０・シンガポール・ベドク・サウス・ロード・５・ヨコガワ・エンジニアリング・アジア・ピーティーイー・リミテッド内
(72)発明者　エフェンディ・スピマン
シンガポール・４６９２７０・シンガポール・ベドク・サウス・ロード・５・ヨコガワ・エンジニアリング・アジア・ピーティーイー・リミテッド内
(72)発明者　ティン・サパル・ユ
シンガポール・４６９２７０・シンガポール・ベドク・サウス・ロード・５・ヨコガワ・エンジニアリング・アジア・ピーティーイー・リミテッド内
(72)発明者　ジャネット・トリア
シンガポール・４６９２７０・シンガポール・ベドク・サウス・ロード・５・ヨコガワ・エンジニアリング・アジア・ピーティーイー・リミテッド内

Ｆターム(参考) 5H220 AA01　BB07　BB12　CX04　JJ12

【外国語明細書】
2020155102000001.pdf