TAPINSYSTEMS

クラウドサービスの自動化

ペトリネットのモデリング技術を使った高速配備とメンテナンスの容易化

Tap In Systems 創業者 ― ピーター・ロー
2009 年　11 月 12 日
Email: ploh@tapinsystems.com
Web:　www.tapinsystems.com

クラウドサービスの自動化

クラウドコンピューティングサービスの出現は、IT 部門が彼らの顧客にサービスを提供する方法に更なる選択肢を与えました。Amazon Web Services などのクラウドベンダーのサービスは、低コスト、短い配備時間などの利益をもたらしました。無限のクラウドコンピューティング資源は、アプリケーション処理の負荷に対し「オンデマンド」で処理能力を追加するのに使うことができます。オフサイトのクラウドストレージは、オンサイトのストレージの「無限容量」のデータバックアップを提供します。この論文では、これらのクラウドサービスを企業の IT プロセスに効果的に統合するための Tap IN System のアプローチを説明します。

システム自動化の課題

ほとんどの企業では IT サービスは独立して配備されていません。IT サービスは、時間をかけて発展してきた人とプロセスおよびテクノロジーとの複雑な相互関係の一部です。クラウドサービスの適用可能性を評価する場合、人は必ず、これらのサービスが組織内の現在の IT プロセスとポリシーに適合するかをたずねるでしょう。どうやったら IT 運用はこれらの新しいサービスを効果的に管理できるか？

クラウドのベンダーが彼らのサービスを配備するために奨励している方法は、プログラム言語の API です。彼らは、エンドユーザ、サードパティーのベンダーやコンサルティング会社が有用なアプリケーションにクラウドサービスを統合するのに、これらの API を使うことができるのを期待しています。それはすべて単純なプログラミングの問題です。しかし、歴史的に、効率を最適化し、稼働率を向上させ、オペレータの作業負荷を削減する方法としてのシステム自動化の開発は困難でした。システム自動化はいつも、下記の特徴を持つソフトウェアのアプリケーション開発プロジェクトとして進められてきました。

- プロジェクトチームは、エンドユーザとオペレータ、システム管理者およびアプリケーションプログラマで構成されます。ある種のシステム管理ツールは、言語やプロトコル、API の標準化により、このプロセスを容易にする可能性がありますが、それでもコードを書くことになります。
- IT 運用部門の必要から要件が導かれますが、全体システムの設計は、その構築方法を知っているプログラマによって行われます。IT 運用部門は自動化がどのように働くべきかについての運用経験を持ちますが、ソフトウェア開発のスキルを持つ者は稀です。なせなら、ソフトウェアを書くには実践的な IT 運用には関係ない専門知識－コード言語知識、コーディングツール、リポジトリ使用法、コーディングのベストプラクティス、ビルドの手法、が必要だからです。

- ほとんどのソフトウェア開発プロジェクトでは、新しいアプリケーションの開発に時間がかかります。IT 部門は継続的に発展しているので、開発プロセス途中での要件変更がプロジェクトの再スタートを引き起こします。
- 開発後、アプリケーションの元々の目的が達成された後でも、プロセスの変更は自動化アプリケーションの変更を必要とします。このために、些細な変更であったとしても、元々のチームメンバーの関与が必要となるでしょう。メンテナンスのオーバヘッドシステムは、システムの自動化の継続性と一貫性を達成することが困難である主な理由のひとつです。
- 自動化が導入される際には、運用グループ、サービス提供責任者と、サービスに影響するイベントに対応するプログラムである自動化アプリケーションの間に信頼要素が確立している必要がある。これなしには、運用グループは自動化が信用できなくて障害の影響が破滅的だと考えるため、堅牢な自動化アプリケーションであっても配備されないでしょう。

ソフトウェアベンダーは、コーディングなしに使用できる製品を提案することで自動化システムの管理を支援するツールを提供します。しかし、それらのアプリケーションは一般的に下記のようです：

- ***脆い。*** これらのベンダーは自動化シナリオを自分のアプリケーションのコード内にカプセル化しようとし、ユーザにその入力パラメータをカスタマイズできるようにします。これらのツールの実行可能性は、ユーザの定義がうまくできているのに依存します。入力パラメータ以外の変化は、アプリケーションで無視されるでしょう。各々の企業の IT プロセスは自動化プロセスと緊密に強調しているので、ベンダーの自動化はカスタマイズ輔弼用性が高く、上述の問題を引き起こしやすいでしょう。
- ***ベンダーの製品全体を必要とする。*** 一般的に大きなソフトウェアベンダーは、サーバ管理、ネットワーク管理、アプリケーション管理にためのさまざまな製品ー その各々のサブセットの付属製品、たとえば Windows 監視、Linux 監視、Web アプリケーション監視、データベース監視などを持ちます。システム自動化は、さまざまのコンポーネント、テクノロジーやアプリケーション・プログラミング・インタフェースに亘る条件を必要とするので、顧客が個別業務を実行するそのベンダーの製品をすべて使っていれば、カスタマイズは容易化（売り込みも最適化）できます。
- ***高額である。*** これらのアプリケーションを開発は複雑に入り組んでいるため、これらのツールのコストは直ぐに数十万ドル（数千万円）規模になってしまいます。

クラウド内のシステム自動化

クラウドサービスは、とりわけ自動化が困難です。クラウドサービスをサポートするほとんどのアプリケーションはベンダーの API 固有であり、そのため企業が有用だと考えられるアプリケーションは、異機種混合システムにわたるアクセスとコントロールが必要です。

これらのアプリケーションが持つ機能の例として下記があります。

- ***クラウドのリソース・リカバリーの自動化。*** Amazon EC2 などのクラウドコンピューティングの価値はプログラムの制御下でスタートできることです。仮想化されたシステムはダイナミックにリソースを割り当てることができます。リカバリー自動化には、システムアラートに気がつく監視システムと、リソースをリスタートとする、クラウドサービス API 間の相互作用が必要です。IT プロセスへの結び付きには、問題チケットの作成やリカバリー発生時のサービスレベルをトラッキングするための追跡運用担当者への通知が含まれます。
- ***コンピュートクラウドをベースとした自動スケーリング。*** これはリカバリーシナリオのバリエーションです。この場合、自動スケーリング（スケールアップとダウン）のアクションはアプリケーション負荷に依存します。負荷の計算は、明示的あるいは暗黙に監視システム計測基準から提供され、クラウド API を使ってコンピュートインスタンスをスタートするトリガーとなります。 他のアプリケーションコンポーネントとの統合の相互作用がさらなる複雑さを生じさせます。たとえば、Web サーバ機能を追加する場合、追加 Web サーバ資源を定義するためにロードバランサーやアプリケーションコンポーネントを再構成する必要があります。構成変更を構成データベースに反映する必要がある IT プロセスであれば、自動スケーリングプロセスは構成データベースと相互作用する必要があります。
- ***クラウドストレージのバックアップ。***クラウドストレージは理論的には「無限」で施設外にあるため、データセンターストレージのバックアップの良い候補です。自動化アプリケーションは、クラウドストレージへの転送を開始するため、監視システムからのストレージ容量をトリガーとするアラート、アプリケーション制御コマンドおよび、クラウド API アクションを連携させる必要があります。重要なアプリケーションデータを追跡監視するデータストレージのアプリケーションに通知する必要があります。
- ***ハイブリッドアプリケーション。*** 企業 IT サービスは、ハイブリッドなアプリケーション ― クラウド内と自社のデータセンター内にリソースを持つアプリケーションよって最もよく提供されるでしょう。たとえば、スケーラブルなコンポーネントのための、クラウドコンピュートサービスは、高速なネットワーク接続を最適化したデーセンター内のコンポーネントと

組み合わせられるでしょう。クラウドと非クラウドのコンポーネント間の相互関係が自動化アプリケーションによって管理されます。

- ***ベンダー横断的サービス。*** クラウドサービスを使用するデメリットのひとつに、ベンダーサービスがシングル・ポイント障害になる可能性があります。それを緩和するためのひとつの方法は、複数のクラウドサービスにリソースを分散することです。自動化アプリケーションは、下記を含む、これらサービスにまたがるアクティビティをコーディネートする必要があります：:
    - 複数のクラウドサービス API とインタフェースする
    - ネットワークやコンポーネント障害を対比し、ベンダーの障害を認知する。
    - 一方の冗長クラウドサービスと協調するためコンポーネントを再構成する。
    - 両方のサービスでの変更と障害シナリオの計画と対処。

クラウドサービスは、基本的にデータセンターの拡張なので、クラウドの自動化アプリケーションには、システムの自動化アプリケーションと同様な課題に、複数のベンダーのインタフェースに対応する複雑さが加わります。

Tap In Systems のアプローチ

Tap In System は下記を目標として、システム自動化アプローチを開発しました：

- システム自動化の開発プロセスを容易にする
- システム自動化アプリケーションの保守を容易にする
- 自動化のオーナーシップを IT 運用部門に渡す

システム管理ツールは 30 年以上も前から使われているため、自動化の開発問題は既に存在していました。これらの問題に取り組むには、IT 運用部門に自動化のオーナーシップを移すという考え方の変革必要です。それは、組織の観点でいえば、運用の経験がある部門に自動化のオーナーシップがあるということです。しかし、システム自動化の開発の業務には、多くのプログラム専門知識が必要で、開発はアプリケーション開発グループに移ってしまいます。

これらの責任を正しく配置するため、システム自動化のプロセスはふたつの部分つまり、運用部門が所有する*モデリング*、と開発部門が所有する*プログラミング*に分割されるべきです。歴史的に、システム自動化プロセスは、運用手順やランブック（run book）ドキュメントに記述されてきました。これらの手順をモデリングすることは、これらのプロセスを英語の文書よりも正確かつ科学的な方法で記述する必要があります。Tap In のやり方は、これらのプロセスをペトリネットのモデルで定義する ことです。

ペトリネットの情報やチュートリアルには下記のリンクを参照してください：

- Petri Net Worlds - http://www.informatik.uni-hamburg.de/TGI/PetriNets/
- Wikipedia - http://en.wikipedia.org/wiki/Petri_net
- Application of Petri Nets to Workflow Management - http://is.tm.tue.nl/staff/wvdaalst/publications/p53.pdf

ペトリネットは複雑なシステム運用を記述するための証明されたメソッドです。広範囲にわたる技術分野において、ペトリネットの適用性を調べる数十年間の研究が行われてきました。 IT プロセスでは、ペトリネットはプログラミング経験がなくても、どのように運用プロセスが動くべきかを効率的に記述できます。それらは英語ベースのドキュメントよりも厳密な定義が必要ですが、そのための詳しいナレッジは運用部門内にあります。モデルが定義されると、それらはコーディングのためにアプリケーション開発者に渡されます。

また、メンテナンスの変更も、モデルを変更することにより運用側でコントロールされます。この方法論は、特定の変更につき開発者が実施をするに十分な定義がなされるまで、アプリケーション開発者の関与を減らします。全体として、より早く容易な開発プロセスとメンテナンスという結果になります。この方法論では、運用のより高度な分析経験（プログラム経験は不要です）と、組織オーナーシップのよりよい配置が必要です。

Tap In の製品は、このプロセスを下記の方法で実現します。

- モデルを作成、保存、実行できるファイルにコード化。
- モデルの作成、デバッグと検証のためのグラフィカルなツールを提供することでモデル化プロセスを容易化。
- モデルをリアルタイムの監視イベントと結び付けー実質的にドキュメントから実行可能なタスクに変換することを可能にする。
- オペレータがモデルの実行と結果を見られる。
- モデルをコード開発と実行にリンクする、プログラミングのフレームワークを提供する。
- モデルが実稼動環境でスケジュールおよび実行される、実行フレームワークを提供する。

これらの機能は、Tap In の Control Plan Editor 製品に導入されています。そのグラフィカルインタフェースにより、コントロールプラン（Control Plan）と呼ぶ、Tap In バージョンのペトリネットモデルを容易に作成することができます。コントロールプランは、モデル内の各プレース（Place）で業務を実行することができるペトリネットモデルの拡張です。この業務はクラウド API 呼び出し、データベースの値読み出しやリカバリーアクションの実施などの自動化タスクです。プレースのタスクは Tap In Systems によってあらかじめパッケージされており、ユ

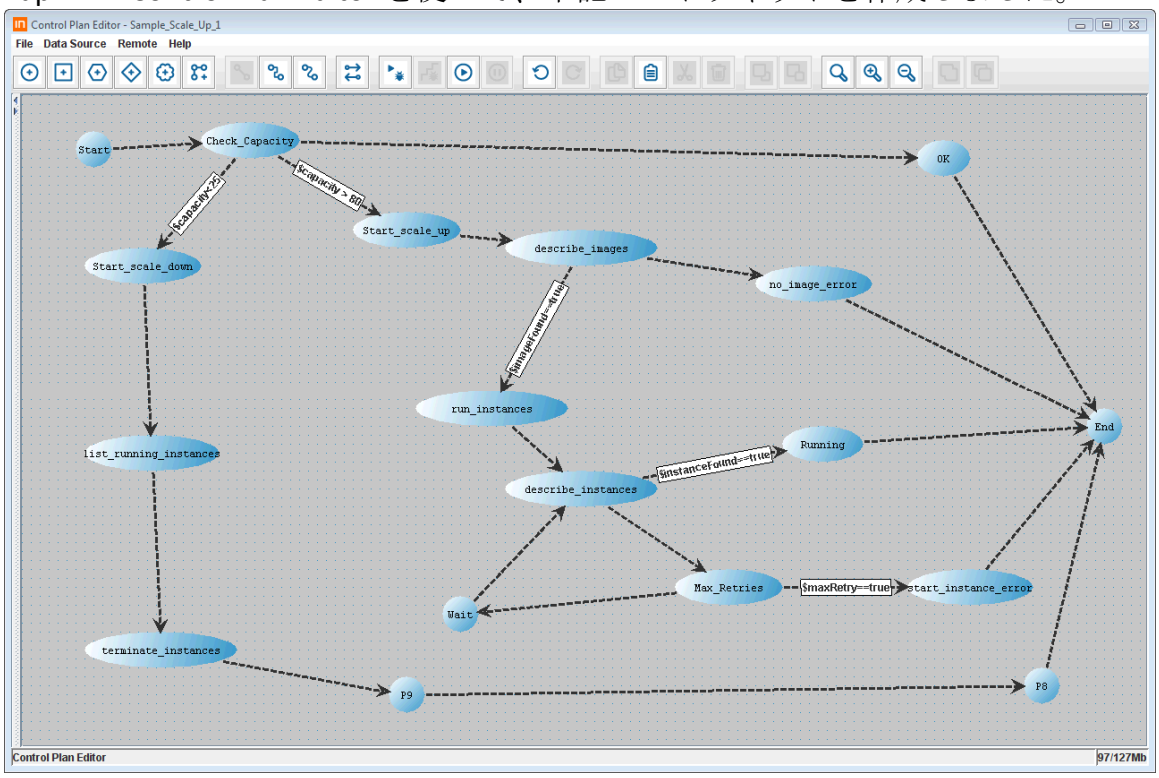
ーザによるカスタマイズができます。ネット内のトリガーは、Tap In のクラウド管理サービス (CMS)からのイベントの調査などの「業務の結果によって起動することができます。

Tap In のコントロールプランは実行された場合、現在にネット状態とどのプレースが現在実行されているかを反映して、ネットモデルが色を変えます。コンソール出力により、業務タスクが詳細なステータス情報をプリントできます。

クラウドの自動スケーリングの例

この方法を使ってどのようにシステム自動化が導入されるかの例を見ていきましょう。クラウドアプリケーションの共通的な要求に負荷に従ってサーバのスケーリングを自動化できることがあります。負荷（ユーザは負荷計算方法を定義できます）が閾値を超えたら、キャパシティを追加する。閾値を下回ったら、課やシティを削減する。

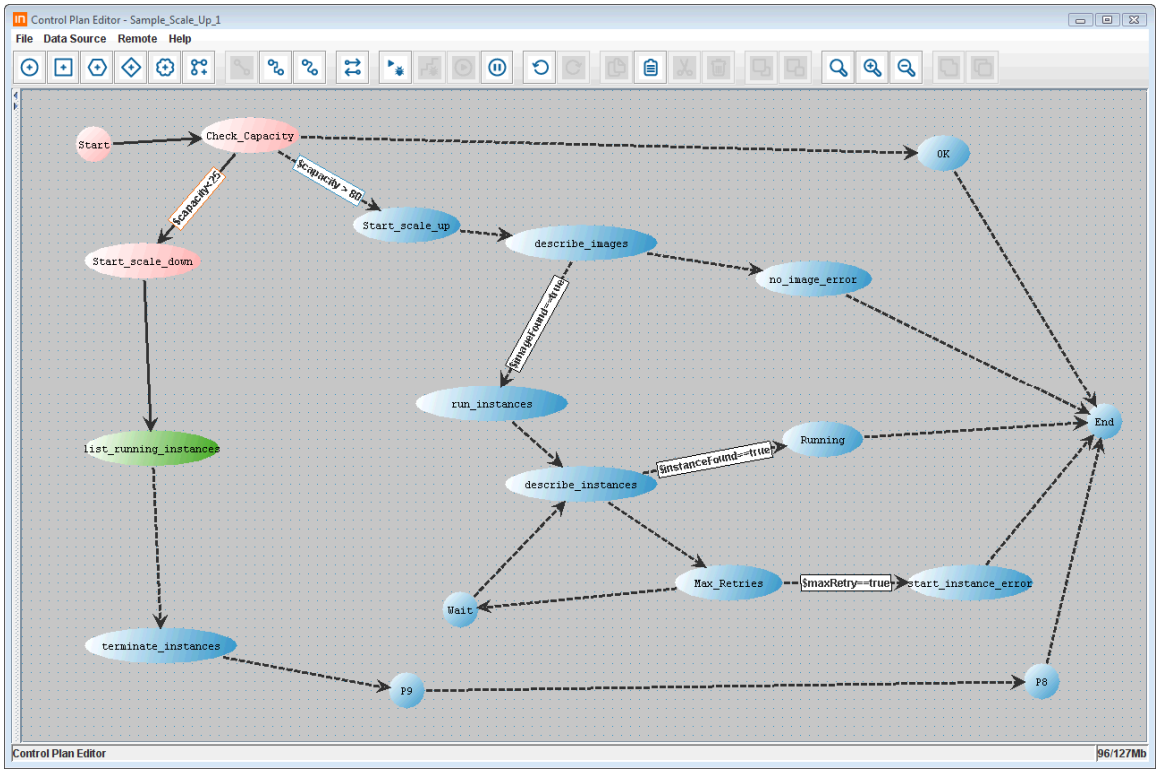
Tap In の Control Plan Editor を使って、下記のペトリネットを作成しました。



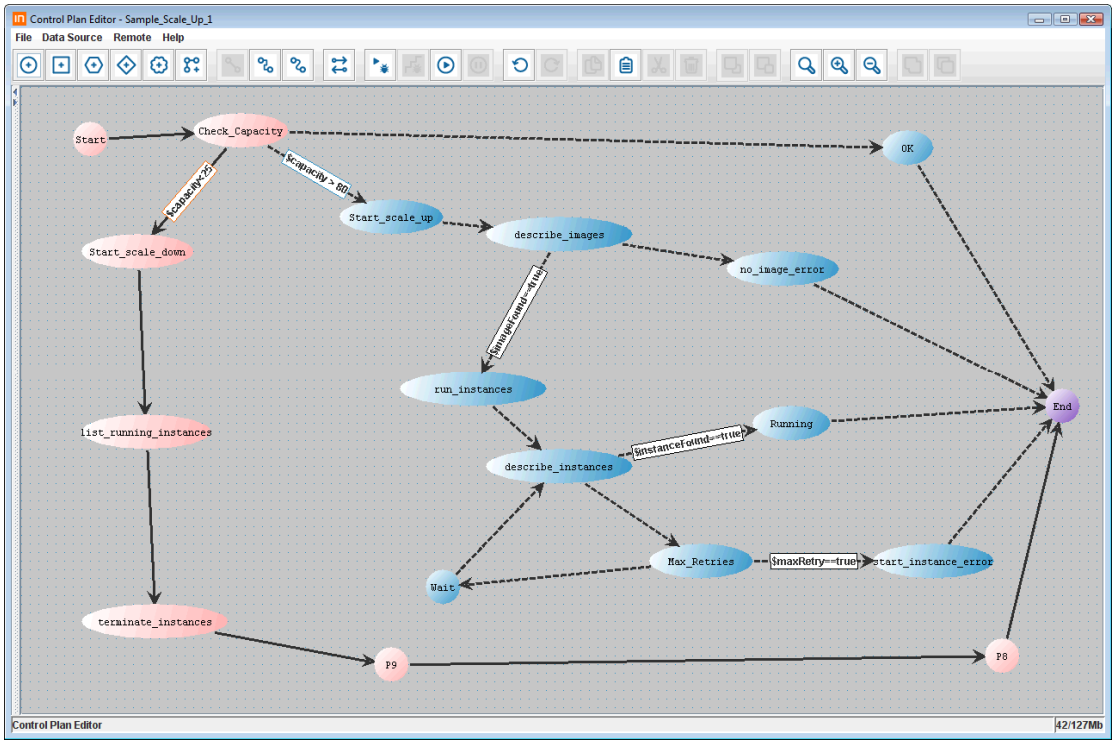
Control Plan Editor ユーザインタフェースでは、ドラック＆ドロップの操作でプレースを作成し、トランジションを接続できます。コントロールプランは自動スケーリングのワークフローを示します。プレースは下記の作業タスクを示します：

- Check_Capacity –Tap In 監視システムからの読み出しイベントで、負荷の閾値に達したことを示唆します。
- インスタンスを List（リスト表示）、run（実行）および terminate（終了）します。 これらは、クラウドシステムの稼動中コンピュートインスタンスをリストしたり、コンピュートインスタンスをスタートしたり、コンピュートインスタンスを終了させたりするために、 API 呼び出しを開始します。このプランでは、これらの API 呼び出しはプレースでカプセル化されているので、必要なら他のクラウドベンダーの呼び出しに容易に変更できます。このことから、社内 IT プロセスとベンダーのクラウドサービスのサポートとの分離が明らかです。
- エラーレポート。これらのプレースは、監視ツールや通知メソッドへのエラーメッセージを生成します。これにより、自動化と運用担当者間のコミュニケーションを保証します。

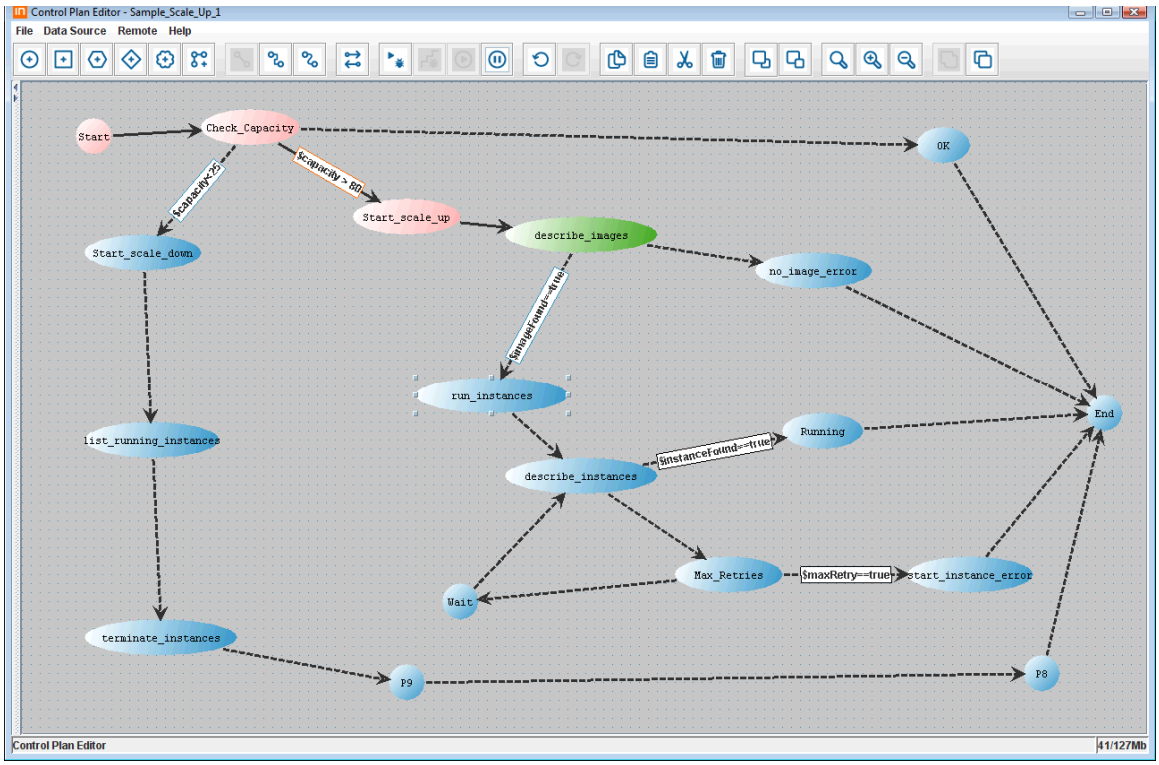
このプランを実行すると、コントロールプランは、プランのカラーコードによってどのタスクが実行されているかを表示します。下図のように、キャパシティの閾値よりも負荷が低かったのでスケールダウンのタスクが実行されました。このプランでは“terminate_instances” タスクがスケールダウン処理中で実施されます。



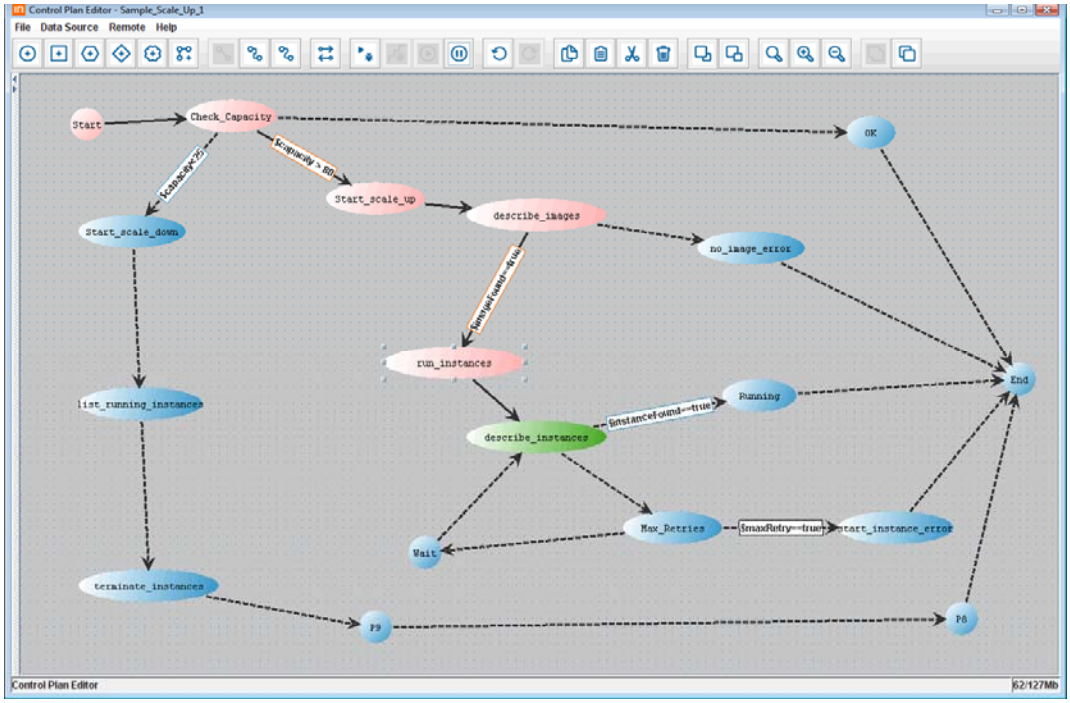
プランは、“end”プレースに着いたため、ここで完了しました。

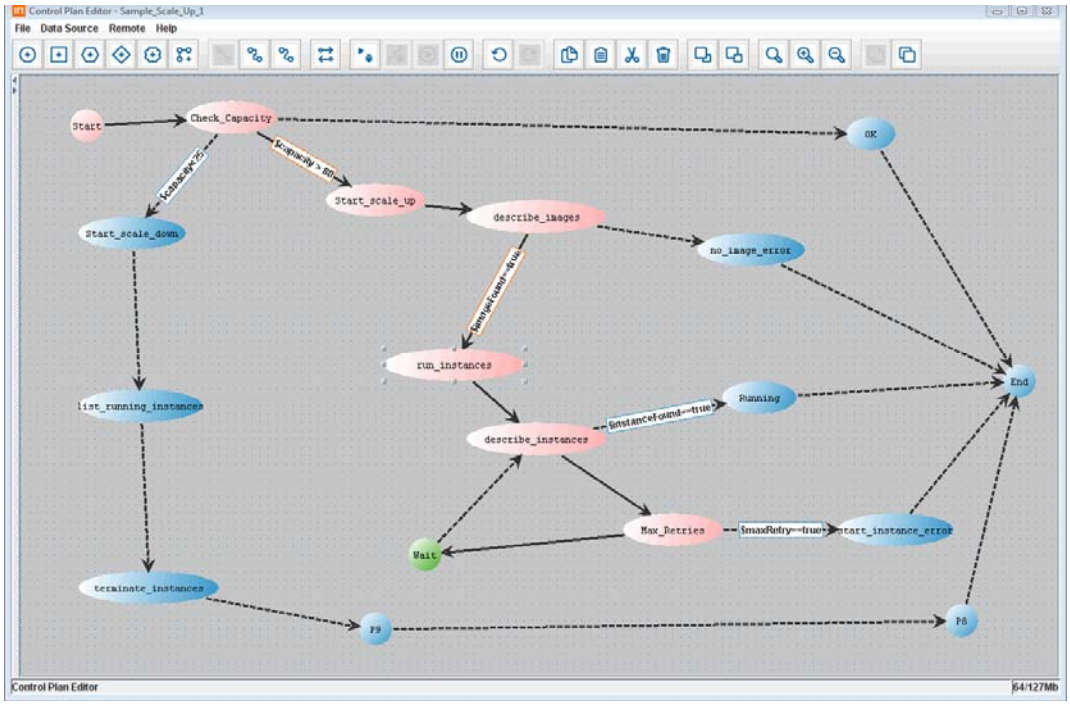


そのプランが実行された場合の他の可能性は、負荷がキャパシティを超え、スケールアップのプロセスを実行するべきだと示すことです。この場合、まず、そのプランは、スタートするよう要求されたイメージが実際に存在するかを見るため、すべてのイメージをリストアップ（“describe_images”）します。存在しなければ、このイメージのインスタンスがスタートされるでしょう。プランは、定期的にチェックして、そのインスタンスが稼動状態になるのを保証します。下の図は、プランが、既存イメージのチェックによりスケールアップのプロセスをスタートするのを示しています。

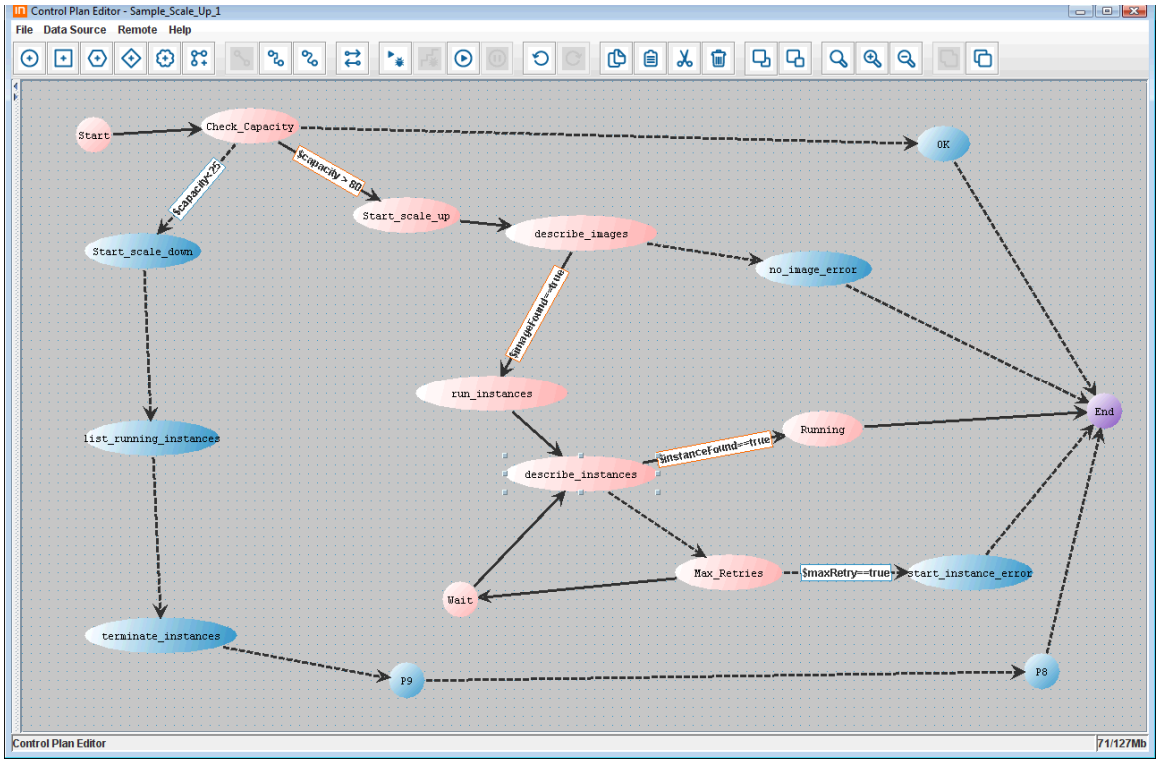

Control Plan Editor - Sample_Scale_Up_1
File Data Source Remote Help
Start
Check_Capacity
OK
$capacity<25
$capacity > 80
Start_scale_up
describe_images
Start_scale_down
no_image_error
$instanceFound==false
run_instances
End
list_running_instances
Running
$instanceFound==true
describe_instances
Max_Retries
$maxRetry==true
start_instance_error
Wait
terminate_instances
P9
P8
Control Plan Editor
41/127Mb

このケースでは、イメージが検出され、インスタンスがスタートされました。そのプランは、今、インスタンスが稼動（“running”）状態になるのを待っています。



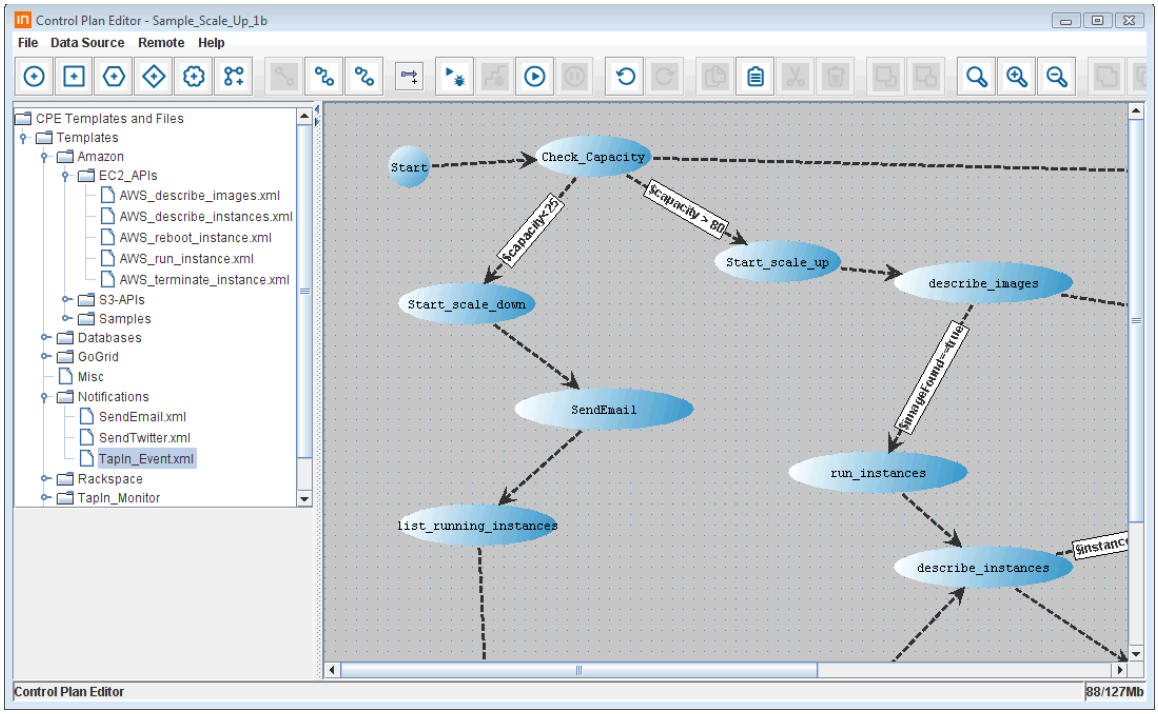


インスタンスは稼動状態になり、そしてインスタンスがうまく稼動した通知が、終了の前に送られます。



他の自動化アプリケーションでなく、コントロールプランを使用する利点は明白です。

1. コントロールプランのグラフィカルエディタを使うことにより、モデルが簡単に作れます。そのモデルはプログラム構成でなく、運用プロセスを表すため、経験のある運用担当者責任がそのプランを確認するのに適しています。
2. 各プレースのタスクは、あらかじめパッケージ化されたモジュールや（Ruby や Java）のコードです。アプリケーション開発者は、彼ら自身の専門エリアーAPI の実行とプログラムコンポーネント間の相互動作に集中することができます。
3. オペレータは、これらのアクションを自分で作成しテストしているので、信頼できます。上述のように、プランの実行状態を目で見ることもできます。 また、オペレータは、プラン中に適当な通知を挿入することで、自動化タスクが適切にステータスの通信をしていることを確認できます。
4. コントロールプランの保守はもっと簡単です。定義済みのタスクはプランの変更によって追加できます。下記の画面図は、プランで使用可能なテンプレートを示します。



モデルにプレースのタスクを追加することは、ユーザはテンプレートをキャンバスにドラッグするだけでできます。たとえば、エラー発生時のｅメール通知を追加するには、email Notification テンプレートをキャンバスにドラッグしてネットのエラー（error）セクションの中に置くだけです。

この運用が自動化プランを保守するための能力は、最も重要なメリットです。新しいプレースタスクを追加するのに追加プログラミングが必要になった場合で自動化プラン全体を書き換える必要はありません。開発者は新しいプレースのタスクを作成し、オペレータがそれをモデルの中に置けるようにするだけです。

コントロールプランのモデルを使った開発方法は非常に柔軟です。オペレータは、関心のあるアプリケーション状態を定義し、それらの状態を判断するアクションとロジックを導入することにより、トップダウンのモデリング方法を使うことができます。基本的でハイレベルなモデルが開発されると、他のコントロールプランで個々のプレースを置き換えることにより、より複雑なロジックが使用できます。他のプラン内でプレースとして使用できるプランをスーパーノード（supernode）と呼びます。

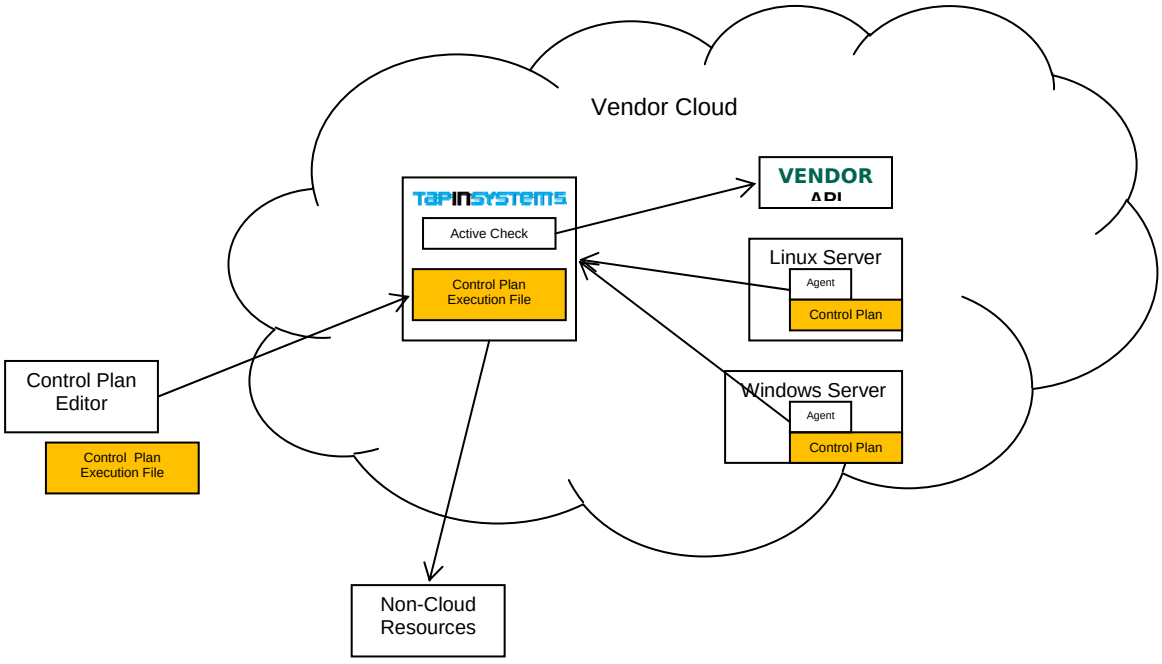
上述の例の中に、システムの現在負荷を調べることでキャパシティが超過しているかどうかを計算するプレースがありました。しかし、この判断ポイントは、単一の変数を見るだけよりも、もっと複雑かもしれません。たとえば、そのロジックは、サーバの負荷、現在のユーザ数、時刻、曜日、稼動中の Web サーバの現

在数や使用中のデータベース接続数などを参照するアルゴリズムを必要とするかもしれません。われわれは、それらの変数をカプセル化し、キャパシティが閾値を超えたか、下回ったか、それとも OK かを出力とするコントロールプランを作成できます。このプランは、上レベルのダイアグラム内での check_capacity プレースのスーパーノードとして定義できます。この機能によって複雑なモデルを作成するための、プランの階層化が行えます。さらに、望むならば、オペレータはその複雑さをハイレベルのプラン内で隠蔽することができます。

実稼動配備

コントロールプラン作成後、いくつかの方法で実稼働環境に配備して使用することができます。最も簡単な方法は、Control Plan Editor がそれらを実行できるので、オペレータは、そのプランの実行をオペレータ制御下でのみ行う選択が可能です。オペレータ運用手順書の手順がプランとして定義されていれば、オペレータはコントロールプランを、手動の問題診断/リカバリー手順を自動的に実行し、その結果をグラフィカルに見る方法として使用できます。ダイアログボックスなどのコントロールプランの機能は、この機能性を補強します。

しかし、コントロールプランはグラフィカルエディタと独立して。無人プログラムとして、コントロールプランサーバで実行することもできます。このシナリオを下図で示します。



コントロールプランは、どの分散ノード上でも、グラフィカルユーザインタフェースなしに、実行できます。もちろん、この方法で動くように設計されていれば、プランにはダイアログボックスなどのユーザインタフェース機能が含まれていてはいけませんし、Tap In 監視システムへのイベントのような他のメカニズムを介してオペレータにステータスを通知するべきです。この構成により、適切な場所、通常は問題の「発生源に隣接した場所」で自動処理を行わせることができます。複数のプランの実行をいろいろなシステム上でスケジュールできるため、各自動化シナリオのプランを実行するよう構成できます。

まとめ

この論文は、システム管理とクラウドアプリケーションの自動化のために新しいアプローチを概説しました。ペトリネットのモデリング手法を採用したことで、IT 運用担当者は自動化アプリケーションの 責任を持つことができ、その結果、開発が早く、メンテナンスが容易になります。希少なリソースであるアプリケーション開発者の関与は最低限で、より効果的に用いられます。運用手順をペトリネットのモデル内にコード化することで、複雑なプロセスを、段階を踏んだ厳密な方式で開発することができます。

Tap In の Control Plan Editor は、これらの自動化モデルを容易に定義し配備するのに使用できます。そのグラフィカルユーザインタフェースは 、これらのモデルの作成と変更を容易にします。リアルタイムの実行結果をみることもできます。これらのツールは、企業の IT プロセスとクラウドサービス機能を統合することで、クラウドサービスのメリットを活用できるようにします。

この論文で説明したトピックについての詳しい情報については、Tap In Systems の info@tapinsystems.com までお問い合わせください。