(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6617617号
(P6617617)

(45)発行日　令和1年12月11日(2019.12.11)　(24)登録日　令和1年11月22日(2019.11.22)

(51)Int.Cl.		F I		
G06F 11/34	(2006.01)	G06F	11/34	147
G06F 11/07	(2006.01)	G06F	11/07	151
		G06F	11/07	140A

請求項の数 7　(全 26 頁)

(21)出願番号　特願2016-47602(P2016-47602)
(22)出願日　平成28年3月10日(2016.3.10)
(65)公開番号　特開2017-162312(P2017-162312A)
(43)公開日　平成29年9月14日(2017.9.14)
審査請求日　平成30年12月10日(2018.12.10)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　110002147
特許業務法人酒井国際特許事務所
(72)発明者　木村　功作
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

審査官　多胡　滋

(56)参考文献　米国特許出願公開第２０１５／０３５０１７４（US，A1）

最終頁に続く

(54)【発明の名称】管理装置、管理プログラム及び管理方法

(57)【特許請求の範囲】
【請求項１】
アプリケーションが送信したＡＰＩ（Application Programming Interface）リクエストの履歴を示すリクエスト履歴情報と、前記アプリケーションの単位時間あたりのＡＰＩアクセス可能回数を示すアクセス回数管理表と、を記憶する記憶部と、
前記ＡＰＩリクエストを受信した場合に、該ＡＰＩリクエストを送信したアプリケーションについて、前記アプリケーションによる前記ＡＰＩリクエストの内容の遷移と前記リクエスト履歴情報とに基づいて安定度を算出する安定度算出部と、
前記安定度に応じて前記アプリケーションの前記ＡＰＩアクセス可能回数を変更する変更部と、
前記アプリケーションによる前記ＡＰＩリクエストがあった場合に、前記アプリケーションのアクセス回数と、前記アプリケーションのＡＰＩアクセス可能回数とを比較し、前記ＡＰＩリクエストの受付を行うか否かを判定する判定部と、
を有することを特徴とする管理装置。
【請求項２】
前記ＡＰＩリクエストを送信したアプリケーションについて、ログインリクエストからログアウトリクエスト或いはタイムアウトの間に前記ＡＰＩリクエストに応じて遷移した各状態間の遷移確率を示す状態遷移インスタンスを作成するインスタンス作成部をさらに有し、
前記記憶部は、前記リクエスト履歴情報を用いて前記アプリケーションごとに作成され

た各状態間の遷移確率を示す状態遷移表を記憶し、
前記安定度算出部は、前記ＡＰＩリクエストを送信したアプリケーションについて、前記状態遷移インスタンスと前記状態遷移表とを比較して前記安定度を算出することを特徴とする請求項１に記載の管理装置。
【請求項３】
前記安定度算出部は、前記状態遷移インスタンスと前記状態遷移表との一致度に応じて前記安定度を増減し、
前記変更部は、前記安定度の増減に応じて前記ＡＰＩアクセス可能回数を増減することを特徴とする請求項２に記載の管理装置。
【請求項４】
前記安定度算出部は、前記状態遷移インスタンスが前記状態遷移表に示されていない状態を含む場合には前記安定度をより少なく算出し、前記状態遷移インスタンスの遷移確率と前記状態遷移表の遷移確率との一致度が所定値よりも低い場合には前記安定度をより多く算出し、
前記変更部は、前記アプリケーションについて、前記安定度がより少ない場合には前記ＡＰＩアクセス可能回数を前回設定値から所定値を減算した値に変更し、前記安定度がより多い場合には前記ＡＰＩアクセス可能回数を前回設定値のまま維持し、または、前記ＡＰＩアクセス可能回数を前回設定値に所定値を加算した値に変更することを特徴とする請求項３に記載の管理装置。
【請求項５】
前記変更部は、所定時間ごとに前記アプリケーションのアクセス回数と、前記アプリケーションのＡＰＩアクセス可能回数とを更新することを特徴とする請求項１～４のいずれか一つに記載の管理装置。
【請求項６】
コンピュータに、
ＡＰＩリクエストを受信した場合に、該ＡＰＩリクエストを送信したアプリケーションについて、前記アプリケーションの前記ＡＰＩリクエストの内容の遷移と、前記アプリケーションが送信したＡＰＩリクエストの履歴を示すリクエスト履歴情報とに基づいて安定度を算出し、
前記安定度に応じて前記アプリケーションのＡＰＩアクセス可能回数を変更し、
前記アプリケーションによる前記ＡＰＩリクエストがあった場合に、前記アプリケーションのアクセス回数と、前記アプリケーションの単位時間あたりのＡＰＩアクセス可能回数とに従って、前記ＡＰＩリクエストの受付を行うか否かを判定する
各処理を実行させることを特徴とする管理プログラム。
【請求項７】
管理装置は、
ＡＰＩリクエストを受信した場合に、該ＡＰＩリクエストを送信したアプリケーションについて、前記アプリケーションの前記ＡＰＩリクエストの内容の遷移と、前記アプリケーションが送信したＡＰＩリクエストの履歴を示すリクエスト履歴情報とに基づいて安定度を算出し、
前記安定度に応じて前記アプリケーションのＡＰＩアクセス可能回数を変更し、
前記アプリケーションによる前記ＡＰＩリクエストがあった場合に、前記アプリケーションのアクセス回数と、前記アプリケーションの単位時間あたりのＡＰＩアクセス可能回数とに従って、前記ＡＰＩリクエストの受付を行うか否かを判定する
処理を実行することを特徴とする管理方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、管理装置、管理プログラム及び管理方法に関する。
【背景技術】

【０００２】
Application Programming Interface（ＡＰＩ）は、外部のアプリケーション（アプリ）システムから、あるデータ及び機能を利用するために提供されるインタフェースであり、データの提供者或いは機能の開発者によって作成、公開される。現在では、HyperText Transfer Protocol（ＨＴＴＰ）でアクセス可能なＷｅｂ ＡＰＩの形でＡＰＩが公開されることが主流となっている。
【０００３】
近年、ＡＰＩ提供の利便性及び安全性の向上のために、ＡＰＩゲートウェイという種類の製品が発表されている。このＡＰＩゲートウェイは、ＡＰＩを提供するバックエンドサーバ（アプリサーバ）側にリバースプロキシの形で導入され、ＡＰＩの提供を管理する。外部アプリケーション或いはシステム開発者等のＡＰＩ利用者は、ＡＰＩゲートウェイを経由して、公開されているＡＰＩを使い、新製品の開発や、既存製品の機能の拡充を行うことができる。今後、企業がＡＰＩを通して自社製品、業務、データ等を多くの企業や個人と繋げて新たな価値を創造することが重要であると考えられている。この取り組みを推進する技術として、ＡＰＩゲートウェイが注目されている。
【０００４】
このＡＰＩゲートウェイは、ＡＰＩ利用者によるＡＰＩ（主にＷｅｂ ＡＰＩ）活用のために種々の機能を有する。具体的には、ＡＰＩゲートウェイは、各アプリケーションに対して、アプリケーションを認証し識別するためのＩＤであるＡＰＩキーを提供する。ＡＰＩゲートウェイは、このＡＰＩキーによって、アプリケーションごとに、ＡＰＩのＵＲＩやパラメータ等の管理、ＡＰＩごとの単位時間あたりのアクセス可能回数の上限値（レートリミット）やアクセス範囲の制御、呼び出し回数等の統計情報の記録や分析を行っている。
【０００５】
近年、ＡＰＩのレートリミット設定方法として、アプリケーションとデータとの優先度に応じてレートリミットを設定する制御方法が提案されている。提案の制御方法では、事前に定義された閾値を考慮し、予測アルゴリズムを用いて時間帯ごとに優先度を導出している。なお、従来、クライアントＰＣの操作ログと正規ログとのマッチング度に応じて、ユーザによる悪意のある操作を検知する方法が提案されている。
【先行技術文献】
【特許文献】
【０００６】
【特許文献１】米国特許出願公開第２０１４／０２８２６２６号明細書
【特許文献２】特開２００９－２０８１２号公報
【発明の概要】
【発明が解決しようとする課題】
【０００７】
ところで、ＡＰＩゲートウェイを用いた既存の業務システムのＡＰＩ化による業務効率化が求められている。既存の業務システムのＡＰＩ化では、元々のシステムの設計によっては、特殊な手順によるＡＰＩアクセスをアプリケーション側に提供する場合がある。この場合、ＡＰＩのアクセス手順によっては、システムに障害が生じる可能性がある。
【０００８】
また、アプリケーションに対してＡＰＩキーが発行された後に、アプリケーションが改版されてアプリケーションのアクセス手順が変化する場合がある。例えば、改版されたアプリケーションが、改版でバグを含むことによって、誤って、推奨されない手順でＡＰＩにアクセスしてしまう場合である。
【０００９】
さらに、アクセス手順が異なる別のアプリケーションから同じＡＰＩキーによってＡＰＩが利用される可能性もある。言い換えると、ＡＰＩキーが悪意のあるアプリケーションに使用される場合である。この場合、悪意のあるアプリケーションによって、ＡＰＩ提供

者が損害を被るようなデータ及び機能の利用が実行されるおそれがある。
【００１０】
しかしながら、従来の方法は、レートリミットを設定するためにアプリケーションとデータとの優先度を導出する手法であるため、システムに危害を与えるような、改版や悪意の可能性があるアプリケーションを検知することが困難であった。このため、従来の方法では、改版や悪意の可能性があるアプリケーションもＡＰＩへのアクセスが可能であったため、改版や悪意の可能性があるアプリケーションからＡＰＩで提供されるデータや機能を保護することが難しいという問題があった。なお、特許文献２記載の技術は、ユーザによる悪意のある操作の検知は可能であるものの、改版や悪意の可能性があるアプリケーションを検知することは困難であった。
【００１１】
本発明は、上記に鑑みてなされたものであって、ＡＰＩで提供されるデータや機能を保護することができる管理装置、管理プログラム及び管理方法を提供することを目的とする。
【課題を解決するための手段】
【００１２】
上述した課題を解決し、目的を達成するために、本発明に係る管理装置は、記憶部と、安定度算出部と、変更部と、判定部とを有する。記憶部は、アプリケーションが送信したＡＰＩリクエストの履歴を示すリクエスト履歴情報と、アプリケーションの単位時間あたりのＡＰＩアクセス可能回数を示すアクセス回数管理表と、を記憶する。安定度算出部は、ＡＰＩリクエストを受信した場合に、該ＡＰＩリクエストを送信したアプリケーションについて、アプリケーションによるＡＰＩリクエストの内容の遷移とリクエスト履歴情報とに基づいて安定度を算出する。変更部は、安定度に応じてアプリケーションのＡＰＩアクセス可能回数を変更する。判定部は、アプリケーションによるＡＰＩリクエストがあった場合に、アプリケーションのアクセス回数と、アプリケーションのＡＰＩアクセス可能回数とを比較し、ＡＰＩリクエストの受付を行うか否かを判定する。
【発明の効果】
【００１３】
本願の開示する管理装置の１つの態様によれば、管理装置は、ＡＰＩで提供されるデータや機能を保護することができる。
【図面の簡単な説明】
【００１４】
【図１】図１は、本実施例に係るＡＰＩ管理システムの構成を示す図である。
【図２】図２は、ＡＰＩゲートウェイの構成を示すブロック図である。
【図３】図３は、ＩＤ管理表のデータ構造の一例を示す図である。
【図４】図４は、ＡＰＩキー管理表のデータ構造の一例を示す図である。
【図５】図５は、ＡＰＩ一覧表のデータ構造の一例を示す図である。
【図６】図６は、リクエスト履歴管理表のデータ構造の一例を示す図である。
【図７】図７は、状態遷移表のデータ構成の一例を示す図である。
【図８】図８は、アクセス回数管理表のデータ構造の一例を示す図である。
【図９】図９は、第１アプリケーションについての各状態遷移の一例を説明するための図である。
【図１０】図１０は、第１アプリケーションについての各状態遷移の挙動の変化の一例を説明するための図である。
【図１１】図１１は、状態遷移インスタンスの一例を示す図である。
【図１２】図１２は、アクセス回数管理表の更新を説明する図である。
【図１３】図１３は、実施例に係るＡＰＩ管理処理の一例を示す図である。
【図１４】図１４は、実施例に係る新規アプリ登録処理の手順の一例を示すフローチャートである。
【図１５】図１５は、実施例に係るリクエスト受付可否判定処理の手順の一例を示すフロ

ーチャートである。
【図１６】図１６は、実施例に係る状態遷移インスタンス作成処理の手順の一例を示すフローチャートである。
【図１７】図１７は、安定度算出処理の手順の一例を示す図である。
【図１８】図１８は、レートリミット算出処理の手順の一例を示す図である。
【図１９】図１９は、ＡＰＩ管理プログラムを実行するコンピュータを示す図である。
【発明を実施するための形態】
【００１５】
以下に、本願の開示する管理装置、管理プログラム及び管理方法の実施例を図面に基づいて詳細に説明する。なお、この実施例は一例であり、構成等は限定しない。
【実施例】
【００１６】
［ＡＰＩ管理システムの一例］
本実施例に係る管理システムとして、ＡＰＩの提供を管理するＡＰＩ管理システムを例に説明する。図１は、本実施例に係るＡＰＩ管理システムの構成を示す図である。図１に示すように、ＡＰＩ管理システム１は、データ或いは機能を提供するアプリサーバ１０と、ＡＰＩゲートウェイ（管理装置）２０と、を有する。
【００１７】
アプリサーバ１０は、データの提供者或いは機能の開発者によって作成されたデータ或いは機能を提供するサーバ装置である。図１の例では、アプリサーバを１台としたが、これに限定されず、任意の数とすることができる。
【００１８】
ＡＰＩゲートウェイ２０は、アプリサーバ１０側にリバースプロキシの形で導入されたサーバ装置であり、アプリサーバ１０が提供するデータ或いは機能をＡＰＩ化し、これらのＡＰＩの提供を管理する。ＡＰＩゲートウェイ２０は、アプリサーバ１０と、ネットワークを介して通信可能に接続される。このネットワークの形態としては、有線または無線を問わず、インターネット（Internet）、ＬＡＮ（Local Area Network）やＶＰＮ（Virtual Private Network）などの任意の通信網が挙げられる。
【００１９】
ＡＰＩゲートウェイ２０は、ＡＰＩを利用する複数のアプリケーション（例えば、第１アプリケーション３０ａ～第３アプリケーション３０ｃ）と、ネットワーク等を介して通信可能に接続する。したがって、ＡＰＩゲートウェイ２０は、ＡＰＩを利用するアプリケーションとアプリサーバ１０との間に設けられる。ＡＰＩゲートウェイ２０は、各アプリケーションに対し、アプリケーションのＩＤを対応付けたＡＰＩキーを付与し、アプリケーションの認証を、このＡＰＩキーを用いて実行している。
【００２０】
各アプリケーションは、多数のユーザによって利用されている。例えば、第１アプリケーション３０ａは、ユーザＡ～Ｃが利用しており、第２アプリケーション３０ｂは、ユーザＤ，Ｅが利用している。なお、ＡＰＩゲートウェイ２０は、ユーザ認証も実施している。
【００２１】
ここで、ＡＰＩゲートウェイ２０は、ＡＰＩキーを用いて各種処理を行っている。具体的には、ＡＰＩゲートウェイ２０は、ＡＰＩキーを用いて、アプリケーションごとに、ＡＰＩのＵＲＩ（Uniform Resource Locator）やパラメータ等の管理を行っている。また、ＡＰＩゲートウェイ２０は、ＡＰＩキーを用いて、ＡＰＩごとのアクセス可能回数の上限値やアクセス範囲の制御、呼び出し（ＡＰＩリクエスト）回数等の統計情報の記録や分析等を行っている。
【００２２】
そして、ＡＰＩゲートウェイ２０は、ＡＰＩリクエストの挙動が過去の挙動と比して変化したアプリケーションについては、単位時間あたりのＡＰＩアクセス可能回数の上限値

（レートリミット）を下げる。言い換えると、ＡＰＩゲートウェイ２０は、改版アプリケーションや悪意アプリケーションに変化した可能性があるアプリケーションに対して、ＡＰＩアクセスを制限する。これによって、ＡＰＩゲートウェイ２０は、アプリケーションが改版アプリケーションや悪意アプリケーションに変化してもＡＰＩを保護している。以下、ＡＰＩゲートウェイ２０について説明する。

【００２３】

［ＡＰＩゲートウェイの構成］

図２は、図１に示すＡＰＩゲートウェイ２０の構成を示すブロック図である。図２に示すように、ＡＰＩゲートウェイ２０は、通信インタフェース（Ｉ／Ｆ）部２１、記憶部２２及び制御部２３を有する。

【００２４】

通信Ｉ／Ｆ部２１は、他の装置との間で通信制御を行うインタフェースである。通信Ｉ／Ｆ部２１は、ネットワークを介して他の装置と各種の情報を送受信する。例えば、通信Ｉ／Ｆ部２１は、アプリケーションからＡＰＩの利用申請やＡＰＩリクエストに関する各種の情報を受信する。また、通信Ｉ／Ｆ部２１は、アプリサーバ１０から提供対象のデータ或いは機能に関する各種の情報を受信する。通信Ｉ／Ｆ部２１としては、ＬＡＮカードなどのネットワークインタフェースカードを採用できる。

【００２５】

記憶部２２は、フラッシュメモリなどの半導体メモリ素子、ハードディスク、光ディスクなどの記憶装置である。なお、記憶部２２は、ＲＡＭ（Random Access Memory）、フラッシュメモリ、ＮＶＳＲＡＭ（Non Volatile Static Random Access Memory）などのデータを書き換え可能な半導体メモリであってもよい。

【００２６】

記憶部２２は、制御部２３で実行されるＯＳ（Operating System）や受信される要求を処理する各種プログラムを記憶する。さらに、記憶部２２は、制御部２３で実行されるプログラムで用いられる各種データを記憶する。例えば、記憶部２２は、ＩＤ管理表２２１、ＡＰＩキー管理表２２２、ＡＰＩ一覧表２２３、リクエスト履歴管理表２２４、状態遷移表２２５及びアクセス回数管理表２２６を有する。

【００２７】

ＩＤ管理表２２１は、ユーザと、ユーザＩＤとを対応付けた情報を保持する表である。図３は、ＩＤ管理表２２１のデータ構造の一例を示す図である。図３に示すように、ＩＤ管理表２２１は、ユーザの識別情報と、ユーザＩＤ（「user_id」）とを対応付ける。例えば、「ユーザＡ」には、ユーザＩＤ「abc」が対応している。ＡＰＩゲートウェイ２０は、「ユーザＡ」をユーザＩＤ「abc」として識別する。

【００２８】

ＡＰＩキー管理表２２２は、アプリケーションと、ＡＰＩキーとを対応付けた情報を保持する表である。図４は、ＡＰＩキー管理表２２２のデータ構造の一例を示す図である。図４に示すように、ＡＰＩキー管理表２２２は、アプリケーションの識別情報と、ＡＰＩキーとを対応付ける。図４の例では、「第１アプリケーション」には、ＡＰＩキー「１２３」が割り当てられている。

【００２９】

ＡＰＩ一覧表２２３は、ＡＰＩゲートウェイ２０がアプリケーションに対して提供するアプリサーバ１０のＡＰＩの一覧を示す表である。図５は、ＡＰＩ一覧表２２３のデータ構造の一例を示す図である。図５に示すように、ＡＰＩ一覧表２２３は、ＡＰＩゲートウェイ２０における、ＵＲＩが指定する情報に対する、「GET」，「POST」，「PUT」，「DELETE」のＨＴＴＰメソッドの実行の可否を対応付ける。ＡＰＩゲートウェイ２０は、このＡＰＩ一覧表２２３に従って、ＡＰＩリクエストを受信したアプリケーションに対し、ＡＰＩの提供を管理する。

【００３０】

例えば、ＡＰＩゲートウェイ２０は、ＵＲＩが「/attributes/ {user_id} （ユーザの

属性）」を指定した「GET」メソッドのＡＰＩリクエストを受信した場合には、このＡＰＩリクエストを送信したアプリケーションに対し、ユーザの属性を取得することを許可する。なお、本実施例では、ＡＰＩゲートウェイ２０が提供するＡＰＩの初期状態は、「GET/attributes/ {user_id} 」とする。

【００３１】

さらに、ＡＰＩゲートウェイ２０が、ＵＲＩが「/attributes/ {user_id} 」を指定したＡＰＩリクエストを受信した場合について説明する。この場合、ＡＰＩゲートウェイ２０は、ＡＰＩリクエストが「POST」メソッドの場合は、アプリケーションに対してリソース作成実行を禁止し、「PUT」メソッドの場合はアプリケーションに対してユーザの属性を更新することを許可する。ただし、ＡＰＩゲートウェイ２０は、ユーザ本人のみ更新を可能とする。また、ＡＰＩゲートウェイ２０は、ＡＰＩリクエストが、「DELETE」メソッドの場合はリソースの削除を禁止する。

【００３２】

リクエスト履歴管理表２２４は、アプリケーションごとに、アプリケーションが送信したＡＰＩリクエストの内容を時系列に示す表である。図６は、リクエスト履歴管理表２２４のデータ構造の一例を示す図である。

【００３３】

例えば、図６に示すように、リクエスト履歴管理表２２４は、各アプリケーションのＡＰＩキーと、ユーザＩＤと、ＡＰＩのリクエストＵＲＩと、このＵＲＩに対するＨＴＴＰメソッドと、パラメータと、ＡＰＩリクエスト時刻と、を対応付ける。リクエスト履歴管理表２２４は、例えば図６の１行目に、ＡＰＩキー「123」であるアプリケーションから、「2015/06/87　09:00:00」にリクエストＵＲＩが「/attributes/ {abc} 」を指定した「GET」メソッドのＡＰＩリクエストを受信したことを記憶する。ＡＰＩリクエストを受信すると、状態遷移インスタンス作成部２３３（後述）が、このＡＰＩリクエストの内容をリクエスト履歴管理表２２４に記録する。

【００３４】

状態遷移表２２５は、アプリケーションごとに作成された状態遷移表であって、アプリケーションによる過去の各ＡＰＩリクエストに応じて遷移した各状態間の遷移確率を示す表である。

【００３５】

図７は、状態遷移表２２５のデータ構成の一例を示す図である。図７に示すように、状態遷移表２２５は、アプリケーションの識別情報、ＡＰＩキーの識別内容、遷移前の状態（「state1」）、ＡＰＩリクエストによる遷移後の状態（「state2」）、及び、「state1」から「state2」への遷移確率を対応づける。状態遷移表２２５は、例えば図７の１行目に、第１アプリケーション３０ａが「GET/attributes/ {user_id} 」状態から「GET　/info/ {id} 」状態に「0.6」の確率で遷移することを記憶する。状態遷移表２２５の更新は、安定度算出部２３４（後述）が、状態遷移インスタンス作成部２３３が作成した状態遷移インスタンスを基に実行する。

【００３６】

アクセス回数管理表２２６は、アプリケーションごとに、アプリケーションのアクセスと、レートリミットとを対応付けた情報を保持する表である。図８は、アクセス回数管理表２２６のデータ構造の一例を示す図である。図８に示すように、アクセス回数管理表２２６は、アプリケーションの識別情報と、アクセス回数と、レートリミットとを対応付ける。

【００３７】

例えば、図８に示すように、アクセス回数管理表２２６は、第１アプリケーション３０ａのアクセス回数が１００回であることを記憶する。そして、アクセス回数管理表２２６は、第１アプリケーション３０ａのレートリミットが３００回であることを記憶する。

【００３８】

レートリミットの初期値は、アクセス回数管理表２２６へのアプリケーションの新規登

録時に新規アプリ登録部２３２（後述）が付与する。その後は、レートリミット算出部２３５（後述）が、レートリミットの値を定期的に更新する。また、アクセス回数管理表２２６のアクセス回数は、リクエスト受付可否判定部２３６（後述）がインクリメント或いはリセットする。なお、単位時間は、１時間であり、レートリミットの初期値は、例えば、１時間あたり５０回である。そして、レートリミットの最小値は、例えば、１時間あたり１０回である。アクセス回数管理表２２６は、例えば１時間ごとに更新され、例えば毎時００分に更新される。

【００３９】

図２に戻り、制御部２３は、ＡＰＩゲートウェイ２０を制御するデバイスである。制御部２３としては、ＣＰＵ（Central Processing Unit）、ＭＰＵ（Micro Processing Unit）等の電子回路や、ＡＳＩＣ（Application Specific Integrated Circuit）、ＦＰＧＡ（Field Programmable Gate Array）等の集積回路を採用できる。制御部２３は、各種の処理手順を規定したプログラムや制御データを格納するための内部メモリを有し、これらによって種々の処理を実行する。制御部２３は、各種のプログラムが動作することにより各種の処理部として機能する。

【００４０】

制御部２３は、ＡＰＩゲートウェイ管理部２３１、新規アプリ登録部２３２、状態遷移インスタンス作成部２３３、安定度算出部２３４、レートリミット算出部２３５（変更部）及びリクエスト受付可否判定部２３６（判定部）を有する。

【００４１】

ＡＰＩゲートウェイ管理部２３１は、アプリケーションから受け付けた情報等をアプリサーバ１０に送信する。また、ＡＰＩゲートウェイ管理部２３１は、アプリサーバ１０による処理結果の情報をアプリケーションへ送信する。ＡＰＩゲートウェイ管理部２３１は、アプリケーションと、アプリサーバ１０との間の通信を管理する。ＡＰＩゲートウェイ管理部２３１は、新たなアプリケーションから利用申請があった場合には、このアプリケーションに対してＡＰＩキーを発行する。ＡＰＩゲートウェイ２０は、このＡＰＩキーを用いて、アプリケーションの認証を実行する。

【００４２】

新規アプリ登録部２３２は、アプリケーションより利用申請を受信した場合、ＡＰＩゲートウェイ管理部２３１に利用申請を送信し、発行されたＡＰＩキーをアプリケーションに返却する。新規アプリ登録部２３２は、アクセス回数管理表２２６に、発行されたＡＰＩキーを登録し、このＡＰＩキーのアプリケーションに対応するレートリミットを所定の初期値に設定する。

【００４３】

状態遷移インスタンス作成部２３３は、各アプリケーションからのＡＰＩリクエストを受信した場合、ＡＰＩリクエストの種類、パラメータ、時刻情報等をリクエスト履歴管理表２２４に記録する。状態遷移インスタンス作成部２３３は、ＡＰＩキー、ユーザＩＤ、ＡＰＩのリクエストＵＲＩ、このＵＲＩに対するＨＴＴＰメソッド、パラメータ及びＡＰＩリクエスト時刻をリクエスト履歴管理表２２４に記録する。

【００４４】

また、状態遷移インスタンス作成部２３３は、ＡＰＩリクエストを送信したアプリケーションの挙動を求めるために、ＡＰＩリクエストを送信したアプリケーションについて状態遷移インスタンスを作成する。この状態遷移インスタンスとは、あるアプリケーションのあるユーザについて、ログインリクエストからログアウトリクエスト、或いは、タイムアウトまでの間に、ＡＰＩリクエストに応じて遷移した各状態間の遷移確率を示すものである。ここで、ログインリクエストは、あるアプリケーションのあるユーザによる最初のＡＰＩリクエストである。ログアウトリクエストは、ＡＰＩサーバが事前に指定した種類のＡＰＩリクエストである。また、あるユーザのリクエストが最後に到着してから事前に定めた時間が経過した場合が、タイムアウトとなる。

【００４５】

安定度算出部２３４は、ＡＰＩリクエストを受信した場合に該ＡＰＩリクエストを送信したアプリケーションについて、アプリケーションによるＡＰＩリクエストの内容の遷移がリクエスト履歴管理表２２４に対してどの程度安定しているかを示す安定度を算出する。具体的には、安定度算出部２３４は、ＡＰＩリクエストを送信したアプリケーションについて、状態遷移インスタンス作成部２３３が作成した状態遷移インスタンスと、記憶部２２の状態遷移表２２５とを比較して安定度を算出する。例えば、安定度算出部２３４は、ＡＰＩリクエストを送信したアプリケーションについて、状態遷移インスタンス作成部２３３が作成した状態遷移インスタンスと状態遷移表２２５との一致度に応じて安定度を増減する。

【００４６】

レートリミット算出部２３５は、ＡＰＩリクエストを送信したアプリケーションについて、安定度算出部２３４が算出した安定度に応じて該アプリケーションのレートリミットを変更する。例えば、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度の増減に応じてレートリミットを増減する。具体的には、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度の値が増えた場合にはレートリミットの値を増やす。一方、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度の値が減った場合にはレートリミットの値を減らす。或いは、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度の値が所定の閾値を超えた場合にはレートリミットの値を増やす。一方、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度の値が所定の閾値以下である場合にはレートリミットの値を減らす。レートリミット算出部２３５は、所定時間ごとに、記憶部２２のアクセス回数管理表２２６のレートリミットを変更した値に更新する。

【００４７】

リクエスト受付可否判定部２３６は、アプリケーションによるＡＰＩリクエストがあった場合に、このアプリケーションに対してＡＰＩリクエストの受付を行うか否かを判定する。具体的には、リクエスト受付可否判定部２３６は、このアプリケーションについて、アクセス回数管理表２２６におけるアクセス回数とアクセス回数管理表２２６におけるレートリミットとを比較し、ＡＰＩリクエストの受付を行うか否かを判定する。なお、リクエスト受付可否判定部２３６は、所定の単位時間ごとに、アクセス回数管理表２２６における各アプリケーションのアクセス回数をリセットする。

【００４８】

例えば、リクエスト受付可否判定部２３６は、ＡＰＩリクエストを送信したアプリケーションのアクセス回数が、このアプリケーションのレートリミットに達する場合には、このアプリケーションに対しＡＰＩリクエストの受付を拒否するメッセージを送信する。一方、リクエスト受付可否判定部２３６は、ＡＰＩリクエストを送信したアプリケーションのアクセス回数が、このアプリケーションのレートリミット未満の場合には、このアプリケーションに対しＡＰＩリクエストの受付を受理するメッセージを送信する。

【００４９】

具体的には、アクセス回数管理表２２６が図８に示す内容である場合、第１アプリケーション３０ａは、アクセス回数が１００回でありレートリミットが３００回であるため、所定の単位時間の残り時間では、２００回のＡＰＩリクエストを送信することができる。このため、リクエスト受付可否判定部２３６は、第１アプリケーション３０ａによるＡＰＩリクエストを受信した場合には、この第１アプリケーション３０ａに対しＡＰＩリクエストの受付を受理するメッセージを送信する。

【００５０】

一方、第２アプリケーション３０ｂは、アクセス回数が２００回であり、２００回であるレートリミットに達している。このため、リクエスト受付可否判定部２３６は、第２アプリケーション３０ｂによるＡＰＩリクエストを受信した場合、この第２アプリケーション３０ｂに対しＡＰＩリクエストの受付を拒否するメッセージを送信する。

【００５１】

このように、ＡＰＩゲートウェイ２０は、状態遷移インスタンス作成部２３３が作成した状態遷移インスタンスを用いて、ＡＰＩリクエストを送信したアプリケーションの実際の挙動を求めている。そこで、まず、あるアプリケーションについての各状態遷移の挙動の変化の一例を説明する。

【００５２】

［アプリケーションの状態遷移］

まず、図９及び図１０を参照して、アプリケーションによるＡＰＩリクエストに応じて遷移した各状態及び各状態間の遷移確率について説明する。図９は、第１アプリケーションについての各状態遷移の一例を説明するための図である。図１０は、第１アプリケーションについての各状態遷移の挙動の変化の一例を説明するための図である。

【００５３】

例えば、本実施例では、ＡＰＩゲートウェイ２０が提供するＡＰＩの初期状態（最初のＡＰＩ呼び出し：initial）は、「GET/attributes/ {user_id} 」としており、任意の状態から、ログアウト或いはタイムアウトが発生し得る。

【００５４】

そして、図９に示すように、第１アプリケーション３０ａでは、これまでの各ＡＰＩリクエストに応じて、この初期状態、「GET/info/ {id} 」状態及び「POST/info」状態のそれぞれの間を遷移している。なお、「state1」と「state2」が同じ状態である場合もある。また、「「state1」から「state2」への遷移確率」は、「「state1」から「state2」への遷移回数」を、「「state1」からの全ての遷移回数」で除することによって計算することができる。なお、任意の状態（「state1」）について、全ての次の状態（「state2」）への遷移確率の総和は１になる。

【００５５】

これまでの第１アプリケーション３０ａによるＡＰＩリクエストによる「GET/attributes/ {user_id} 」状態から「GET/info/ {id} 」状態への遷移確率は「0.6」であり、その逆の遷移は「0.7」であった。これに対し、「GET/attributes/ {user_id} 」状態から「POST/info」状態への遷移確率は「0.1」であった。このため、第１アプリケーション３０ａは、「GET/attributes/ {user_id} 」状態と「POST/info」状態との間の遷移と比較して、「GET/attributes/ {user_id} 」状態と「GET/info/ {id} 」状態との間の遷移を多く行うような挙動を行っていた。

【００５６】

これに対し、第１アプリケーション３０ａに対して、改版やＡＰＩキーの乗っ取りがあった場合には、これまでと比べて挙動が変わる場合が多い。例えば、図１０に示すように、初期状態である「GET/attributes/ {user_id} 」状態から「GET/info/ {id} 」状態への遷移確率が「0.4」まで減っている（図１０のＡ１参照）。そして、これまでなかった「GET/attributes/ {other_user_id} 」状態が生じる（図１０のＡ２参照）。さらに、この「GET/attributes/ {other_user_id} 」状態から「GET/attributes/ {other_user_id} 」状態の遷移確率は「0.9」と高く、「GET/attributes/ {other_user_id} 」状態と「GET/attributes/ {user_id} 」状態との間の遷移確率は「0.1」に変化する。

【００５７】

このように、アプリケーションに対して改版やＡＰＩキーの乗っ取りがあった場合には、ＡＰＩリクエストに応じて遷移する状態や状態間の遷移確率、つまり、アプリケーションの挙動に変化が生じる。そこで、アプリケーションの挙動の変化の有無を求めるために、状態遷移インスタンス作成部２３３が、ＡＰＩリクエストを送信したアプリケーションの挙動を示す状態遷移インスタンスを作成する。次に、状態遷移インスタンス作成部２３３の処理について説明する。

【００５８】

［状態遷移インスタンス作成部の処理］

まず、状態遷移インスタンス作成部２３３は、ＡＰＩリクエストを受信すると、ＡＰＩリクエストの内容、ＡＰＩリクエストを送信したアプリケーションのＡＰＩキー、ユーザ

ＩＤ及び受信時刻を対応付けて、リクエスト履歴管理表２２４に順次記録する。
【００５９】
そして、状態遷移インスタンス作成部２３３は、所定のタイミングで、これまでにリクエスト履歴管理表２２４に蓄積された該ＡＰＩキーに対応する全てのＡＰＩリクエスト列を読み出す。具体的には、状態遷移インスタンス作成部２３３は、あるＡＰＩキーのアプリケーションについて、ログアウトリクエストを受信、或いは、あるユーザのタイムアウトが生じると、ＡＰＩリクエスト列の読み出しを行う。また、状態遷移インスタンス作成部２３３は、ログインリクエストからログアウトリクエスト、或いは、タイムアウトまでの間までの、ＡＰＩキーに対応するＡＰＩリクエスト列を読み出す。
【００６０】
そして、状態遷移インスタンス作成部２３３は、読み出した全ＡＰＩリクエスト列を時系列順にソートした後、このＡＰＩキーに対応するアプリケーションの、「state1」と「state2」との組み合わせ、及び、「state1」から「state2」への遷移確率を求める。
【００６１】
ここで、ＡＰＩのそれぞれに引数について、値が異なる場合に、異なる状態とみなすか、或いは、同じ状態とみなすかは、アプリサーバ１０等の制御によって予め決まっている。すなわち、属性ごとに状態の扱いは予め決まっている。
【００６２】
例えば、「/data/ {id} 」は、「id」の値に関係なく全て同じ状態とする。そして、「/attributes/ {user_id} 」は、「user_id」がユーザ自身か、或いは、他のユーザであるかの２状態で区別する。この決まりに従って、状態遷移インスタンス作成部２３３は、このＡＰＩキーに対応するアプリケーションの各状態を判別する。なお、前述したように、「「state1」から「state2」への遷移確率」は、「「state1」から「state2」への遷移回数」を、「「state1」からの全ての遷移回数」で除することによって計算することができる。
【００６３】
図１１は、状態遷移インスタンスの一例を示す図である。状態遷移インスタンス作成部２３３は、図１０に示す挙動に変化した第１アプリケーション３０ａについては、図１１に示す状態遷移インスタンス２２５ａを作成する。
【００６４】
図１１に示すように、状態遷移インスタンス２２５ａは、「GET/attributes/ {user_id} 」状態から「GET/info/ {id} 」状態への遷移確率が「0.4」であることを示す（図１１のＡ１１参照）。そして、状態遷移インスタンス２２５ａは、図１１のＡ１２に示すように、状態遷移表２２５が示す状態とは異なる新しい状態を含む。この新しい状態とは、「GET /attributes/ {other_user_id} 」状態である。そして、状態遷移インスタンス２２５ａは、この新しい状態遷移として、「GET/attributes/ {user_id} 」状態との間の遷移、「GET /attributes/ {other_user_id} 」状態と「GET /attributes/ {other_user_id} 」状態との間の遷移を含む。
【００６５】
このように、状態遷移インスタンス作成部２３３が、ＡＰＩリクエストを行ったアプリケーションについての状態遷移インスタンスを作成することによって、このアプリケーションの現在の挙動と、状態遷移表２２５が示す過去の挙動とが比較可能となる。
【００６６】
この状態遷移インスタンス作成部２３３が作成した状態遷移インスタンス２２５ａを用いて、安定度算出部２３４は、例えば第１アプリケーション３０ａの安定度を算出する。そこで、次に、安定度算出部２３４による安定度算出処理について説明する。
【００６７】
［安定度算出部の処理］
安定度算出部２３４は、状態遷移インスタンス作成部２３３が作成した状態遷移インスタンスと、記憶部２２が記憶する状態遷移表２２５との一致度に応じて安定度を増減する

。この安定度算出部２３４は、安定度を算出することによって、ＡＰＩリクエストの挙動が過去の挙動と比して変化したアプリケーションを判別している。

【００６８】

具体的には、安定度算出部２３４は、状態遷移インスタンスが状態遷移表２２５に示されていない状態を含む場合には、状態遷移インスタンスに対応するアプリケーションが、ＡＰＩリクエストの挙動が過去の挙動と比して大きく変化したものであると判断する。したがって、この場合には、安定度算出部２３４は、安定度を最も少ない値（例えば「０」）と算出する。

【００６９】

そして、安定度算出部２３４は、状態遷移インスタンスが状態遷移表２２５に示されていない状態遷移を含む場合には、状態遷移インスタンスに対応するアプリケーションが、ＡＰＩリクエストの挙動が過去の挙動と比して変化したものであると判断する。したがって、この場合には、安定度をより少ない値（例えば「１」）と算出する。

【００７０】

そして、安定度算出部２３４は、状態遷移インスタンスの遷移確率と状態遷移表２２５の遷移確率との一致度が所定値よりも低い場合、状態遷移インスタンスに対応するアプリケーションが、過去の挙動からの変化が小さいものであると判断する。したがって、この場合には、安定度算出部２３４は、安定度をより多い値（例えば「２」）と算出する。なお、安定度算出部２３４は、例えば、文字列の編集距離を計算するアルゴリズムを用いて、状態遷移インスタンスと状態遷移表２２５との一致度を算出する。

【００７１】

そして、安定度算出部２３４は、状態遷移インスタンスの遷移確率と状態遷移表２２５の遷移確率との一致度が所定値以上である場合、状態遷移インスタンスに対応するアプリケーションが、過去の挙動からの変化が最も小さいものであると判断する。したがって、この場合には、安定度算出部２３４は、安定度を最も多い値（例えば「３」）と算出する。

【００７２】

例えば、第１アプリケーション３０ａについて、図１１に示す状態遷移インスタンス２２５ａと、図７に示す状態遷移表２２５との一致状態を判断する場合について説明する。この場合、図１１に示すように、「GET/attributes/ {user_id} 」状態から「GET /info/ {id} 」状態への遷移確率については、状態遷移表２２５が「0.6」であるのに対し、状態遷移インスタンス２２５ａでは「0.4」に低下している（図１１のＡ１１参照）。さらに、状態遷移インスタンス２２５ａは、状態遷移表２２５と比較して、「GET/attributes/ {other_user_id} 」状態を新たに含む（図１１のＡ１２参照）。

【００７３】

したがって、安定度算出部２３４は、状態遷移インスタンス２２５ａが状態遷移表２２５に示されていない状態を含むと判断し、この状態遷移インスタンス２２５ａに対する安定度を「０」と算出する（図１１の矢印Ａ１３参照）。

【００７４】

このように、安定度算出部２３４は、安定度を算出することによって、状態遷移インスタンスに対応するアプリケーションの挙動が過去の挙動と比して、どの程度変化したかを数値化している。この安定度によって、ＡＰＩリクエストの挙動が過去の挙動と比して変化したアプリケーションを判別することができる。

【００７５】

例えば、安定度が低いアプリケーションほど、ＡＰＩリクエストの挙動が過去の挙動に対して大きく変化したアプリケーションであると言える。言い換えると、このアプリケーションは、改版アプリケーションや悪意アプリケーションに変化したアプリケーションであるおそれが高い。また、安定度が高いアプリケーションほど、ＡＰＩリクエストの挙動の、過去の挙動に対する変化が小さいアプリケーションであると言える。言い換えると、このアプリケーションは、改版アプリケーションや悪意アプリケーションに変化したアプ

リケーションであるおそれが低い。

【００７６】

安定度算出部２３４は、算出した安定度をレートリミット算出部２３５に出力する。このため、次に、レートリミット算出部２３５によるレートリミット算出処理について説明する。なお、安定度算出部２３４は、安定度を算出した後に、この第１アプリケーション３０ａについて、状態遷移インスタンス２２５ａが示す状態遷移の内容と各状態間の遷移確率とを用いて、状態遷移表２２５を更新する。

【００７７】

［レートリミット算出部の処理］

レートリミット算出部２３５は、安定度算出部２３４が出力したアプリケーションの安定度の増減に応じて、このアプリケーションのレートリミットを増減する。そして、レートリミット算出部２３５は、アクセス回数管理表２２６のレートリミットの値を、変更した値に更新する。

【００７８】

具体的には、レートリミット算出部２３５は、安定度が最も少ない場合、例えば、安定度が「０」である場合、このアプリケーションのレートリミットを予め設定した最小値（例えば「１０」）に変更する。そして、レートリミット算出部２３５は、安定度がより少ない場合、例えば、安定度が「１」である場合、このアプリケーションのレートリミットを前回設定値から所定値（例えば「５」）を減算した値に変更する。そして、レートリミット算出部２３５は、安定度がより多い場合、例えば、安定度が「２」である場合、このアプリケーションのレートリミットを前回設定値のまま維持する。また、レートリミット算出部２３５は、安定度が最も高い場合、例えば安定度が「３」である場合、レートリミットを前回設定値に所定値（例えば「５」）を加算した値に変更する。なお、レートリミットについては、上限値或いは下限値を予め設定してもよい。

【００７９】

図１２は、アクセス回数管理表２２６の更新を説明する図である。安定度算出部２３４によって安定度が「０」と算出された第１アプリケーション３０ａについて、レートリミット算出部２３５は、レートリミットが「３００」であるアクセス回数管理表２２６ａ（図１２の（ａ）参照）から、レートリミットを「１０」に変更したアクセス回数管理表２２６ａ（図１２の(ｂ)参照）に更新する。なお、単位時間が経過していた場合には、全アプリケーションについてのアクセス回数は、リクエスト受付可否判定部２３６がリセットする。

【００８０】

この結果、ＡＰＩゲートウェイ２０は、安定度が「０」である第１アプリケーション３０ａに対してアクセス回数を制限することができる。そして、ＡＰＩゲートウェイ２０は、安定度が「１」である場合には、レートリミットを前回回数から減らすことで、アクセス回数を下げることができる。また、ＡＰＩゲートウェイ２０は、安定度が「２」である場合には、レートリミットを変更せず、安定度が「３」である場合には、レートリミットを増やしている。

【００８１】

このように、ＡＰＩゲートウェイ２０は、安定度の低いアプリケーションについては、改版アプリケーションや悪意アプリケーションに変化したアプリケーションであるおそれが高いため、単位時間あたりのレートリミットを下げている。これによって、ＡＰＩゲートウェイ２０は、改版アプリケーションや悪意アプリケーションに変化したアプリケーションからＡＰＩのデータ機能を保護することができる。

【００８２】

一方、ＡＰＩゲートウェイ２０は、安定度の高いアプリケーションについては、改版アプリケーションや悪意アプリケーションに変化したアプリケーションであるおそれが低いため、単位時間あたりのレートリミットを維持或いは上げている。これによって、ＡＰＩゲートウェイ２０は、改版アプリケーションや悪意アプリケーションに変化したアプリケ

ーションであるおそれが低いアプリケーションについては、ＡＰＩ提供の利便性を高めている。そこで、次に、ＡＰＩゲートウェイ２０におけるＡＰＩ管理処理の全体の流れを説明する。

【００８３】

［ＡＰＩ管理処理の流れ］

図１３は、実施例に係るＡＰＩ管理処理の一例を説明する図である。まず、ＡＰＩゲートウェイ２０では、新規アプリ登録部２３２が、アプリケーション３０より利用申請（図１３の（１）参照）を受信すると、ＡＰＩゲートウェイ管理部２３１に利用申請を送信する（図１３の（２）参照）。

【００８４】

そして、新規アプリ登録部２３２が、発行されたＡＰＩキー（図１３の（３）参照）をこのＡＰＩキーをアクセス回数管理表２２６に登録する（図１３の（４）参照）。この際、新規アプリ登録部２３２は、このＡＰＩキーに対応するアプリケーション（例えば第２アプリケーション）について、レートリミットを初期値（例えば「５０」）に設定する（図１３のセルＣ１の左表Ｔａ参照）。続いて、新規アプリ登録部２３２が、アプリケーション３０にＡＰＩキーを返却する（図１３の（５）参照）。

【００８５】

続いて、アプリケーション３０によるＡＰＩリクエストを受信した場合（図１３の（６）参照）、リクエスト受付可否判定部２３６は、アクセス回数管理表２２６から、このアプリケーション３０のアクセス回数とレートリミットとを読み出す（図１３の（７）参照）。

【００８６】

アプリケーション３０のアクセス回数がレートリミットに達していた場合、リクエスト受付可否判定部２３６は、ＡＰＩリクエストを拒否し（図１３の（８）参照）、エラーレスポンス（図１３の（９）参照）をアプリケーション３０に送信する。一方、アプリケーション３０のアクセス回数がレートリミット未満である場合、リクエスト受付可否判定部２３６は、ＡＰＩリクエストを受理し（図１３の（１０）参照）、アクセス回数管理表２２６のアクセス回数を加算する。そして、リクエスト受付可否判定部２３６は、ＡＰＩハンドラ１１によるレスポンスを（図１３の（１１）参照）をアプリケーション３０に送信する。

【００８７】

また、状態遷移インスタンス作成部２３３は、受信したＡＰＩリクエストの内容を時系列に、リクエスト履歴管理表２２４に記録する。そして、状態遷移インスタンス作成部２３３は、一定のタイミングで、状態遷移インスタンスを作成し、安定度算出部２３４に出力する（図１３の（１２）参照）。状態遷移インスタンスは、このアプリケーション３０について、ログインリクエストからログアウトリクエスト或いはタイムアウトの間にＡＰＩリクエストに応じて遷移した各状態間の遷移確率を示すものである。状態遷移インスタンス作成部２３３は、リクエスト履歴管理表２２４から、このアプリケーション３０のリクエスト履歴を読み出して、状態遷移インスタンスを作成する。

【００８８】

アプリ安定度算出部２３４は、状態遷移表２２５から、ＡＰＩリクエストを送信したアプリケーションについての過去の状態遷移を読み出す（図１３の（１３）参照）。前述したように、状態遷移表２２５は、アプリケーションごとに、アプリケーションによる過去の各ＡＰＩリクエストに応じて遷移した各状態間と、各状態間の遷移確率とを示す（図１３のセルＣ２参照）。

【００８９】

安定度算出部２３４は、この読み出した過去の状態遷移と、状態遷移インスタンス作成部２３３が作成した状態遷移インスタンスとを比較して安定度を算出する。安定度算出部２３４は、この算出した安定度をレートリミット算出部２３５に出力する（図１３の（１４）参照）。また、安定度算出部２３４は、状態遷移インスタンス作成部２３３が作成し

た状態遷移インスタンスを基に状態遷移表２２５の更新を実行する（図１３の（１５）参照）。

【００９０】

続いて、レートリミット算出部２３５は、安定度が算出されたアプリケーションについて、入力された安定度に応じてレートリミットを変更し、アクセス回数管理表２２６のレートリミットも、変更した値に更新する（図１３の（１６）参照）。例えば、第１アプリケーション３０ａの安定度が「０」である場合には、第１アプリケーション３０ａのレートリミットを最小値「１０」に変更する（図１３のセルＣ１の右表Ｔｂ参照）。この結果、ＡＰＩゲートウェイ２０では、安定度が「０」である第１アプリケーション３０ａのアクセス回数を最小値まで制限することができる。

【００９１】

このように、ＡＰＩゲートウェイ２０は、ＡＰＩリクエストを送信したアプリケーションについて状態遷移インスタンスを作成し、状態遷移インスタンスが状態遷移表２２５に対してどの程度安定しているかを示す安定度を算出する。この安定度を用いて、ＡＰＩゲートウェイ２０は、ＡＰＩリクエストの挙動が過去の挙動と比して変化したアプリケーションを判別する。そして、ＡＰＩゲートウェイ２０は、ＡＰＩリクエストの挙動が過去の挙動と比して変化したアプリケーションについては、単位時間あたりのレートリミットを下げる。これによって、ＡＰＩゲートウェイ２０は、改版アプリケーションや悪意アプリケーションに変化したアプリケーションからＡＰＩのデータや機能を保護している。

【００９２】

［新規アプリ登録処理の手順］

次に、ＡＰＩゲートウェイ２０の各機能構成部が実行する処理の手順について詳細に説明する。まず、新規アプリ登録部２３２による新規アプリ登録処理の手順について説明する。図１４は、本実施例に係る新規アプリ登録処理の手順の一例を示すフローチャートである。

【００９３】

図１４に示すように、新規アプリ登録部２３２が、アプリケーションより利用申請を受信すると（ステップＳ１）、ＡＰＩゲートウェイ管理部２３１に利用申請を送信し、発行されたＡＰＩキーを受信する（ステップＳ２）。新規アプリ登録部２３２は、このＡＰＩキーに対応するアプリケーションをアクセス回数管理表２２６に登録し、レートリミットを初期値（例えば「５０」）に設定する（ステップＳ３）。続いて、新規アプリ登録部２３２が、アプリケーションにＡＰＩキーを返却して（ステップＳ４）、新規アプリ登録処理を終了する。

【００９４】

［リクエスト受付可否判定処理の手順］

次に、リクエスト受付可否を判定する処理の手順について説明する。図１５は、本実施例に係るリクエスト受付可否判定処理の手順の一例を示すフローチャートである。

【００９５】

図１５に示すように、リクエスト受付可否判定部２３６は、アプリケーションによるＡＰＩリクエストを受信した場合、到着したＡＰＩリクエストからＡＰＩキーを参照する（ステップＳ１１）。リクエスト受付可否判定部２３６は、アクセス回数管理表２２６から、該ＡＰＩキーのアクセス回数と、レートリミットとを参照する（ステップＳ１２）。

【００９６】

リクエスト受付可否判定部２３６は、アクセス回数管理表２２６から参照したデータを基に、単位時間でのアクセス回数が該ＡＰＩキーのレートリミットに達しているか否かを判定する（ステップＳ１３）。

【００９７】

リクエスト受付可否判定部２３６は、単位時間でのアクセス回数がレートリミットに達していると判定した場合（ステップＳ１３：Ｙｅｓ）、ＡＰＩリクエストを拒否するエラーレスポンスを作成する（ステップＳ１４）。

【００９８】
これに対し、リクエスト受付可否判定部２３６は、単位時間でのアクセス回数がレートリミットに達していないと判定した場合（ステップＳ１３：Ｎｏ）、アクセス回数管理表２２６における該ＡＰＩキーのアクセス回数をインクリメントする（ステップＳ１５）。続いて、リクエスト受付可否判定部２３６は、ＡＰＩハンドラ１１によるリクエスト処理を行う（ステップＳ１６）。
【００９９】
ステップＳ１４或いはステップＳ１６終了後、リクエスト受付可否判定部２３６は、レスポンスを、アプリケーションに返却する（ステップＳ１７）。続いて、レートリミット算出部２３５は、所定の単位時間が経過していた場合、アクセス回数管理表２２６のアクセス回数をリセットして（ステップＳ１８）、リクエスト受付可否を判定する処理を終了する。
【０１００】
［状態遷移インスタンス作成処理の手順］
次に、状態遷移インスタンスを作成してレートリミットを変更するまで処理の手順について説明する。まず、状態遷移インスタンスを作成する処理の手順について説明する。図１６は、本実施例に係る状態遷移インスタンス作成処理の手順の一例を示すフローチャートである。
【０１０１】
図１６に示すように、状態遷移インスタンス作成部２３３は、あるＡＰＩキーのアプリケーションについて、ＡＰＩリクエストが到着するまで、または、あるユーザのタイムアウトが発生するまで待機する（ステップＳ２１）。状態遷移インスタンス作成部２３３は、ログアウトリクエストの到着、または、タイムアウトの発生があるかを判断する（ステップＳ２２）。状態遷移インスタンス作成部２３３は、ログアウトリクエストの到着及びタイムアウトの発生のいずれもないと判断した場合（ステップＳ２２：Ｎｏ）、リクエスト履歴管理表２２４に、今回のＡＰＩリクエストの内容を格納する（ステップＳ２３）。その後、ステップＳ２１に戻り、待機を継続する。
【０１０２】
一方、状態遷移インスタンス作成部２３３は、ログアウトリクエストの到着またはタイムアウトの発生があると判断した場合（ステップＳ２２：Ｙｅｓ）、「user:=ログアウトリクエストのユーザまたはタイムアウトが発生したユーザ」とする（ステップＳ２４）。続いて、状態遷移インスタンス作成部２３３は、このＡＰＩキーに対応するアプリケーションについて状態遷移インスタンス（「instance」）を作成する（ステップＳ２５）。状態遷移インスタンス作成部２３３は、「instance.key:=APIキー」とする。そして、状態遷移インスタンス作成部２３３は、「state1:=null」、「transition:= {} 」、「total:= {} 」とする（ステップＳ２６）。続いて、状態遷移インスタンス作成部２３３は、ステップＳ２７に進み、リクエスト履歴管理表２２４から、このＡＰＩキーに対応するアプリケーションの履歴の全てを順次取り出す処理を行う。
【０１０３】
まず、状態遷移インスタンス作成部２３３は、「state2:=リクエスト履歴管理表の最も古いuserのリクエスト」とし、「state2」をリクエスト履歴管理表２２４から削除する（ステップＳ２７）。続いて、状態遷移インスタンス作成部２３３は、「state1!=null」であるか否かを判断する（ステップＳ２８）。
【０１０４】
状態遷移インスタンス作成部２３３は、「state1!=null」であると判断した場合（ステップＳ２８：Ｙｅｓ）、状態遷移カウンタtransitionについて、「transition[state1,state2]:=transition[state1,state2]+1」とする（ステップＳ２９）。状態遷移インスタンス作成部２３３は、「total[state1]:=total[state1]+1」として（ステップＳ３０）、ステップＳ３２に進む。
【０１０５】

一方、状態遷移インスタンス作成部２３３は、「state1!=null」でないと判断した場合（ステップＳ２８：Ｎｏ）、「instance.initial:=state2」とし（ステップＳ３１）、ステップＳ３２に進む。

【０１０６】

続いて、状態遷移インスタンス作成部２３３は、「instance.states」に「state2」を追加し、「state1:=state2」とする（ステップＳ３２）。状態遷移インスタンス作成部２３３は、リクエスト履歴管理表２２４に「user」のリクエストが存在しないか否かを判断する（ステップＳ３３）。

【０１０７】

状態遷移インスタンス作成部２３３は、リクエスト履歴管理表２２４に「user」のリクエストが存在すると判断した場合（ステップＳ３３：Ｎｏ）、ステップＳ２７に戻る。この場合、状態遷移インスタンス作成部２３３は、このＡＰＩキーに対応するアプリケーションの履歴が残っていると判断できるため、ステップＳ２７に戻り、このＡＰＩキーに対応するアプリケーションの履歴の取り出しを継続する。

【０１０８】

一方、状態遷移インスタンス作成部２３３が、リクエスト履歴管理表２２４に「user」のリクエストが存在しないと判断した場合（ステップＳ３３：Ｙｅｓ）について説明する。この場合には、リクエスト履歴管理表２２４から、このＡＰＩキーに対応するアプリケーションの履歴を全件取り出したと判断できる。このため、状態遷移インスタンス作成部２３３は、全ての「state1, state2」について、「instance.transition[state1,state2]:=transition[state1,state2]/total[state1]」を実行する（ステップＳ３４）。言い換えると、状態遷移インスタンス作成部２３３は、各「state1, state2」の状態遷移の組み合わせに対し、遷移確率をそれぞれ算出する。そして、状態遷移インスタンス作成部２３３は、算出した全ての「state1, state2」に対する「instance.transition[state1,state2]」を安定度算出部２３４に出力する。状態遷移インスタンス作成部２３３は、具体的には、図１１に示すデータ構造を有する状態遷移インスタンス２２５ａを安定度算出部２３４に出力する。

【０１０９】

続いて、安定度算出部２３４は、この状態遷移インスタンス作成部２３３が作成した「instance」についての安定度を算出する安定度算出処理を行う（ステップＳ３５）。ステップＳ３５終了後、状態遷移インスタンス作成部２３３は、ステップＳ２１に戻り、待機を行う。

【０１１０】

［安定度算出処理の手順］

次に、図１６に示す安定度算出処理の手順について説明する。図１７は、図１６に示す安定度算出処理の手順の一例を示すフローチャートである。

【０１１１】

図１７に示すように、まず、安定度算出部２３４は、状態遷移インスタンス作成部２３３から「instance」を受信すると、この「instance」と、状態遷移表２２５の「instance.key」についての状態遷移とを比較する（ステップＳ４１）。言い換えると、安定度算出部２３４は、「instance」に示された遷移状態及び状態間の遷移確率と、状態遷移表２２５のうち「instance」が作成されたアプリケーションに関する遷移状態及び状態間の遷移確率とを比較する。

【０１１２】

続いて、安定度算出部２３４は、「instance」が新しい状態を含むか否かを判断する（ステップＳ４２）。言い換えると、安定度算出部２３４は、このアプリケーションにおいて、今までなかった型のリクエストが存在するか否かを判断する。

【０１１３】

安定度算出部２３４は、「instance」が新しい状態を含むと判断した場合（ステップＳ４２：Ｙｅｓ）、該ＡＰＩキーに対応するアプリケーションの安定度が０であると算出し

（ステップＳ４３）、算出した値をレートリミット算出部２３５に出力する。この場合、該ＡＰＩキーに対応するアプリケーションが、これまでとは全く異なる状態に遷移するという挙動を行ったことが判別できるためである。

【０１１４】

一方、安定度算出部２３４は、「instance」が新しい状態を含まないと判断した場合（ステップＳ４２：Ｎｏ）、「instance」が新しい状態遷移を含むか否かを判断する（ステップＳ４４）。言い換えると、安定度算出部２３４は、このアプリケーションにおいて、今までなかったリクエストの受信順序が存在するか否かを判断する。

【０１１５】

安定度算出部２３４は、「instance」が新しい状態遷移を含むと判断した場合（ステップＳ４４：Ｙｅｓ）、該ＡＰＩキーに対応するアプリケーションの安定度が１であると算出し（ステップＳ４５）、算出した値をレートリミット算出部２３５に出力する。この場合、該ＡＰＩキーに対応するアプリケーションが、これまでと同じ状態間を遷移するものの、異なる順序で遷移するという挙動を行ったことが判別できるためである。

【０１１６】

一方、安定度算出部２３４は、「instance」が新しい状態遷移を含まないと判断した場合（ステップＳ４４：Ｎｏ）、「instance.transition」の遷移確率が状態遷移表２２５の遷移確率と所定の割合で一致するか否かを判断する（ステップＳ４６）。この所定の割合は、予め設定された値であり、例えば「９０％」である。もちろん、この所定の割合は、変更が可能である。

【０１１７】

安定度算出部２３４は、「instance.transition」の遷移確率が状態遷移表２２５の遷移確率と所定の割合で一致しないと判断した場合（ステップＳ４６：Ｎｏ）、該ＡＰＩキーに対応するアプリケーションの安定度が２であると算出する（ステップＳ４７）。そして、安定度算出部２３４は、算出した値をレートリミット算出部２３５に出力する。この場合、該ＡＰＩキーに対応するアプリケーションが、一致度は低いものの、これまでの挙動と同様の挙動を行ったことが判別できるためである。

【０１１８】

一方、安定度算出部２３４は、「instance.transition」の遷移確率が状態遷移表２２５の遷移確率と所定の割合で一致すると判断した場合（ステップＳ４６：Ｙｅｓ）、該ＡＰＩキーに対応するアプリケーションの安定度が３であると算出する（ステップＳ４８）。そして、安定度算出部２３４は、算出した値をレートリミット算出部２３５に出力する。この場合、該ＡＰＩキーに対応するアプリケーションが、これまでの挙動に対し、一致度の高い挙動を行ったことが判別できるためである。

【０１１９】

そして、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度に応じて、「instance.key」についてのレートリミットを算出するレートリミット算出処理を行う（ステップＳ４９）。続いて、安定度算出部２３４は、状態遷移表２２５の「instance.key」についての遷移確率を「instance.transition」を用いて更新して（ステップＳ５０）、安定度算出処理を終了する。

【０１２０】

［レートリミット算出処理の手順］

次に、図１７に示すレートリミット算出処理の手順について説明する。図１８は、図１７に示すレートリミット算出処理の手順の一例を示すフローチャートである。

【０１２１】

図１８に示すように、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度が０、１、２または３のいずれであるかを判断する（ステップＳ５１）。

【０１２２】

レートリミット算出部２３５は、安定度算出部２３４が算出した安定度が０であると判断した場合（ステップＳ５１：０）、アクセス回数管理表２２６の「instance.key」のレ

ートリミットを最小値（例えば、１０）に更新する（ステップＳ５２）。
【０１２３】
また、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度が１であると判断した場合（ステップＳ５１：１）、アクセス回数管理表２２６の「instance.key」のレートリミットを減らす（ステップＳ５３）。
【０１２４】
また、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度が２であると判断した場合（ステップＳ５１：２）、アクセス回数管理表２２６の「instance.key」のレートリミットを変更しない（ステップＳ５４）。
【０１２５】
そして、レートリミット算出部２３５は、安定度算出部２３４が算出した安定度が３であると判断した場合（ステップＳ５１：３）、アクセス回数管理表２２６の「instance.key」のレートリミットを増やす（ステップＳ５５）。
【０１２６】
［実施例の効果］
このように、実施例に係るＡＰＩゲートウェイ２０は、ＡＰＩリクエストを送信したアプリケーションについて状態遷移インスタンスを作成し、状態遷移インスタンスが状態遷移表２２５に対してどの程度安定しているかを示す安定度を算出する。これによって、ＡＰＩゲートウェイ２０は、ＡＰＩリクエストの挙動が過去の挙動と比して変化したアプリケーションを判別する。そして、ＡＰＩゲートウェイ２０は、ＡＰＩリクエストの挙動が過去の挙動と比して変化したアプリケーションについては、単位時間あたりのレートリミットを下げている。これによって、ＡＰＩゲートウェイ２０は、改版アプリケーションや悪意アプリケーションに変化したアプリケーションに対するアクセス制限を実現する。したがって、実施例によれば、改版アプリケーションや悪意アプリケーションに変化したアプリケーションから、ＡＰＩで提供されるデータや機能を保護することが可能になる。
【０１２７】
［各装置の各構成要素］
なお、図示した各装置の各構成要素は機能概念的なものであり、必ずしも物理的に図示の如く構成されていることを要しない。すなわち、各装置の分散・統合の具体的状態は図示のものに限られず、その全部または一部を、各種の負荷や使用状況などに応じて、任意の単位で機能的または物理的に分散・統合して構成することができる。例えば、ＡＰＩゲートウェイ２０の制御部２３の各処理部が適宜統合されてもよい。また、各処理部の処理が適宜複数の処理部の処理に分離されてもよい。さらに、各処理部にて行なわれる各処理機能は、その全部または任意の一部が、ＣＰＵおよび当該ＣＰＵにて解析実行されるプログラムにて実現され、あるいは、ワイヤードロジックによるハードウェアとして実現され得る。
【０１２８】
［ＡＰＩ管理プログラム］
また、上記の実施例で説明した各種の処理は、あらかじめ用意されたプログラムをパーソナルコンピュータやワークステーションなどのコンピュータシステムで実行することによって実現することもできる。そこで、以下では、上記の実施例と同様の機能を有するプログラムを実行するコンピュータシステムの一例を説明する。図１９は、ＡＰＩ管理プログラムを実行するコンピュータを示す図である。
【０１２９】
図１９に示すように、コンピュータ１３００は、ＣＰＵ（Central Processing Unit）１３１０、ＨＤＤ（Hard Disk Drive）１３２０、ＲＡＭ（Random Access Memory）１３４０を有する。これら１３１０～１３４０の各部は、バス１４００を介して接続される。
【０１３０】
ＨＤＤ１３２０には上記のＡＰＩゲートウェイ２０の制御部２３と同様の機能を発揮す

るＡＰＩ管理プログラム１３２０ａが予め記憶される。すなわち、ＡＰＩ管理プログラム１３２０ａは、ＡＰＩゲートウェイ２０と同様の機能を発揮するプログラムとを有する。なお、ＡＰＩ管理プログラム１３２０ａについては、適宜分離してもよい。また、ＡＰＩ管理プログラム１３２０ａは、ＡＰＩゲートウェイ２０の通信Ｉ／Ｆ部２１及び記憶部２２と同様の機能を発揮するプログラムをさらに有してもよい。

【０１３１】

また、ＨＤＤ１３２０は、各種情報を記憶する。例えば、ＨＤＤ１３２０は、ＯＳや範囲選択の要求分散に用いる各種データを記憶する。

【０１３２】

そして、ＣＰＵ１３１０が、ＡＰＩ管理プログラム１３２０ａをＨＤＤ１３２０から読み出して実行することで、実施例の各処理部と同様の動作を実行する。すなわち、ＡＰＩ管理プログラム１３２０ａは、ＡＰＩゲートウェイ２０の制御部２３と同様の動作を実行する。

【０１３３】

なお、上記したＡＰＩ管理プログラム１３２０ａについては、必ずしも最初からＨＤＤ１３２０に記憶させることを要しない。

【０１３４】

例えば、コンピュータ１３００に挿入されるフレキシブルディスク（ＦＤ）、ＣＤ－ＲＯＭ、ＤＶＤディスク、光磁気ディスク、ＩＣカードなどの「可搬用の物理媒体」にプログラムを記憶させておく。そして、コンピュータ１３００がこれらからプログラムを読み出して実行するようにしてもよい。

【０１３５】

さらには、公衆回線、インターネット、ＬＡＮ、ＷＡＮなどを介してコンピュータ１３００に接続される「他のコンピュータ（またはサーバ）」などにプログラムを記憶させておく。そして、コンピュータ１３００がこれらからプログラムを読み出して実行するようにしてもよい。

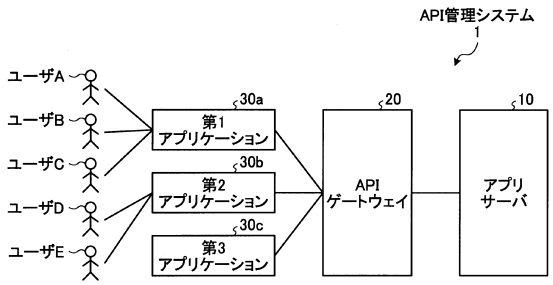
【符号の説明】

【０１３６】

１　ＡＰＩ管理システム
１０　アプリサーバ
２０　ＡＰＩゲートウェイ
２１　通信インタフェース（Ｉ／Ｆ）部
２２　記憶部
２３　制御部
３０　アプリケーション
３０ａ　第１アプリケーション
３０ｂ　第２アプリケーション
３０ｃ　第３アプリケーション
２２１　ＩＤ管理表
２２２　ＡＰＩキー管理表
２２３　ＡＰＩ一覧表
２２４　リクエスト履歴管理表
２２５　状態遷移表
２２６　アクセス回数管理表
２３１　ＡＰＩゲートウェイ管理部
２３２　新規アプリ登録部
２３３　状態遷移インスタンス作成部
２３４　安定度算出部
２３５　レートリミット算出部
２３６　リクエスト受付可否判定部

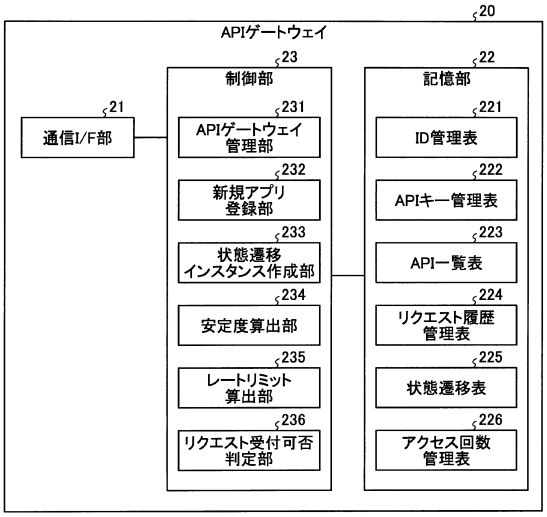
【図１】

本実施例に係るAPI管理システムの構成を示す図



【図２】

APIゲートウェイの構成を示すブロック図



【図３】

ID管理表のデータ構造の一例を示す図

221

ユーザ	user_id
ユーザA	abc
ユーザB	def
ユーザC	efg
ユーザD	hij
ユーザE	klm

【図４】

APIキー管理表のデータ構造の一例を示す図

222

アプリケーション	APIキー
第1 アプリケーション	123
第2 アプリケーション	456
第3 アプリケーション	789

【図５】

API一覧表のデータ構造の一例を示す図

223

URI	HTTPメソッド			
	GET	POST	PUT	DELETE
/attributes/{user_id}	ユーザの属性を取得する	禁止	ユーザの属性を更新する (ユーザ本人のみ更新可能)	禁止
/data/{id}	データを取得する	禁止	データを更新する	データを削除する
/data	データ一覧を取得する	データを新規追加する	禁止	禁止

【図６】

リクエスト履歴管理表のデータ構造の一例を示す図

224

APIキー	ユーザID	リクエストURI	HTTPメソッド	パラメータ	時刻
123	abc	/attributes/abc	GET		2015/06/08 09:00:00
123	def	/attributes/def	GET		2015/06/08 09:01:00
123	abc	/info/123	GET		2015/06/08 09:02:00
123	abc	/info	POST	新しいデータの中身	2015/06/08 09:03:00

【図７】

状態遷移表のデータ構成の一例を示す図

225

アプリケーション	APIキー	state 1	state 2	遷移確率
第1アプリケーション	123	GET/attributes/{user_id}	GET/info/{id}	0.6
	123	GET/attributes/{user_id}	POST/info	0.1
	…	…	…	…
第2アプリケーション	456	…	…	…
	456	…	…	…
	…	…	…	…
第3アプリケーション	789	…	…	…
	789	…	…	…
	…	…	…	…

【図８】

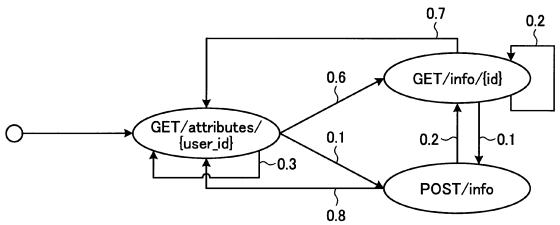
アクセス回数管理表のデータ構造の一例を示す図

226

アプリケーション	アクセス回数	レートリミット
第1アプリケーション	100	300
第2アプリケーション	200	200
第3アプリケーション	40	50

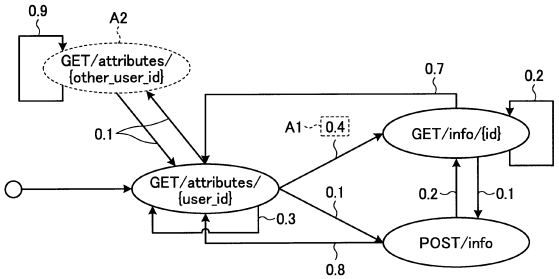
【図９】

第1アプリケーションについての各状態遷移の一例を説明するための図



【図１０】

第1アプリケーションについての各状態遷移の
挙動の変化の一例を説明するための図



【図１１】

状態遷移インスタンスの一例を示す図

アプリケーション	APIキー	state 1	state 2	遷移確率
第1アプリケーション	123	GET/attributes/{user_id}	GET/info/{id}	0.4
	123	GET/attributes/{user_id}	POST/info	0.1
	⋮	⋮	⋮	⋮
	123	GET/attributes/{other_user_id}	GET/attributes/{user_id}	0.1
	123	GET/attributes/{other_user_id}	GET/attributes/{other_user_id}	0.9
	123	GET/attributes/{user_id}	GET/attributes/{other_user_id}	0.1

A11, 225a, A12 新しい状態を含む 安定度0, A13

【図１２】

アクセス回数管理表の更新を説明する図

(a)

226

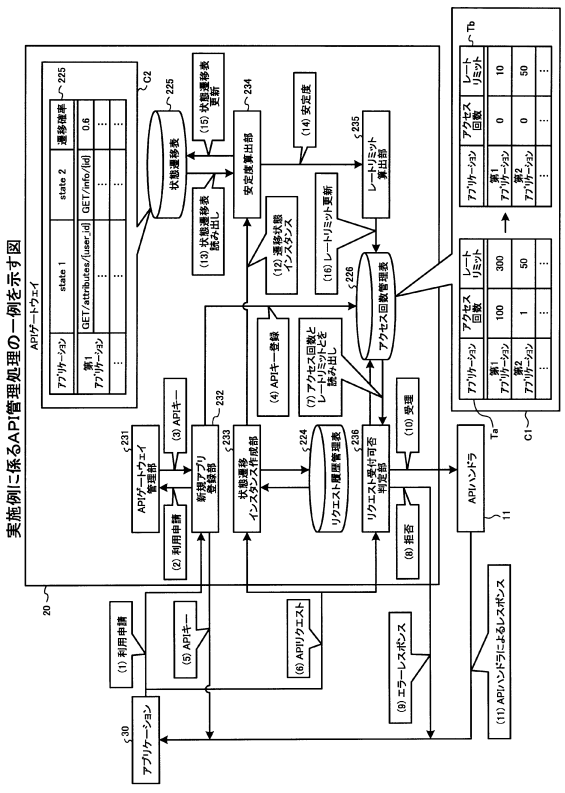
アプリケーション	アクセス回数	レートリミット
第1アプリケーション	100	300
第2アプリケーション	200	200
第3アプリケーション	40	50

(b)

226a

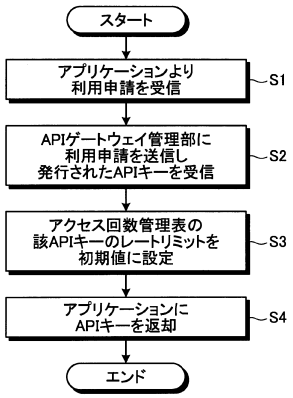
アプリケーション	アクセス回数	レートリミット
第1アプリケーション	0	10
第2アプリケーション	0	200
第3アプリケーション	0	50

【図１３】



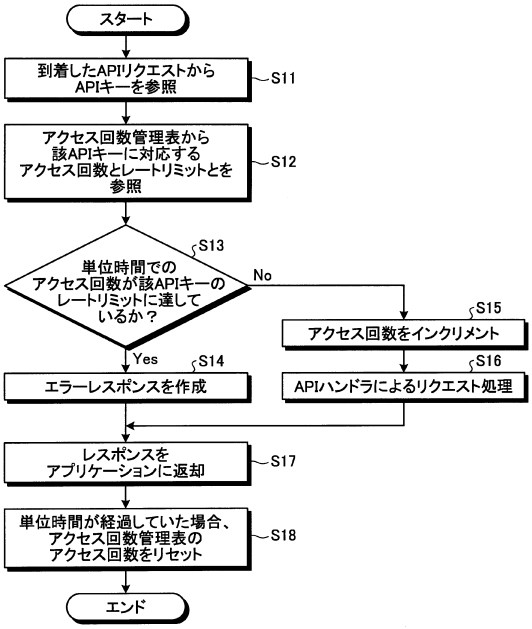
【図１４】

実施例に係る新規アプリ登録処理の手順の一例を示すフローチャート



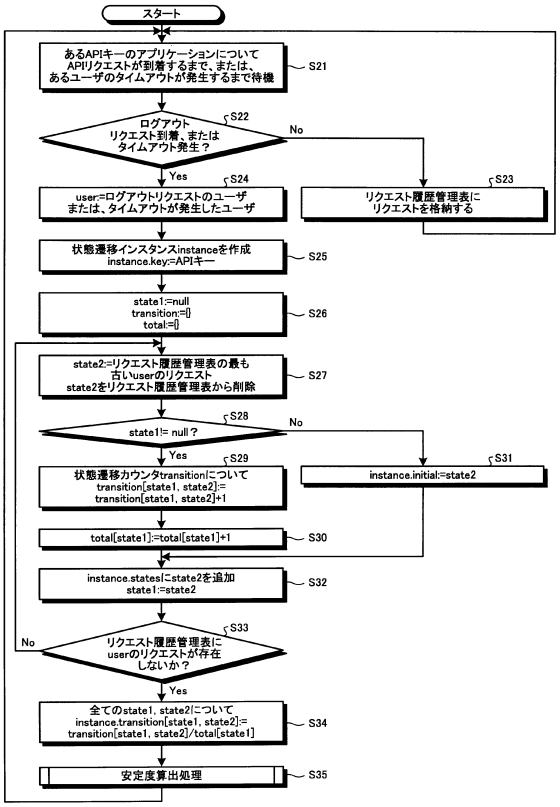
【図１５】

実施例に係るリクエスト受付可否判定処理の手順の一例を示すフローチャート



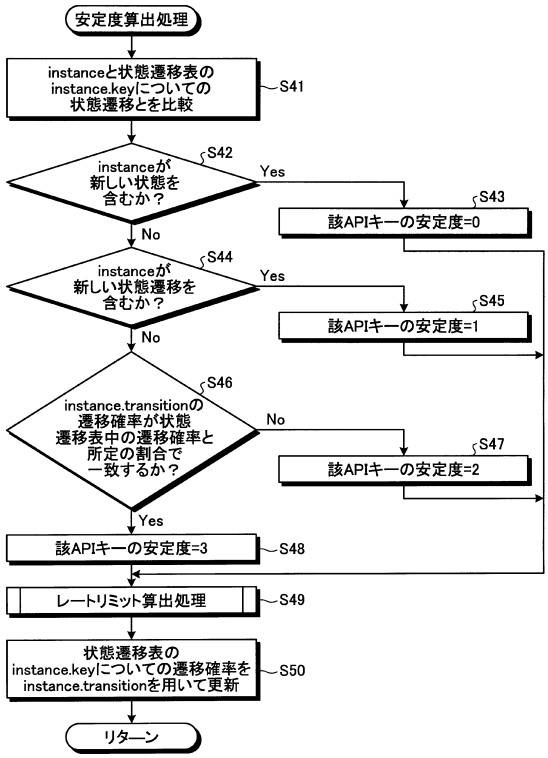
【図１６】

実施例に係る状態遷移インスタンス作成処理の手順の一例を示すフローチャート



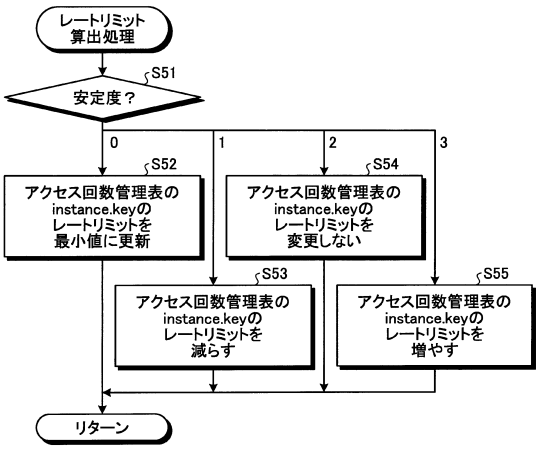
【図１７】

安定度算出処理の手順の一例を示す図



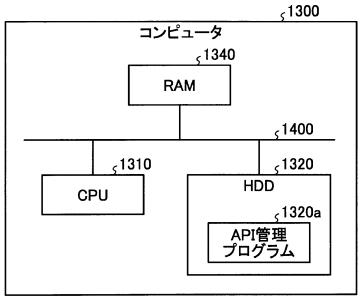
【図１８】

レートリミット算出処理の手順の一例を示す図



【図１９】

API管理プログラムを実行するコンピュータを示す図



フロントページの続き

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ　１１／０７
Ｇ０６Ｆ　１１／３４