自律コンピューティングのための運用知識を共有できる自己修復技術

Self-Healing Technique Based on Shared Operation Knowledge for Autonomic Computing

善明 晃由 ■ ZENMYO Teruyoshi
吉田 英樹 ■ YOSHIDA Hideki
木村 哲郎 ■ KIMURA Tetsuro

情報システムの自律的な運用を目指す技術の一つとして，システム障害の検知から修復までを人手を介さずに自律的に動作させる自己修復技術が注目を集めている。自己修復の動作は運用知識と呼ばれる情報が規定するため，的確な修復動作ができるかどうかは運用知識の品質により決まる。

東芝は，運用知識をコンポーネント単位で構成し，情報システム間で共有できる次世代の自己修復技術を開発している。運用知識が多くの情報システム間で共有されることにより，多数のシステム管理者が協力して運用知識をより良いものしていくことができる。

Self-healing functionality, from detecting system failures to troubleshooting, is a promising technique for reducing the management cost of information systems. However, the adoption cost is very high because a variety of operation knowledge related to failures is necessary.

Toshiba has been developing a technique that promotes the efficient construction of self-healing systems by reusing operation knowledge. The improvement of reusability makes accumulation and sharing of operation knowledge possible, facilitating the efficient construction of self-healing systems.

1 まえがき

企業の基幹業務や社会インフラなど重要な役割を担う情報システムが増加するとともに，故障やシステム障害への迅速かつ的確な対応ができる運用管理体制の重要性が増している。しかし，経験豊富なシステム管理者を24時間体制で待機させるような人手による管理は限界に達しており，故障やシステム障害の検知から修復までの一連の処理を自動的に行う，自己修復技術が注目を集めている。

自己修復技術としては，情報システムの一部として稼働しているサービスプログラムの稼働状態を監視し，障害の検知から再起動までを自動的に行う技術が既に実用化され広く利用されている。しかし，今後情報システムの大規模化及び複雑化が進むにつれ，サービス間の依存関係が複雑化し，故障の検知から原因の特定，そして復旧という一連の処理の自動化が飛躍的に難しくなることが予想される。このため，複雑な依存関係にも対応できる，次世代の自己修復技術が求められている。

自己修復機能は，故障を検知してから原因を特定し，復旧方法を策定して実行するまでの処理を，運用知識と呼ばれる情報に基づき実行する。このため，用意された運用知識では次に示す品質が非常に重要となる。

(1) 網羅性　可能性のあるすべての障害をカバーできているか
(2) 正当性　障害に対する適切な復旧方法が導出できるか

これまで，運用知識は，システムに固有なものとして情報システムごとに構築されてきた。更に，自己修復機能が集中管理型であったことから，運用知識もシステム全体に対応して一体化された形となっていた。しかし，このような運用知識の管理の枠組みでは，ある情報システムの運用知識に施された固有の改善は，他のシステムの運用知識に簡単には適用できない。そのため，情報システムの大規模化及び複雑化が進行すると，正当性や網羅性などの運用知識の品質の維持や向上が難しくなると予想される。

東芝は，情報システムを構成するウェブサーバやデータベースといったソフトウェア コンポーネントが広く共通に使われていることに着目し，運用知識をシステム単位ではなく，コンポーネント単位で構成する自己修復技術を提案している。この自己修復技術では，システム固有情報やコンポーネント間の依存関係を運用知識から排除することに成功し，再利用性のある運用知識として扱うことが可能になる。

2 運用知識の蓄積・共有による効率的な自己修復システムの構築

自己修復機能を組み込んだ情報システムの構成はそれぞれ異なるため，自己修復システムが必要とする運用知識もシステムに固有なものとなる。しかし，自己修復システムを構成する個々のソフトウェア コンポーネントは多くのシステムで共通に使われており，同じコンポーネントに関する運用知識は，組み込

まれているシステムが異なっても本質的には同じと見なすことができる。

しかし，実際の運用知識には，IP（Internet Protocol）アドレスやホスト名などのシステム固有情報が含まれていたり，依存関係にあるほかのコンポーネントに関する情報が入り込んでいるため，運用知識をコンポーネントごとに整理しても，そのままでは再利用できない。

運用知識を，システム固有情報やコンポーネント間の依存関係などの情報を排除した形で記述できれば再利用が可能になり，個々の情報システムの枠を越えて蓄積と共有が可能になる。このコンセプトを更に発展させれば，オープンソース ソフトウェア開発のモデルに倣い，多くのシステム管理者が協力して運用知識の品質を向上させることができると考えられる。

運用知識を蓄積・共有することの効果を**図1**に示す。まず，共有された運用知識の再利用により，自己修復システムの構築が効率化できる。次に，情報システム運用の経験を蓄積していくことで，運用知識の品質を向上させることができる。ある情報システムに配置された運用知識は，そこでの経験を基に修正されていく。この修正の蓄積が，運用知識の品質を向上させる。更に，様々な情報システムでの経験が蓄積された運用知識を用いることで，品質の高い運用知識を備えた自己修復システムを実現できる。

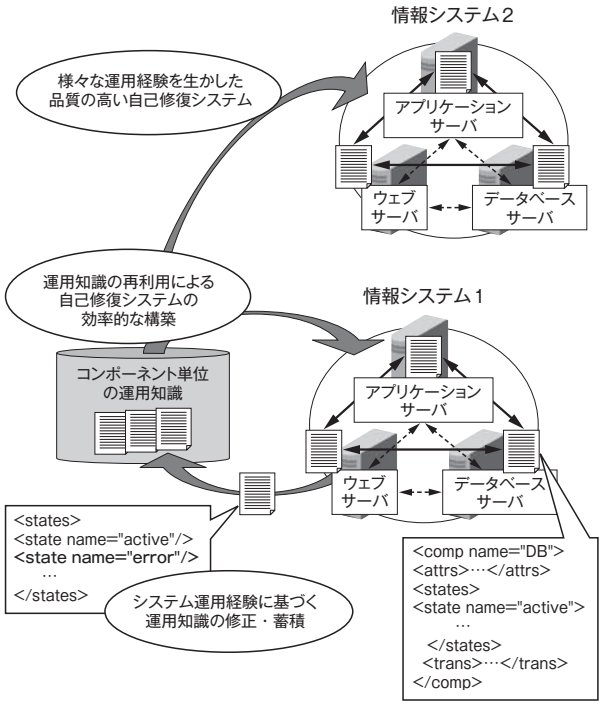

図1．運用知識を蓄積・共有することの効果 — 共有された運用知識の再利用による自己修復システム構築の効率化や，経験の蓄積による運用知識の品質向上が期待できる。

Benefits of sharing operation knowledge

3　運用知識の再利用性向上

ある情報システム用に構成された運用知識を，ほかの情報システムで再利用できるようにするための要件と，その実現方法について次に述べる。

3.1　再利用可能な運用知識の要件

運用知識の再利用性を低下させる要因は，以下の2種類の情報が運用知識の中に埋め込まれることである。

(1)　システム固有情報

(2)　コンポーネント間の依存関係

情報システムによって異なるシステム固有情報が運用知識に混在すると，再利用性を低下させる。また，コンポーネント間の依存関係は情報システムによって異なるため，様々なコンポーネントの情報が密に関連している運用知識は再利用性が低い。

これらの再利用性を低下させる要因を，**図2**に示す運用知識を例に説明する。この運用知識は，サーバ1のアプリケーションサーバAがサーバ2のデータベース管理システムB（DBMS-B：DataBase Management System-B）との接続に失敗している場合，DBMS-Bの状態を確認し，問題があれば再起動すればよいことを示している。

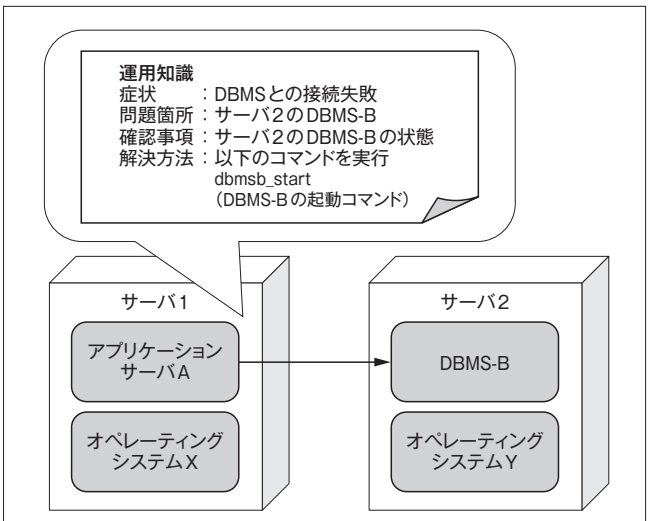

図2．運用知識の例 — 様々なシステム固有情報が混在すると，運用知識の再利用性が低下する。

Example of operation knowledge

システム固有情報とは，図2の例では“サーバ2”や“DBMS-B”といった，その情報システムに固有な情報である。この運用知識をほかの情報システムで利用するには，アプリケーションサーバAが依存するDBMSの配置場所の情報（ホスト名，IPアドレス）などが異なり，そのままでは適用できない。再利用性を高めるためには，このようなシステム固有の依存関係情報をコンポーネントの運用知識から取り除く必要がある。

一般論文

次に，コンポーネント間の依存関係とは，運用知識の中に複数のコンポーネントの知識が混在することを指す。図2の例では，アプリケーションサーバAで検知された障害の修復方法として，DBMS-Bの起動コマンドが記述されている。ほかの情報システムでこの知識を利用する場合，アプリケーションサーバAが依存するDBMSの種類が異なると，状態確認や起動の方法を修正しなければならず，再利用性が低い。再利用性を高めるためには，アプリケーションサーバA側の障害検知に関する知識と，依存先であるDBMS側の修復方法に関する知識を独立させておく必要がある。

3.2　運用知識の分散管理と情報システムごとの依存関係情報の注入

ここでは，再利用性向上のための運用知識の形式とシステム固有情報の扱いを説明する。

まず，複数のコンポーネントに関する情報が混在するのを防ぐため，運用知識はコンポーネント単位で分散管理する。図2の例では，症状や問題箇所に関する情報はアプリケーションサーバA側で管理し，確認事項や解決方法に関する情報はDBMS-B側で管理する。

次に，システム固有である依存関係に関する情報をコンポーネントの運用知識から取り除く。通信先サーバのホスト名やDBMSの種類などのシステム固有情報は，一般に，各コンポーネントの設定ファイルなどに定義される。そこで，運用知識からホスト名などのシステム固有情報を取り除き，代わりに設定ファイルのパラメータが指定される箇所への参照に置き換える。このようにして，運用知識をシステムに依存しないものにして蓄積・共有する。取り除いた固有の依存関係情報は，コンポーネントを各情報システムに配置する際に，そのコンポーネント用の運用知識に組み込む。

これを依存関係情報の注入といい，一例を**図3**に示す。図3は，DBMSに依存するアプリケーションサーバの運用知識が，ある情報システムに配置されるようすを示している。共有されている状態の運用知識では，DBMSの配置先ホスト名などのシステム固有情報は，設定ファイルのパラメータへの参照として記述されている。参照として記述されたシステム固有情報は，各情報システムに配置される際に，設定ファイルで定義された実際の値に置き換えられる。

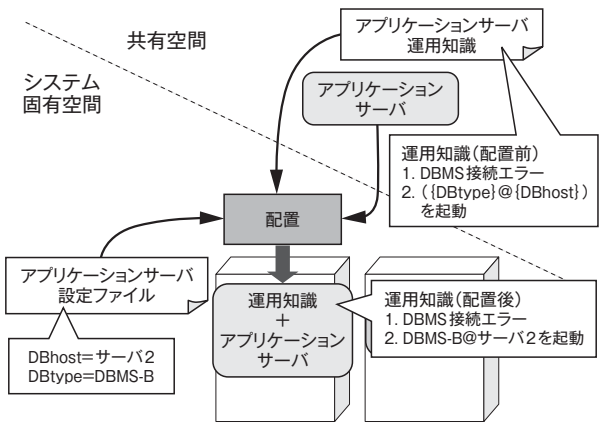

図3．依存関係情報注入の例 — システム固有の依存関係情報は，コンポーネントを情報システムに配置する際に注入される。

Treatment of system-specific dependency information

3.3　抽象的インタフェースによるコンポーネント操作

ここでは，あるコンポーネントの運用知識にほかのコンポーネントの情報が混入しないようにするための依存関係の扱いについて説明する。

障害検知に関する知識と修復方法に関する知識の関連を疎にするために，抽象的インタフェースを介したコンポーネント操作を行う。抽象的インタフェースでは，例えば，“起動”，“停止”などコンポーネントの種類に依存しない抽象的な命令表現を用いる。その結果，外部コンポーネントの詳細な情報がなくともそれを操作することができ，障害検知に関する運用知識を修復操作の対象となるコンポーネントとは独立したものにできる。

4　自己修復システムの構成

共有された運用知識に基づく自己修復システムの構成を**図4**に示す。ここでは，コンポーネントを依存元コンポーネントと依存先コンポーネントに分類する。依存元コンポーネントの運用知識としては，正常動作するために必要な依存先についての制約条件（制約知識）を蓄積・共有する。依存先コンポーネントについては，その監視方法や起動・停止方法などに関する運用知識（操作知識）を蓄積・共有する。

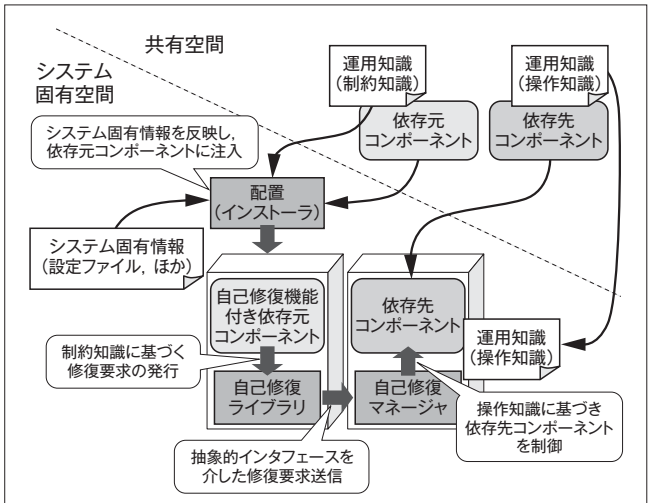

図4．自己修復システムの全体像 — 従来の構築プロセスに影響を与えることなく，自己修復機能の導入が実現できる。

Overview of self-healing system based on proposed technique

各コンポーネントの運用知識は，対応するコンポーネントとともに共有空間で管理され，個々の情報システムに配置される際にコンポーネントとともに配置される。このとき，制約知識は，設定ファイルなどシステム固有情報を反映して依存元コンポーネントに注入する。更に，依存元コンポーネントには，修復要求の送信機能を提供する自己修復ライブラリの呼出し機能も注入する。これに対して，操作知識は，依存先コンポーネントと同じサーバに配置される自己修復マネージャが参照できるようにしておく。

図4の情報システムで自己修復機能は，依存元コンポーネントが自己修復ライブラリを呼び出すことで開始される。呼び出された自己修復ライブラリは，制約知識に基づいて依存先コンポーネントが配置されるサーバの自己修復マネージャに，抽象インタフェースを介して修復要求を送信する。修復要求を受けた自己修復マネージャは，依存先コンポーネントの操作知識に基づいて，依存先コンポーネントの状態を修復要求に指定される状態に制御する。

自己修復機能の導入は，インストーラの拡張などにより，従来の情報システム構築プロセスを変化させることなく実現できる。また，自己修復マネージャは，抽象的インタフェースを介して受信した修復要求に対する処理を，操作知識に基づいて局所的に実行する。操作に関する情報の詳細が外部から隠蔽（いんぺい）されるため，制約知識から依存先コンポーネントの情報を排除できる。

5　あとがき

ここでは，自己修復動作，すなわち障害の検知から復旧処理までの動作を規定する運用知識の再利用が可能な，次世代の自己修復技術について述べた。この自己修復技術では，運用知識をシステム単位ではなくコンポーネント単位で構成し，しかもシステム固有情報が排除された形で記述できるため，あるコンポーネントを利用するほかの情報システムにも簡単に適用することができる。更に，個々の情報システムの運用経験が反映された運用知識を蓄積・共有することで，多くのシステム管理者が協力して運用知識の品質を向上させることが可能になる。

当社は，これまで，プロトタイプ実装[1]などを通してコンセプトの具体化を行ってきた。今後は，実用化に向けて，大規模な情報システムでの適用実験などを行っていく予定である。

文　献

(1) Zenmyo, T., et al. "A self-healing technique based on encapsulated operation knowledge". Proceedings of the third IEEE International Conference on Autonomic Computing (ICAC'06). Dublin, Ireland, 2006-06, IEEE Computer Society. IEEE Computer Society Press, 2006, p.25－32.

善明　晃由　ZENMYO Teruyoshi
研究開発センター　コンピュータ・ネットワークラボラトリー。分散システムの研究・開発に従事。
電子情報通信学会，情報処理学会会員。
Computer & Network Systems Lab.

吉田　英樹　YOSHIDA Hideki
研究開発センター　コンピュータ・ネットワークラボラトリー研究主務。グリッドコンピューティング及びクラスタシステムの研究・開発に従事。情報処理学会，ACM，IEEE会員。
Computer & Network Systems Lab.

木村　哲郎　KIMURA Tetsuro, Ph.D.
研究開発センター　コンピュータ・ネットワークラボラトリー主任研究員，工博。分散システム及び仮想化技術の研究・開発に従事。情報処理学会，ACM，IEEE会員。
Computer & Network Systems Lab.