D-Script: 障害対応スクリプトと回復戦略を行うスクリプトフレームワークの概要

中田晋平[†1]　菅谷みどり[†2]　倉光君郎[†3]

近年，システムの障害に対応するための対応策が失敗して障害を深刻化させる事例報告が相次いでいる．このような障害を避けるため，システムに起こるリスクは事前によく議論され，対策が練られているべきである．しかし，二次リスクが明るみに出るのは，実際に障害が起こってからである場合がほとんどであり，そのようなリスクを想定していないことが多い．これらを事前に議論し，対策を練るためには，まず，障害対応にどのようなリスクがあるのかを把握することが重要である．本論文では記述された障害対応から，キーワードを抽出し，キーワードを基にリスクを列挙するためのリスクデータベースの構築を提案する．

Task-orient Policy-based Failure Recovery Script Framework

SHINPEI NAKATA[†1]　MIDORI SUGAYA[†2]　KIMIO KURAMITSU[†3]

This paper presents the idea and design on script-based extensible framework for fault management in distributed open systems. Today's distributed systems are facing an increasing number of faults that are hard to predict at the design time, in part due to ever-lasting software updates. To safely apply such scripts, the modularization of scripts with D-Task and D-Control (based on business process management) is introduced with policy-based error handler of partial failures. In this paper, we discuss the property of each task in failure recovery workflow. According these property, we propose task recovery policy for each kind of tasks.

1. あらまし

近年、我々の社会生活の多くは情報システムに依存している．これらの情報システムに障害が起き，サービスが停止してしまうと，企業の経済低損失だけでなく社会生活も行えない深刻な事態を招いてしまう可能性がある．このような情報システム障害による被害を回避する，または被害を最小化する努力は，情報システムの提供側，利用側の双方で行われている．中でも，障害が発生した直後にとられる対応は，応急措置と呼ばれ，障害の原因特定よりも先にサービスの復旧を最優先としたものである．このため，復旧に必要な手順は即座に実行できるよう，予めシェルスクリプト化されている場合が多い．

しかし，近年，これら予め用意した手順に含まれていた間違いや，想定外の要因により，障害を悪化させてしまう事例が報告されている[1][2]．これは，障害対応手順が失敗した際の措置があまり考えられていないというよりも，障害対応手順を記述するためのシェルスクリプト言語の機能によるものだと我々は考えている．そもそも，シェルスクリプトはシステム運用を自動化することを主目的に発展してきた．システム運用は，様々なシステムコマンドの知識を持つ運用者が適切なコマンドを打ち込み，その出力を確認しながら次のコマンドを打つ，という作業となる．これらの作業を自動化するための言語は，簡単な制御構造や，データ構造を持ち，多くのシステムコマンドをそのまま利用できればよい．しかし，先に挙げた事例のように，記述された手順が失敗した場合の対処を行う仕組みへの考慮はあまりされておらず，自動化された作業が失敗した際に対応を行えず，問題となる．

そこで我々は，既に記述されているシェルスクリプトの失敗を外部から検出し，実行が失敗した場合の対応を記述できる仕組みである D-Script を提案する．

本概要では，D-Script が持つ概念の説明を行う．

2. D-Script の概念

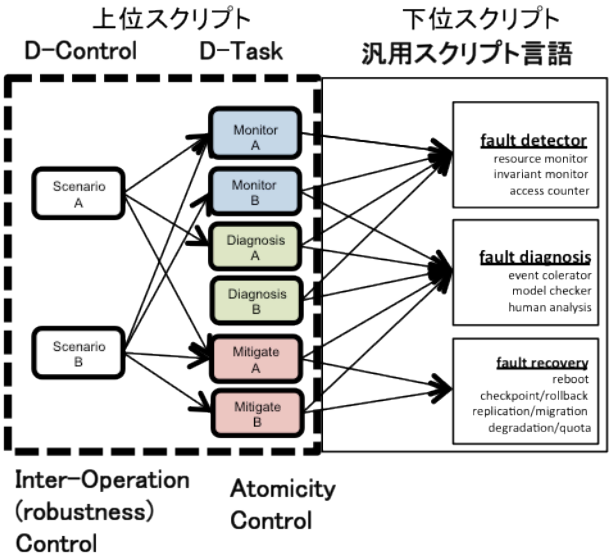
我々はまず，D-Script によるスクリプトの実行モデルである Script of Scripts について説明を行う．次に，障害対応手順をモデル化した障害対応ワークフローについて説明を行う．最後に，障害対応ワークフロー中のタスク（作業）が失敗した場合の対応方法について述べる．

2.1 Script of Scripts

障害対応を記述したスクリプトは，システムの各ノード内でそれぞれ個別に保管されている．我々は，これらヘテロジニアスに存在するスクリプトを管理，運用するための実行モデルを Script of Scripts と呼ぶ．我々は，一つのスクリプトで運用処理のすべてを記述しようとするのではなく，

†1 横浜国立大学大学院
Graduate School of Yokohama National University
†2 横浜国立大学
Yokohama National University
†3 JST/CREST
Japan Science and Technology Agency

処理を行うスクリプトと，運用管理を行うスクリプトを分離する，ということが基本的なアイディアである．Script of Scripts の説明図を図 1 に示す．我々は運用管理を行うスクリプトを D-Scenario，処理を行うスクリプトを D-Task と呼び，D-Task は D-Scenario からその運用を管理される．また，Script of Scripts では，D-Task は特定の目的を持つ論理単位で分割する．図ではシステムやアプリケーションの状態を監視する Monitor，障害の解析を行う Diagnose，障害被害の最小化を図る Mitigate などがある．



2.2 障害対応ワークフロー

障害対応は，途中で実行を失敗してしまうことがあり，部分障害(Partial Failure)と呼ばれる．部分障害に対処するためには，部分障害で失敗した仕事だけを再度実行したり，代替したりする必要がある．そこでまず，障害対応を適切な仕事粒度で分割しておく必要がある．我々は，適切な仕事粒度で分割された作業（仕事の流れ）を障害対応ワークフローと呼ぶ．ワークフロー中の仕事をタスクと呼び，これら，どのタスクも潜在的に障害が発生して失敗する可能性を持つ．

2.3 障害対応タスクの分類

障害対応ワークフロー中のどのタスクも潜在的に失敗する可能性があるため，失敗した場合の対応を考える必要が有る．簡易な対応としては，そのタスクをやり直すことが考えられる．しかし，タスクの内容によっては，単純にやり直すことが難しいこともある．例えば，ファイルの削除を行う処理がタスクに含まれていた場合，削除後にタスクが失敗しても，ファイルの復元ができない限りやり直しは難しい．より一般的に言えば，外部の環境を変化させてしまう処理がタスクに含まれていた場合は，再実行が困難になる．この例のように，タスクの内容に応じて回復戦略が異なってくるため,我々はまずタスクを以下の通り分類した．

2.3.1 可逆タスク

可逆タスクとは，実行した処理を元に戻すことができるものを指す．例えば，ウェブサーバを起動するというタスクは可逆タスクである．なぜなら，ウェブサーバを停止するという作業を行うことで状態を元に戻すことができるためである．可逆タスクが失敗した場合は，単純に状態を戻す作業を行い，再実行を行う．ここで，可逆タスクに起こった障害の種類に応じて対応が異なってくる．再実行でタスクが成功した場合は，その障害は一時的なもの(Transient)である．しかし，再実行してもタスクが失敗する場合は，なんど再実行しても失敗してしまうため，別の代替タスクへ処理を切り替える必要が有る．

2.3.2 非可逆タスク

非可逆タスクとは，実行した処理を元に戻すことが困難，または不可能なタスクを指す．非可逆はチェックポイントなどをタスク実行前に置く事で可逆タスク化することはできるため，この場合の対処は可逆タスクに準じる．非可逆タスクで，元に戻すことを諦める場合は，他の処理で代替する必要がある．しかし，非可逆タスクに続くタスクは，非可逆タスクが成功することを前提とした処理であるため，非可逆タスクだけを代替タスクで置き換えて，元のワークフローへ戻る事は難しい．このため，非可逆タスクを諦めて処理を進める場合は，代替ワークフローを用意する必要が有る．代替ワークフローは，例えば機能の一部を諦めて縮退運用するようなものが挙げられる．

2.3.3 不変タスク

不変タスクとは，実行した処理が外部の状態を変化させないタスクを指す．例えば，システムの状態を監視するためのモニタリングは，外部の状態を変化させないタスクであるといえる（実際には CPU を利用するが，現在は CPU 負荷やメモリ，キャッシュの状態変化は無視できるものとして議論を進めている）．不変タスクが失敗した場合は，再実行を試み，再実行が失敗するようならそのタスクを省略して次のタスクへ進むことで障害対応を進めていけばよい．

3. まとめ

本概要では，我々の提案している D-Script の基本概念の説明を行った．我々は，障害対応ワークフロー中のタスクの性質に着目し，回復戦略をそれぞれに用意することで，障害対応失敗に備える試みを行っている．今後は，本概要で説明した議論を元に実際の障害対応へ適用していく．

謝辞 本研究は，JST/CREST「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」領域の研究助成を受けて行われた．

参考文献

1) http://itpro.nikkeibp.co.jp/article/NEWS/20120202/380044/?ST=NC
2) Google Apps – Gmail Incident Report, February 24, 2009, www.google.com/appsstatus/ir/1nsexcr2jnrj1d6.pdf