オープンシステムディペンダビリティの実現に向けた
領域全体の活動紹介

DEOSポスター集

２０１１年１１月

科学技術振興機構
Japan Science and Technology Agency

OSDとは、DEOSとは、その究極のメリットは

現代のコンピュータシステムは、

- システムの複雑化・巨大化
- システムのネットワーク化
- 構成要素のブラックボックス化、レガシーコードへの依存度増大

が進み、全体を把握し、詳細を理解し、完璧に構築する事は至難の業で、常に不完全です。

その上、

- サービス利用者の要求の変容、社会のルールや仕組みの変容
- 事業目的の変容、サービスの多様化
- ネットワークで接続された他システムの仕様の変容

などに常にさらされ、動作環境は不確実です。

すなわち、現代のコンピュータシステムは機能、構造、システム境界が変化しつづけるオープンシステム（開放系）であり、これに起因する**不完全さ**と**不確実さ**を完全に排除することができず、未来に障害となりうる要因（**開放系障害要因**）を本質的に抱えています。

私たちは、それらの要因を顕在化する前にできる限り取り除き、また、顕在化した後に迅速かつ適切に対応し、影響を最小とするようにマネージし、利用者が期待する便益をできる限り安全にかつ継続的に提供し、社会への責任ある説明、およびそれらを継続的に行う能力が必要だと考えています。

私たちは、この開放系対応力を「**OSD：オープンシステムディペンダビリティ（Open Systems Dependability）**」と呼び、このOSDを実用化し社会に生かすための、プロセスやアーキテクチャや要素技術などの知識・技術体系を「**DEOS：開放系のためのディペンダビリティ工学（Dependability Engineering for Open Systems）**」と名付けました。

DEOSの実行によるOSDの実現、すなわち「**変化しつづけるシステムのサービス継続と説明責任の全う**」のためには以下が必須であると考えています。

- ▶ 継続的な改良改善のための反復的なプロセス（**DEOSプロセス**）
- ▶ これを仕組みとして支えるアーキテクチャ（**DEOSアーキテクチャ**）
- ▶ 仕組みを実行する構成要素プロセス群・要素技術群

私たちは、DEOSの**究極のメリット**は次の３点であると考えます。

- ◆ エンドユーザーのサービス享受の権利を護る
- ◆ サービス提供者のビジネス（ブランド）を護る
- ◆ サービス提供経営陣が社会的責任を果たす

DEOSプロジェクト

●DEOSプロジェクトは(独)科学技術振興機構の戦略的創造研究推進事業CRESTの研究領域のひとつである「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」における研究開発です。

●DEOSプロジェクトは総括・副総括のもと９つの研究機関が参加しています。

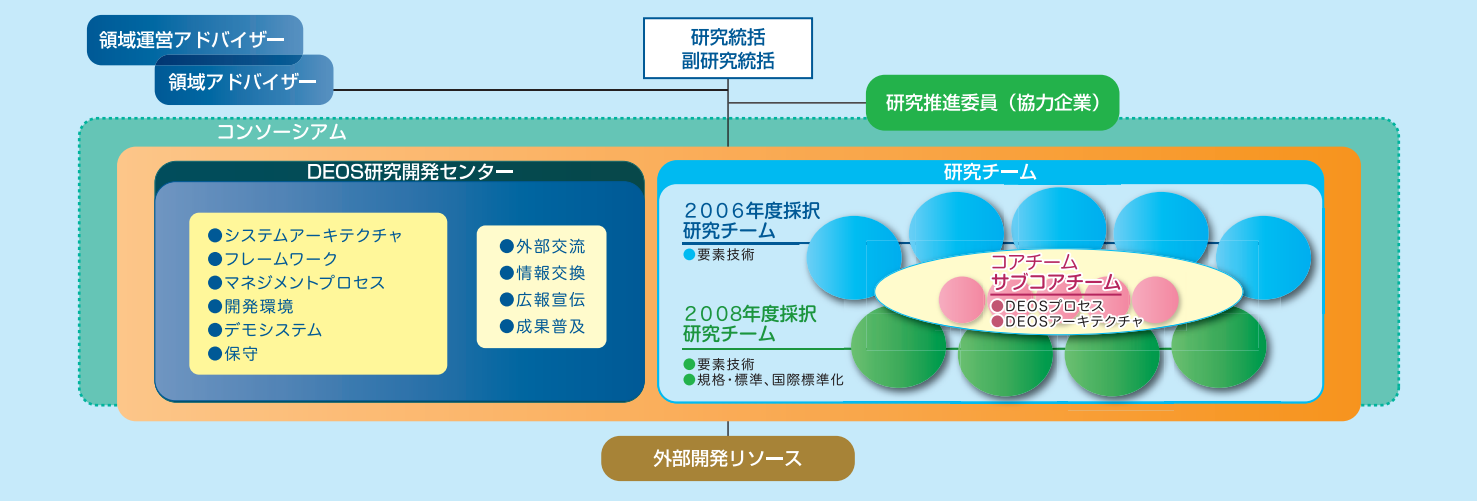
●DEOSプロジェクトでは、変化しつづけるシステムのサービス継続と説明責任の全うを目指したソフトウェア開発・保守のための手法や技術の研究・開発を行います。

●DEOSプロジェクトでは、研究・開発の成果を生かして成果の国際標準化を行いコンソーシアムを設立することを目指します。

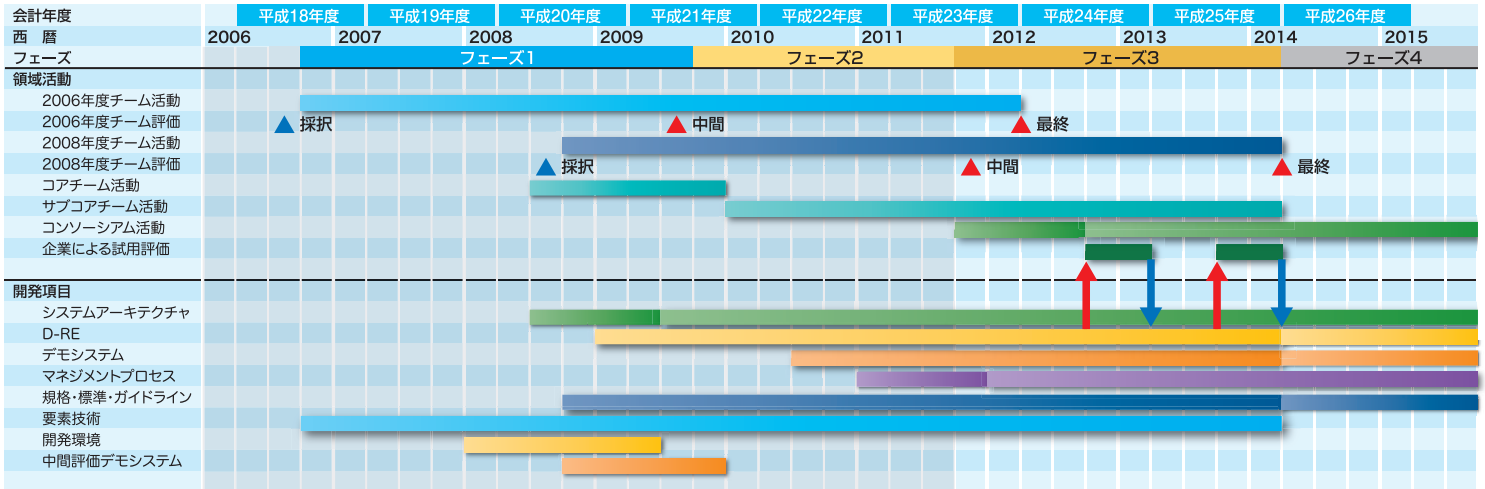
■DEOSプロジェクト参加研究機関

研究総括	
所　眞理雄	株式会社ソニーコンピュータサイエンス研究所　代表取締役会長
副研究総括	
村岡　洋一	早稲田大学　理工学術院　教授
H18年度採択　研究代表者	
石川　裕	東京大学　情報基盤センター　教授
佐藤　三久	筑波大学　計算科学研究センター　センター長
徳田　英幸	慶應義塾大学　環境情報学部　教授
中島　達夫	早稲田大学　理工学術院　教授
前田　俊行	東京大学　大学院情報理工学系研究科　助教
H20年度採択　研究代表者	
加賀美　聡	(独)産業技術総合研究所デジタルヒューマン工学研究センター 副センター長
木下　佳樹	(独)産業技術総合研究所　情報技術部門　主幹研究員
倉光　君郎	横浜国立大学　大学院工学研究院　准教授
河野　健二	慶應義塾大学　理工学部　准教授

■研究開発体制



■ロードマップ



DEOSプロジェクト連絡先

DEOSプロジェクトに関する詳細な情報は ☛ http://www.dependable-os.net/

DEOSプロジェクトに関する詳細な情報は ☛ ディペンダブル組込みOS研究開発センター
E-mail: center@dependable-os.net

DEOSプロセス、DEOSアーキテクチャ

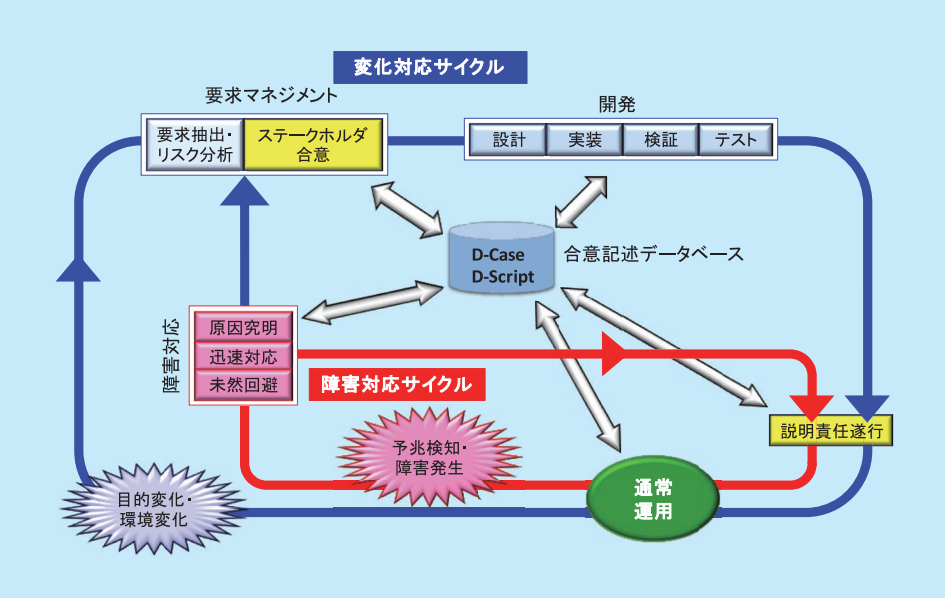
■ DEOSプロセス：オープンシステムディペンダビリティを実現するプロセス

◆ 反復的アプローチ

- 目的や環境の変化に対してシステムを継続的に変更して行くためのサイクル
- 障害に対して迅速に対応するためのサイクル

◆ プロセスのプロセス

- 相互に有機的に結びつけられた構成要素プロセス

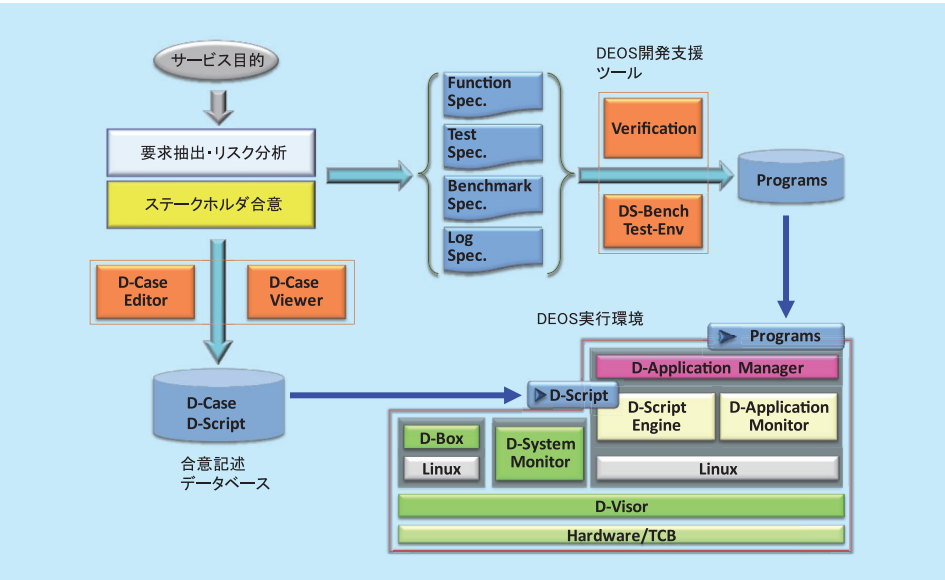


■ DEOSアーキテクチャ：DEOSプロセスを支援する仕組み

◆ 要求マネジメントプロセスを支えるツールや合意記述データベース

◆ ディペンダブルなソフトウェアを開発するためのツール群

◆ システムの監視・記録や障害時の迅速対応を担う実行環境



DEOSプロセスデモ

■ デモシナリオ

シナリオ 1

新サービスを追加したところ、サービス要求以上のアクセスが発生した為、予め合意記述データベースに登録されていた対処方法を実施する。原因究明の結果、新サービスが想定以上に好評だった事が判明した為、Webシステムの増強を決定

- システムの復元
- システム増強

シナリオ 2

サーバのレスポンスが急激に低下した為、予め合意記述データベースに登録されていた対処方法を実施する。原因究明の結果、バッチジョブが本来起動してはいけない時間に起動していた事が判明した為、バッチジョブに起動制限をかける事に決定

- 障害部位を切り離す
- バッチジョブの起動制限

シナリオ 3

メモリ使用量がシステムの許容値を超えた為、予め合意記述データベースに登録されていた対処方法を実施する。原因究明の結果、追加した新サービスにメモリーリークの疑いがある事が判明した為、アプリを修正し、サービスを継続する事に決定

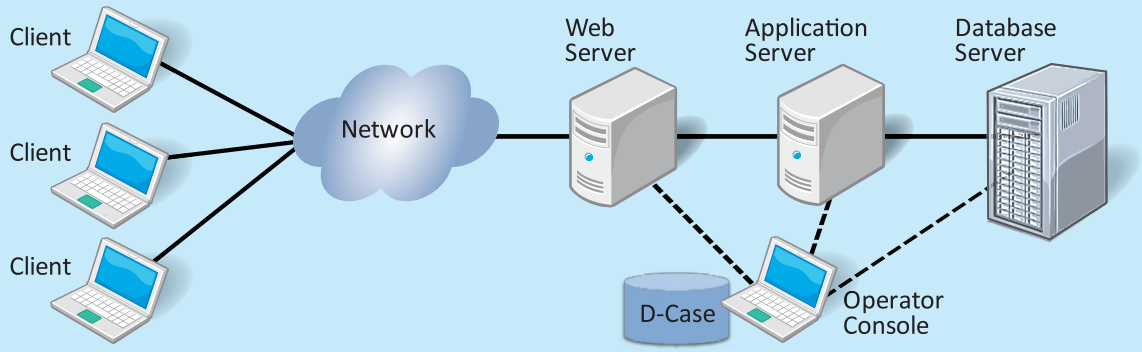
- コンテナ再起動
- 診断モジュール投入
- アプリを修正

シナリオ 4

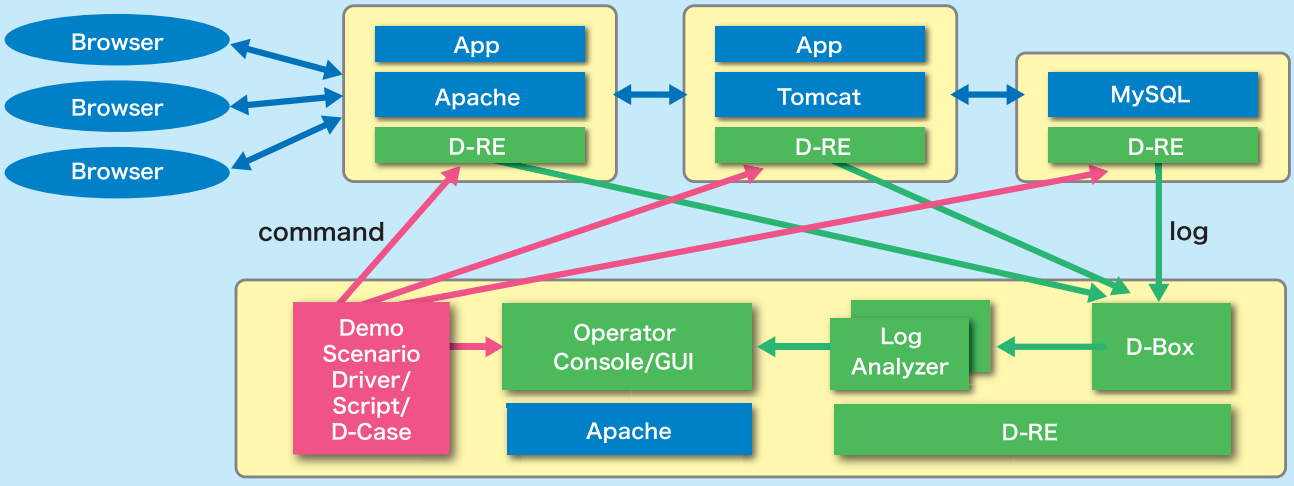
翌日のサービスが OK か確認したところ、サービスが実行できないという結果が得られた。原因究明の結果、ライセンス切れである事が判明した為、ライセンスを更新した

- 未来時間リハーサル

■ デモシステム物理構成図



■ デモシステム論理構成図



要求マネジメント

山本 修一郎(名古屋大学)

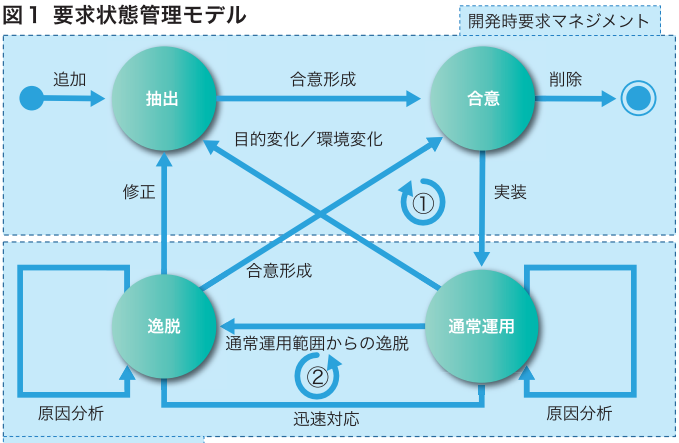
要求マネジメントでは、開発時と運用時を通じて要求の状態とプロセスをマネジメントすることにより、要求ビュウポイントから次の２つのサイクルのプロセス定義と可視化を支援する

①目的環境変化対応サイクル
②障害対応サイクル

■要求状態マネジメントモデル

開発時の要求状態には、ステークホルダから抽出された状態と、抽出された要求が合意された状態がある。
運用時の要求状態には、合意された要求が実装され通常運用されている状態と、通常運用範囲から逸脱した状態がある。
図１の要求状態を、要求ごとにマネジメントすることで、システムの状態を複数の要求状態の集合によって表現できる。
通常運用されている要求が、目的環境変化によって、変更が必要になると、変更要求として、新たに抽出される。
逸脱状態の要求では、迅速対応によって通常運用状態に遷移する場合、要求の実装に問題があるために再合意が必要になる場合、要求そのものに問題があるため要求変更が必要な場合がある。

図１ 要求状態管理モデル



①目的環境変化対応サイクル
②障害対応サイクル

■要求マネジメントプロセス

開発時の要求プロセス

①要求抽出
②リスク分析
③ステークホルダとの合意形成
④実装要求確認

運用時の要求プロセス

①説明責任遂行
②要求変更
③合意更新

要求プロセス活動の定義

目的、入力、出力、手順、開始条件、完了条件、役割分担に基づいて要求プロセス活動を明確に定義

図２ 要求マネジメントプロセス

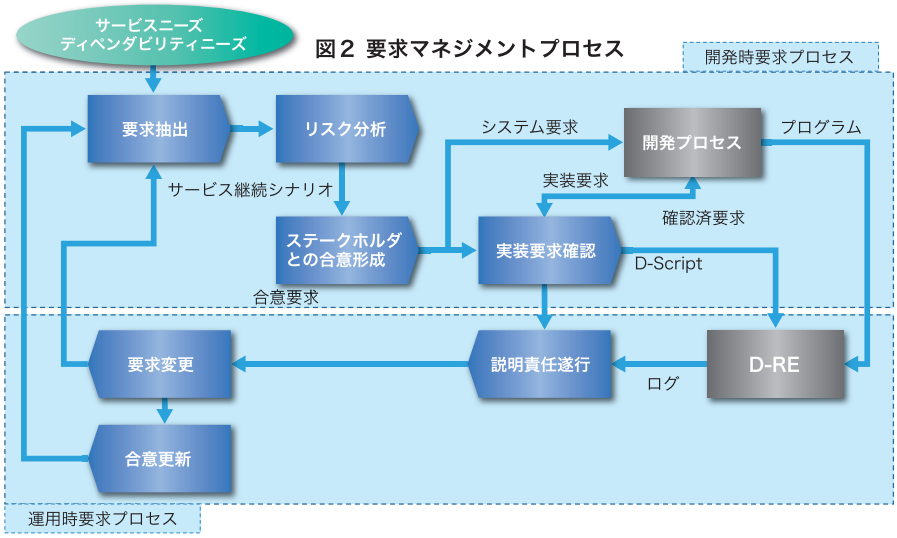


図３ リスク分析活動の例

目的	サービス要求リスクを分析し、サービス継続シナリオを作成する		
入力	サービス要求 ディペンダビリティ要求	出力	サービス継続シナリオ
手順	・サービス要求の逸脱を識別する ・逸脱原因への対策を決定する ・ディペンダビリティ要求を分析する ・ディペンダビリティ要求を満足するようにサービス継続シナリオを作成する		
開始条件	リスク分析規則が定義されている サービス要求、ディペンダビリティ要求が定義されている		
完了条件	リスク分析規則に基づいて、サービス要求とディペンダビリティ要求から、サービス継続シナリオが抽出されている		
役割	サービス、製品の提供者	サービス要求とディペンダビリティ要求を提示する	
	システム提供者	サービス継続シナリオを確認する	

プログラム検証による
オープンシステムディペンダビリティ支援

研究代表者：前田俊行（東京大学）
（DEOS検証研究チーム）

■プログラム検証とは

プログラムがある性質（例えばメモリエラーが生じないことなど）を満たすことを「証明」すること

●「テスト」との違い

―テストはエラーが無いことの「証明」まではしてくれない

■プログラム検証の限界とオープンシステムディペンダビリティ

プログラム検証
事前に全く想定していないような性質の検証はできない

対立

オープンシステム ディペンダビリティ
事前に予測・想定できなかった障害要因（開放系障害要因）に対してもサービスを継続しようとすること

オープンシステムディペンダビリティに対してプログラム検証はどのような貢献ができるか？

■本研究のアプローチ

2つのプログラム検証手法を補完的に用いる

検査型 & **モデル検査**

開放系障害要因に対応するための
プログラム開発・修正の安定性・継続性向上を目指す

■本研究の具体的な手法

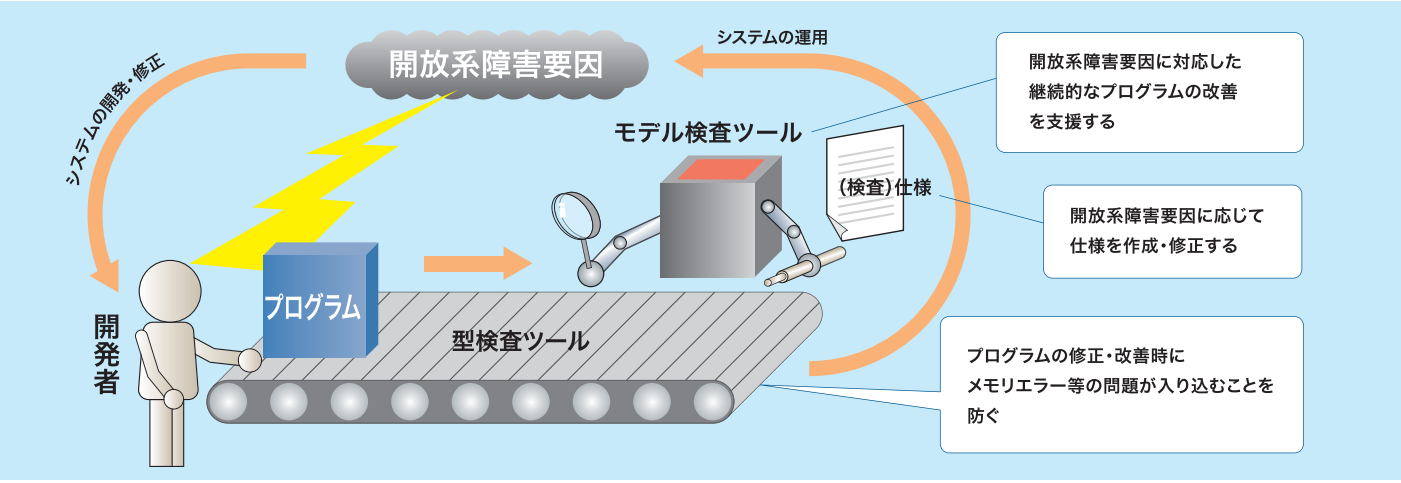
●C言語を対象にした型検査ツールの設計・実装
●C言語を対象にしたモデル検査ツールの設計・実装

―システムソフトウェアの仕様をモデル検査ツールに与えるための 仕様記述言語の設計

■2つの検証ツールの比較

	型検査ツール	モデル検査ツール
検証できる安全性	メモリ安全性など基礎的な安全性	API使用の安全性など高度な安全性
検証の対象	Cのソースコード 機械語プログラム	Cのソースコード
仕様記述	ほとんど必要なし	必要あり（検証したい性質を仕様記述言語で書く等）
実行時間	短い	長い

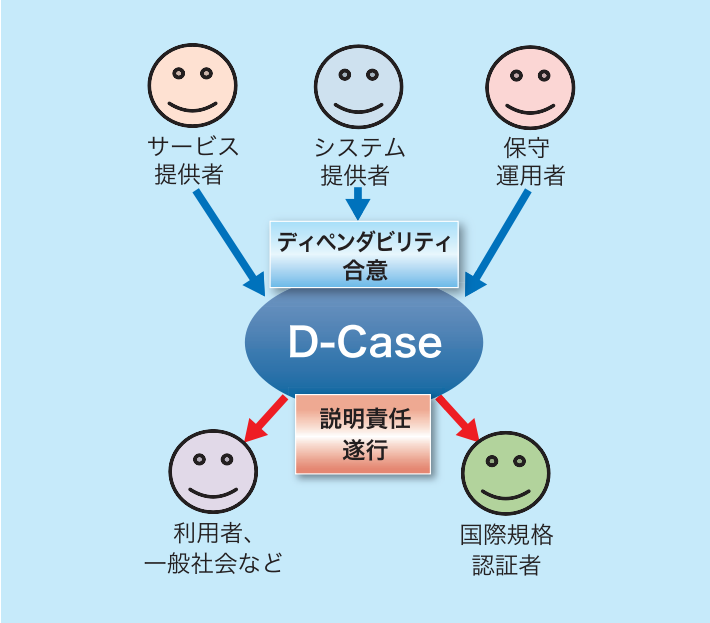
■2つのツールを用いた開放系障害要因への対応のイメージ



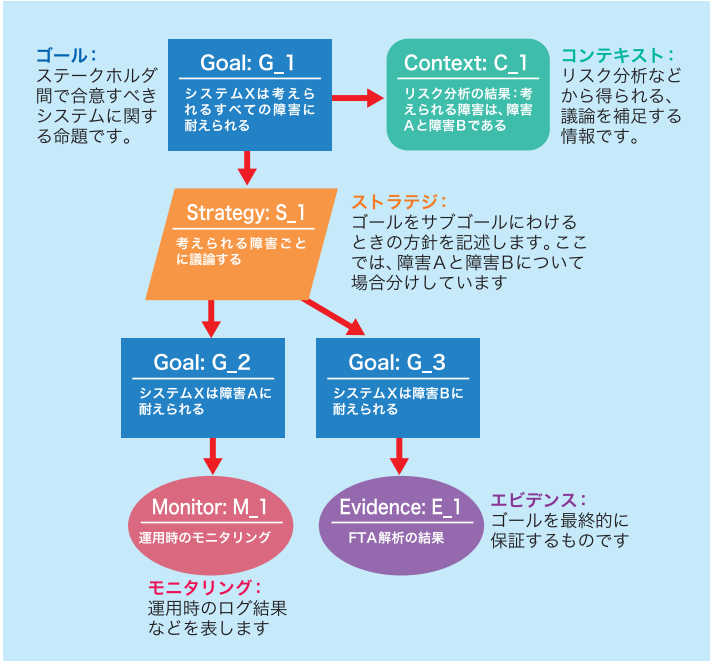
D-Case:変化し続けるシステムのディペンダビリティ合意形成の方法とツール

■D-Case: いつでもどこでもディペンダビリティ

ライフサイクルの各フェーズで、システムのディペンダビリティを合意し、説明責任を果たすための手法とツールとしてD-Caseを研究開発しています



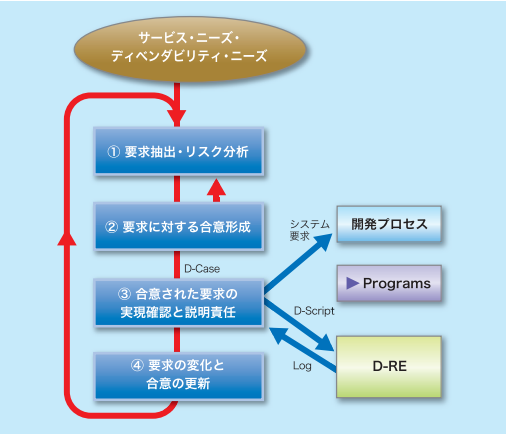
D-Caseは、欧米で高安全性システムの安全性を保証するための手法として普及しているSafety Caseをもとにしています



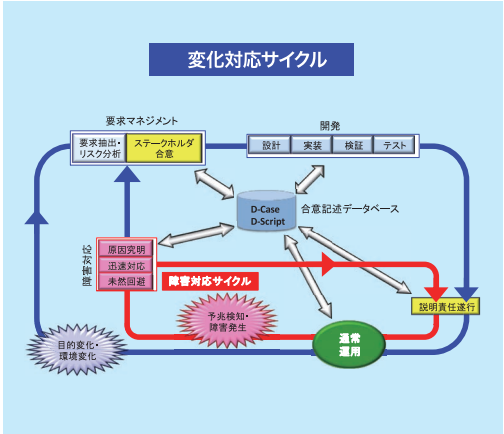
Safety Caseを記述するための主要な記述法であるGSN(Goal Structuring Notation)を、

- 型を導入しデータ化し、他のツールと連携を容易にし
- モニタリングノードにより、運用時のシステムの挙動とのトレーサビリティを保てるよう拡張しています

■D-Caseの役割



D-Caseは、要求抽出・リスク分析より得られるサービス・ディペンダビリティ要求、リスク分析の結果を元に、システムのディペンダビリティを合意し説明責任を果たすために用いられます



D-CaseはDEOSプロセスで合意記述データベースとして用いられ、あらゆる場面で参照・更新されます

国際規格認証支援

ISO26262などのためのD-Caseパターンを作成予定です

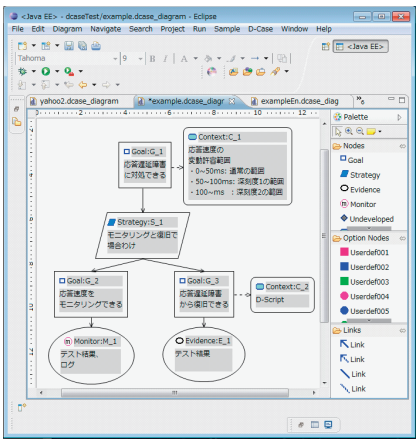
OMGなどでの国際規格標準化

Safety Caseメタモデル標準化などに参加中です

D-Case:変化し続けるシステムのディペンダビリティ合意形成の方法とツール

■D-Caseツール

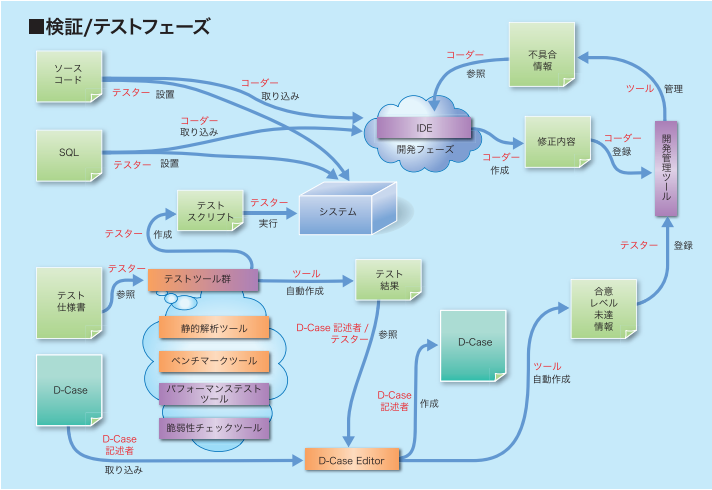
D-Caseの記述支援を行うD-Case Editorをフリーウエアとして公開中です。オープンソース化、商用化などを検討中です。産総研木下チームのD-Case整合性検査のためのD-Case/Agdaツール、東大石川チームと筑波大佐藤チームのDS-Bench/TEST-ENV、慶応大徳田チームのセンサーネットワークシステムなどで応用されています

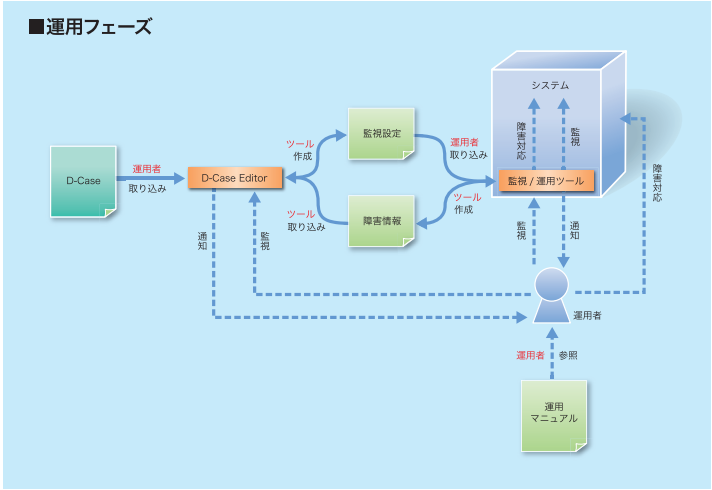


■EclipseプラグインとしてD-Case描画ツールを実装
■D-RE, DS-BenchなどのDEOSツール、Redmineなどとのツールチェーンをプロトタイプ実装
■過去の事例やテンプレートを蓄積し、ネットワーク経由で参照可能なD-Case DBの実装を予定

http://www.il.is.s.u-tokyo.ac.jp/deos/dcase

D-Caseツールは、さまざまな開発、運用ツールと連携して、ディペンダビリティ合意内容のトレーサビリティを常に保ち、説明責任を果たせるよう、ライフサイクルのあらゆる場面で用いられることを目指しています

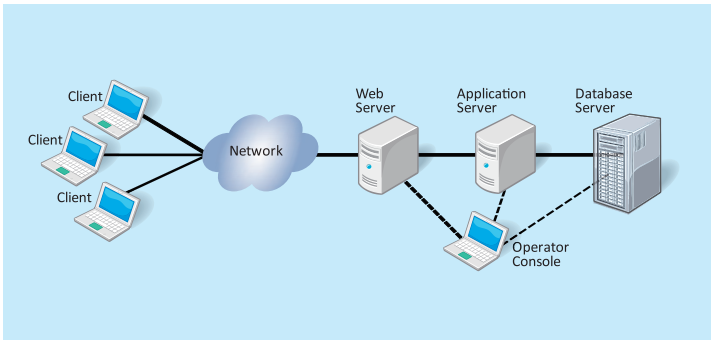




■D-Case記述実験

ウェブサーバデモシステム(DEOSプロセス、DS-Bench/TEST-ENV)でのD-Case記述実験を行なっています

■DEOSプロセスのデモシステム



ETロボコンにおけるD-Case記述実験や、品質保証部門での新商品のD-Case記述実験を行なっています

■意見例

D-Caseの評価	●議論やプロセスの振り返り・チェックをするにつかえそうだ。「この項目は検証の議論ばかりしているが、もっとリスクをmitigateする機能の議論をすべきだ」というようなことがわかるかもしれない ●作業やノウハウをプロセス化するのに使えそうだ ●ゴールとエビデンスの構造で、合意形成の状況や開発・運用の実施状況を把握できるのはよい
改善点要望等	●ツリー表示は大きくなると読みづらくなる ●全容がわかりづらく、じっくり読まないといけない。エンドユーザなど全容を理解していない人が、ここから情報をよみとるのは負担が大きそうだ。全容の分かりやすさや見やすさ、項目の検索性、作りやすさを考慮してほしい

ディペンダビリティテスト支援ツール
DS-Bench / Test-Env

研究代表者：石川　裕（東京大学）
研究代表者：佐藤 三久（筑波大学）
研究分担者：恩田 昌徳（富士ゼロックス）

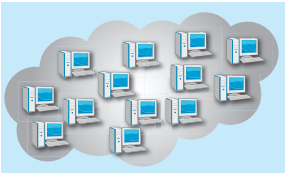
■概　要

ディペンダブルなシステムを開発するためには、あらかじめ、異常状態に対してシステムが要求通りの振る舞いをしているかを定量的に計測し、システムの限界を検証する必要があります。システムの更新時には事前に動作テストを行い、問題がないことを検証する必要もあります。そこで、本プロジェクトではディペンダビリティの計測ツールDS-Bench、およびシステムテストを迅速に行うツールTest-Envを開発し、オープンシステムディペンダビリティの向上に寄与することを目指しています。

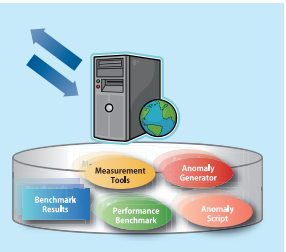
そのためには、ハードウェア異常、ソフトウェアバグ、過負荷、人的ミスなどを系統立てて扱う実行環境が必要です。これらの異常状態の観測やシステムの動作検証のために、テスト工程を自動化し、計算資源を適切に管理することによって、大規模なシステムテストを実現し、多くのテストパターンを用いた複雑なテストを加速します。また、デバイスの仕様がSpecCシステム記述言語で記述されていれば、仮想マシン技術と組み合わせて故障を模擬し、システムのディペンダビリティを評価することができます。

■DS-Bench/Test-Envを使うと...

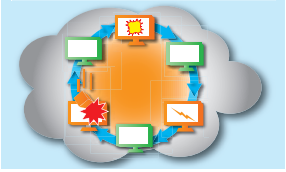
- クラウドによって提供される計算資源で、多数のテスト・ベンチマーク項目を同時に実行し、テストやベンチマークを加速することができます。



- システム設定やテスト・ベンチマーク項目をシナリオとオとして記述することで、複雑なテストやベンチマークの自動化を支援します。
- 既存のソフトウェアや自作プログラムの出力結果を統一的に管理できます。異なる評価環境におけるテスト・ベンチマーク結果を記録したり比較したりすることができます。

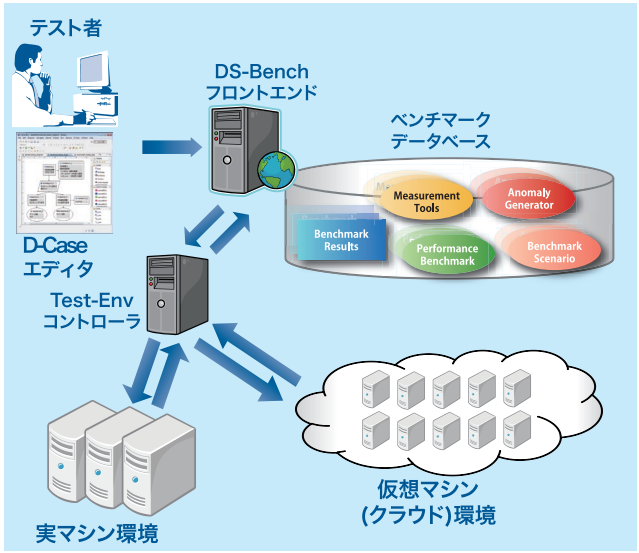


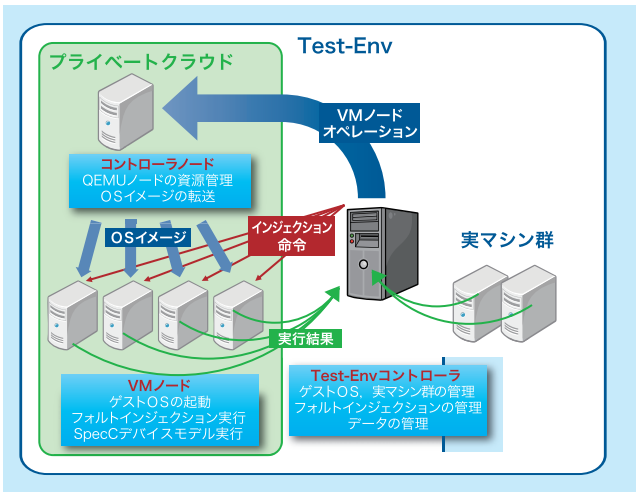
- ハードウェア故障や他のソフトウェアによる異常状態を自在に何度でも模擬できます。



- 独自デバイスでも仕様をSpecCシステム記述言語で記述すると、仮想マシンと組み合わせてシステムを構成してテストを実現することができます。

■DS-Bench/Test-Envの構成

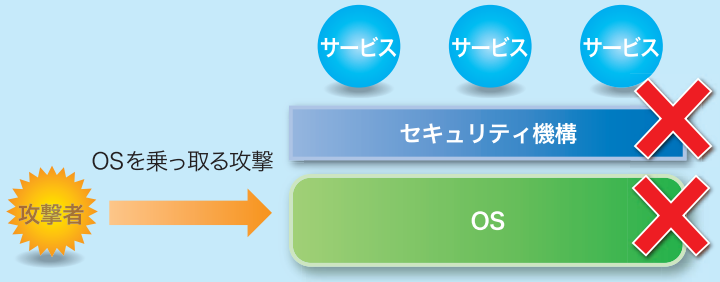




仮想化技術を用いた“進化”するセキュア実行基盤

■OS カーネルを狙った攻撃

不正攻撃手法の進化に伴って、OSカーネル自身を乗っ取る攻撃が猛威を奮っています。OSカーネルを乗っ取られると、OSカーネル上で動作しているセキュリティ機構（認証や認可、アクセス制御）は信頼できません。

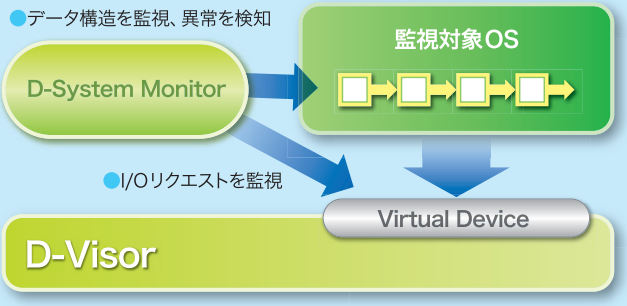


OSカーネルを乗っ取ると、その上で動作するセキュリティ機構の無効化やウィルスのインストールは簡単

サービスの停止や情報漏洩の危険！

■仮想化技術を用いた“進化”するセキュリティ基盤

こうした脅威に対抗するべく、我々のセキュリティ基盤はOSカーネルを監視することで、OSカーネル上で 動作するセキュリティ機構が正しく動作することを保証します。



●D-System Monitor
OSカーネルの振る舞いをして、期待通りに動作しているかを確認する
・データ構造、I/O リクエスト etc.

●実例
FoxyKBD：キーロガー検知機構
RootkitLibra：ファイルメタデータ改竄型ルートキット検知機構
Waseda LMS：カーネルデータ構造の整合性を検査する機構

●D-Visor
D-System Monitorを安全に実行する環境を提供する
・監視対象のOSとD-System Monitorを別VMにすることで、実行環境をisolateする

●実例
SPUMONE：組込みマルチコアシステム向けD-Visor
Art-Linux：ハードリアルタイムサポートD-Visor

◎特徴

✓不正攻撃の進化にも追従できる再帰的なアーキテクチャ
D-System Monitorを無効化する攻撃に対しては、D-System Monitorを監視するD-System Monitorを用意できる

✓“振る舞い”監視によるマルウェアの広範囲な検知
従来のマルウェアの検体１つ１つに対して対策をとる方式とは異なり、振る舞い監視によって１度の対策で様々な種類のマルウェアを検知する

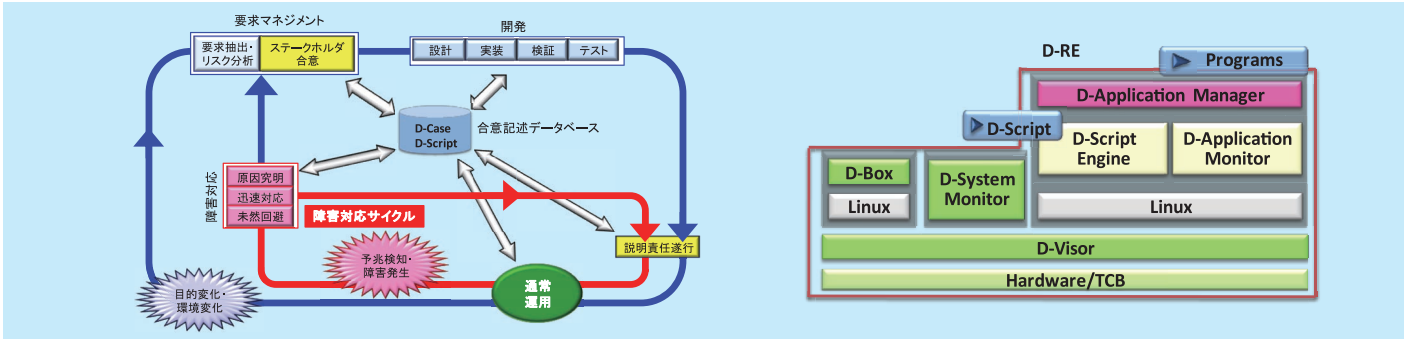
✓D-System Monitor の開発・導入を支援
D-System Monitor Support APIを提供し、新たなD-System Monitorの開発、導入をしやすくする
新たな攻撃への対策を支援することで、攻撃の影響を小さくする

✓モデル検査による D-Visor の検証
D-Visor への新たな機能要求があった際にも、開発段階でD-Visorのコードの品質を向上させる
運用時のバグによる障害を開発段階で可能な限りなくす

仮想化技術を用いた“進化”するセキュア実行基盤デモ

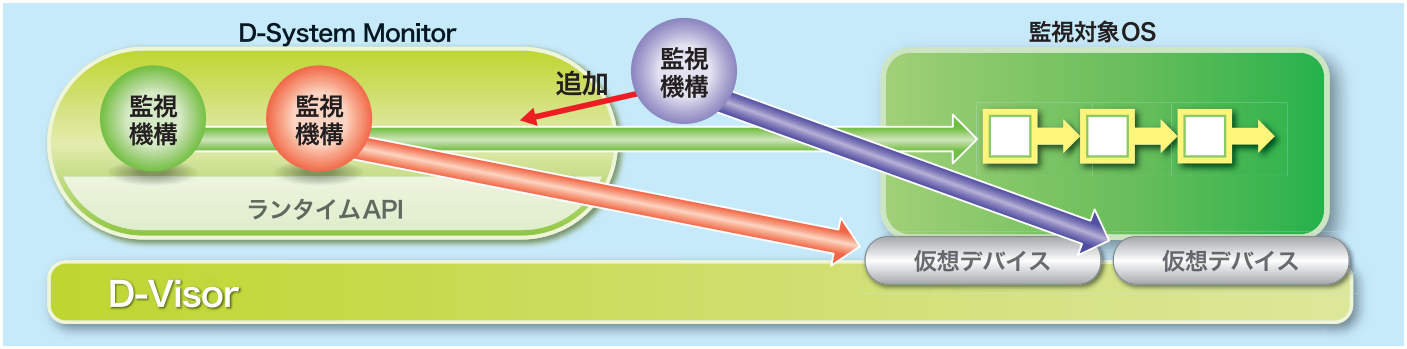
■DEOSプロセス・D-REとの対応

我々のセキュリティ基盤は、DEOSプロセスの障害対応サイクルをサポートします。通常運用時においては予兆検知を行い、障害発生時には原因究明、迅速対応、未然回避などの障害対応を行います。これは、D-REにおけるD-VisorおよびD-System Monitorを用いて実現されます。



■“進化”に対応するOS監視ランタイムAPI

我々のセキュリティ基盤は OS カーネルを監視するにあたり、統合したランタイムAPIを提供することで、新たな脅威に対応する監視機能の迅速な開発を支援します。



■デモ

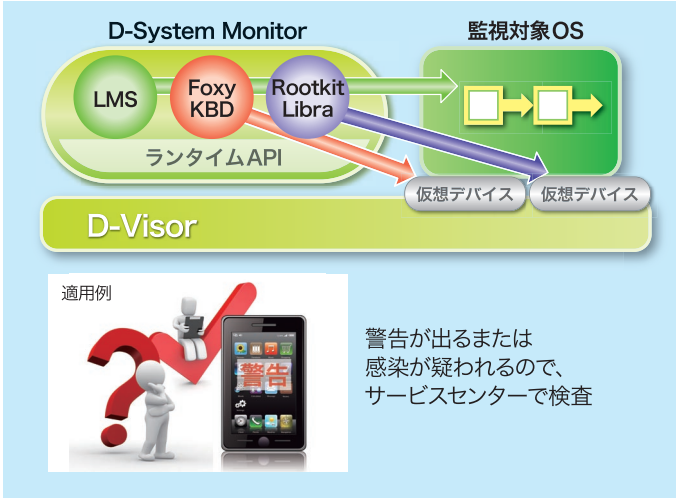
通常運用時における予兆検知

軽量な監視機構により、隠れたマルウェアの実行を監視

一般ユーザによる利用時

障害発生時における原因究明・迅速対応

D-System Monitorに複数の監視機構を追加し原因究明



警告が出るまたは
感染が疑われるので、
サービスセンターで検査

組込みシステム向けマルチコアプロセッサ用 D-Visor：SPUMONE

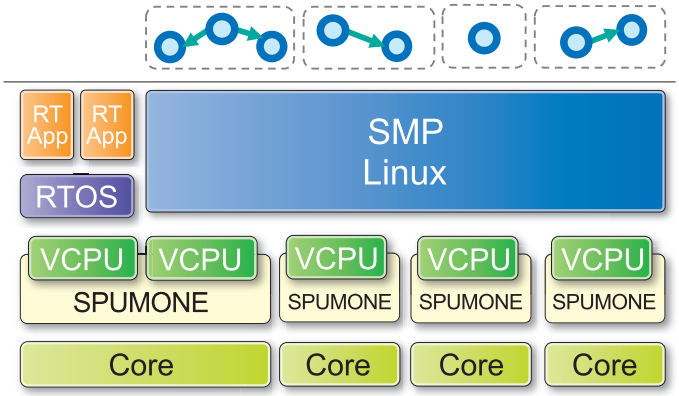
早稲田大学 基幹理工学部 情報理工学科
中島研究室

■組込みシステム向けD-Visorとは?

ディペンダブルな組込みシステムを構築するための仮想化レイヤー(SPUMONE)のことで以下の特徴を持つ。

MSRP1BASE02
RP1 (SH-4A MP ISA)
600MHz x 4

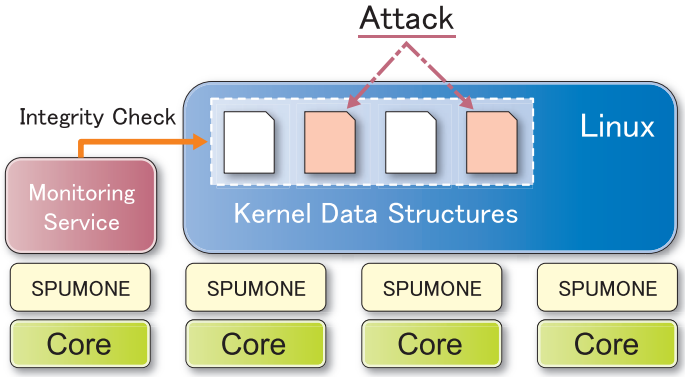
❶ゲストOSのSMP対応
❷低オーバヘッドとゲストOSへの少量の変更量
❸RTOSのリアルタイム性の実現
❹マルチコア環境におけるリスースの有効活用



■D-System Monitor：モニタリングサービス

自動的に生成されたカーネルデータ構造の正常状態を用いて、OSを監視する。

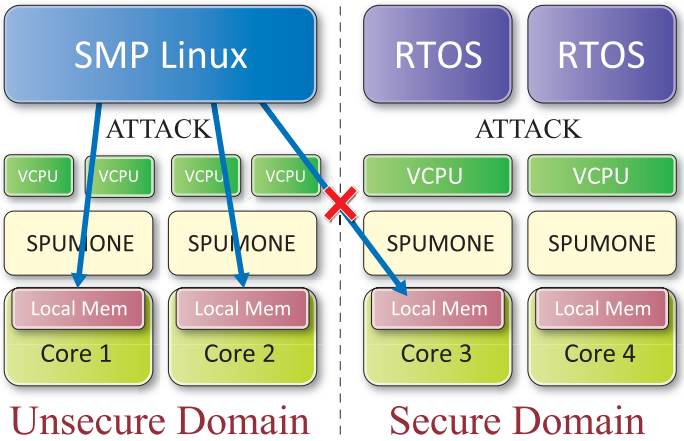
❶対象となるOSを実際に動かし、その時のカーネルデータ構造のデータを取得
❷取得してきたデータを元にカーネルデータ構造の正常状態を生成
❸生成された正常状態を用いたモニタリングサービスを生成する。



■SPUMONE自体の保護

分散モデルを用いたSPUMONEの特徴。

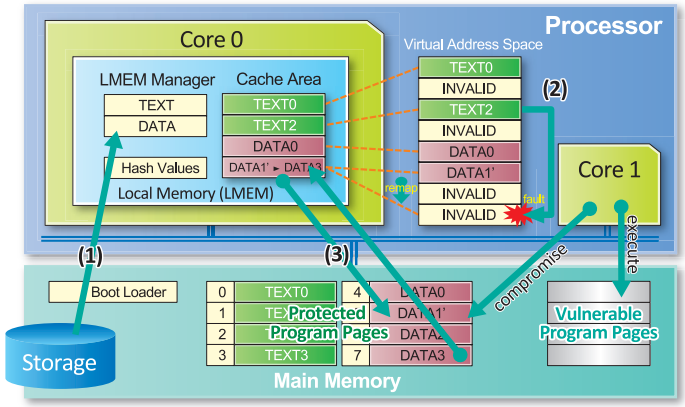
❶各コアのローカルメモリにSPUMONEのインスタンスが存在する
❷他のコアのローカルメモリへアクセスすることは出来ない
❸クロスドメインアタックを防ぐ
❹信頼性やセキュリティの向上



■D-System Monitorの保護

プログラムが保持されているメモリをコアローカルメモリを用いて保護する。

❶ローカルメモリマネージャはコアローカルメモリ内で動作する
❷保護対象のカーネルは仮想アドレススペースの上で動作する
❸ページのスワップの時に、ハッシュ値を計算・比較することで、改変を検知する



連絡先：中島 達夫 e-mail：tatsuo@waseda.jp

OS の振る舞い監視によるセキュリティ実行基盤

■OS カーネルを狙うマルウェア

マルウェアとは悪意のあるプログラムの総称です。
マルウェアの進化に伴って、OS カーネル自身を乗っ取る攻撃が猛威を奮っています。
OS カーネルを乗っ取られると、OS カーネル上で動作しているセキュリティ機構(認証や認可、アクセス制御) は信頼できません。

■既存の対策手法

1. 個々の検体に対して対策をとる

検体のバイナリコードの特徴を記述したシグネチャとマッチングをとる
× 検体の亜種に対して無力
- 亜種は簡単に作れてしまう

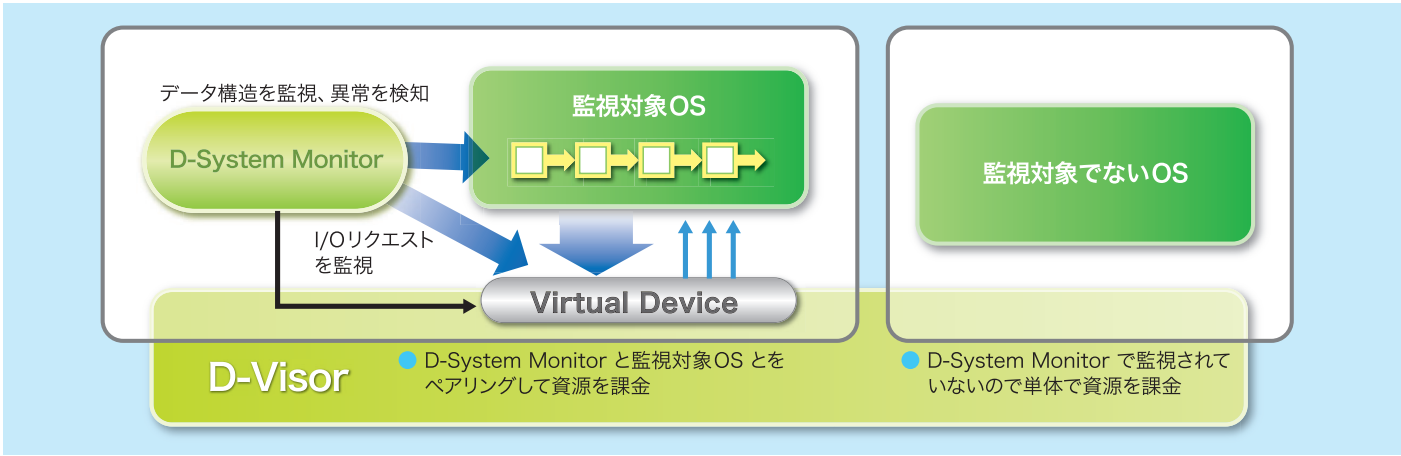
2. 攻撃ベクタごとに対策をとる

特定の攻撃手段に対する対策をとる
- 割り込みベクタの書き換えに対する対策
- プロセステーブルのエントリの改竄に対する対策

× 攻撃ベクタが変わると無力
- 攻撃手段はすぐに新しいものができてしまう

■OS の“振る舞い”監視によるセキュリティ実行基盤

我々のセキュリティ実行基盤は, OS の“振る舞い”に着目し、OS カーネルが健全に動作していることを保証します。また、D-System Monitor と監視対象 OS との関係を考慮することで、柔軟な資源管理を行います。



振る舞い監視機構

- **OS の振る舞いを D-System Monitor で監視する**
  OS カーネルの振る舞いをして、期待通りに動作しているかを確認する
  ・データ構造、I/O リクエスト etc.
- **様々なイベントを OS カーネルに挿入してその反応を監視**
  異常な振る舞いを検知したら、OS が乗っ取られていると判断
  ・イベント: 割り込みの挿入、システムコール列の挿入など
  ・反応: デバイスI/O や制御レジスタへのアクセス等
- **特徴**
  マルウェアの“クラス”ごとに対策をとればよい
  ・キーロガー対策、ルートキット対策など

D-System Monitorの動作を考慮した資源管理機構

- **D-System Monitor と監視対象 OS とをペアにして資源を割り当てる**
  ・D-System Monitor は監視対象OSのためだけに動くため
  ・既存のスケジューラはそれぞれを独立でスケジューリング

例: 監視対象OSでブラウザ、監視対象でない OS で MP3プレイヤーが動作
- ブラウザはインターネットに繋がるため D-System Monitor で監視
- D-System monitor : ブラウザ : MP3 プレイヤー = 25%: 25% : 50% になるべき
  ・ 既存の手法では 33% : 33% : 33% になってしまう

- **特徴**
  理想的な資源割り当てを実現する
  ・上の例だと、25% : 25% : 50 % の資源割り当てを実現

高精度資源管理による
モバイルノードのディペンダビリティ向上

Keio University

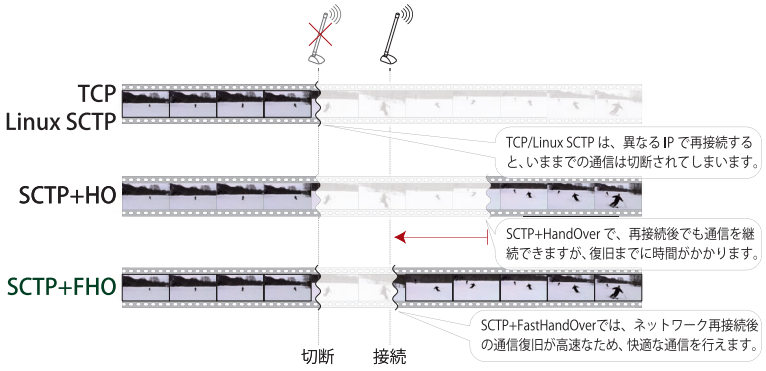
研究代表者：徳田英幸（慶應義塾大学）

■概要

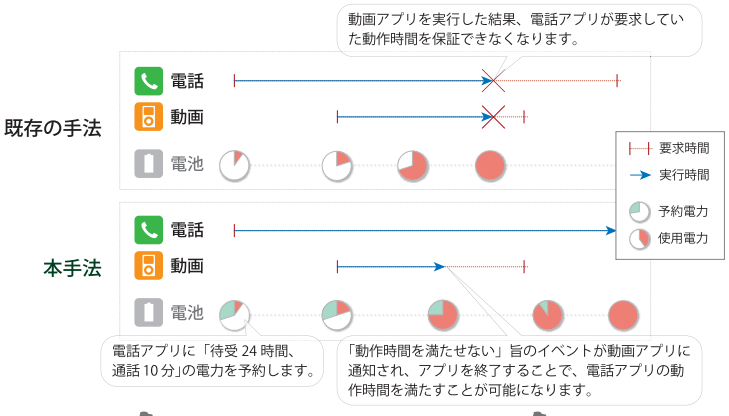
スマートフォンなどの高性能モバイル機器は私たちの生活に深く浸透してきています. これらの機器における問題として, ユーザの移動により通信品質が不安定であること, アプリケーション毎の消費電力の違いによりバッテリ管理が困難なことが挙げられます. 本研究ではモバイル機器上のネットワークおよびバッテリ資源を高精度に監視・制御することで, 予期しない通信の切断やバッテリ切れを防ぎます. それにより, モバイル機器におけるディペンダビリティを向上させます.

■研究内容

通信が切れると…



電池が切れると…

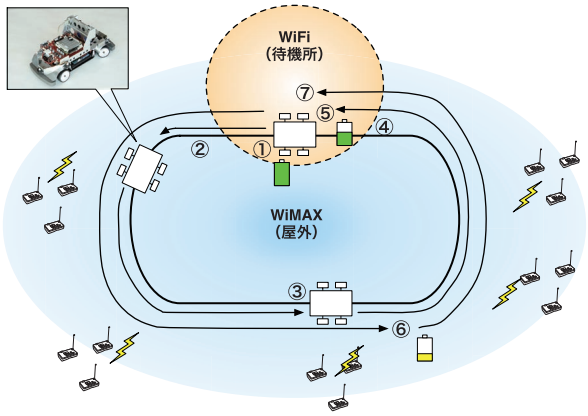


Michio Honda, Jin Nakazawa, Yoshifumi Nishida, Masahiro Kozuka and Hideyuki Tokuda, "A Connectivity-Driven Retransmission Scheme Based On Transport Layer Readdressing", In ICDCS 2011, IEEE International Conference on Distributed Computing Systems (2008)

Mori Masato, Michio Honda, Jin Nakazawa, Hidehyuki Tokuda, "pSurvive: A process lifetime reservation system with fine-grained energy monitoring for multifunctional mobile nodes", In IWCMC 2011, 7th International Wireless Communications and Mobile Computing Conference (2011)

■デモンストレーション

巡回ロボットによるデータ収集

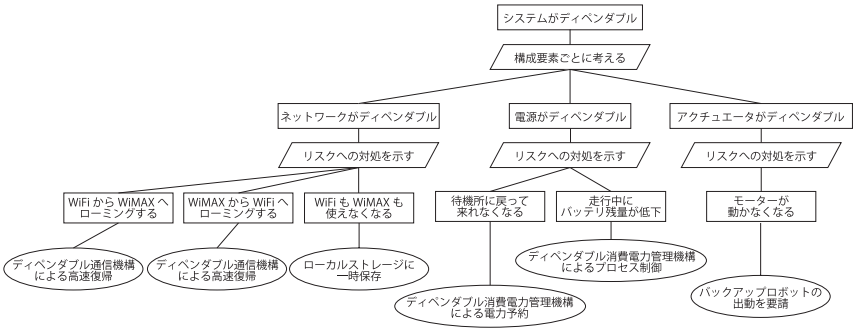


ロボットが敷地内に設置されたセンサからデータを収集する際のディペンダビリティを向上させます.

A. ②や④でWiFiエリアとWiMAXエリアを切り替える際, SCTP+FHOにより通信の切断時間は最小限に留められます. これによりリアルタイム性を要求するサービスが継続して利用可能です.

B. ①や⑤にて待機所から発信する前に移動のための消費電力予約を行い, 戻ってくるまでの電力をあらかじめ確保します. また, ⑥の様に巡回中に電池残量が予想以上に消耗した場合, データ収集のレートを落とし, 無事に待機所に帰還することを優先させます.

D-Case



D-Caseはあらかじめシステムに発生し得る問題を記述し, ゴール, ストラテジ, エビデンスに分けてリスクとそれに対する対処方法を記述できます.
作成したD-Caseはシステムの動作中に継続的に監視を行い, システムのディペンダビリティを確保します.

Jin Nakazawa, Yutaka Matsuno, Hideyuki Tokuda, " Evaluating Degree of Systems' Dependability with Semi-Structured Assurance Case ", In proceedings of 13th European Workshop on Dependable Computing (2011)

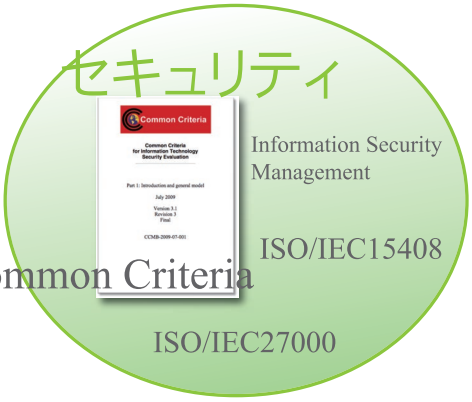
研究代表者: 木下 佳樹（産業技術総合研究所）

オープンシステムディペンダビリティの国際標準化

■ 背景

既に存在する規格の中にもオープンシステムの考え方が明示的でないにせよいくつか表れてきている

IEC61508 や ISO26262 などの機能安全の考え方は、絶対安全がもはや期待できない複雑なシステムに対して、いかにコスト相応な安全性を確保するか、または評価するかを 規定する規格といえる。



さらに、IT サービスマネジメント規格 ISO/IEC20000 やセキュリティマネジメント規格 ISO/IEC27000 は、変化し続けるシステムに対するサービスレベルやセキュリティの継続的な確保のための規格といえる。

	安全性	セキュリティ	ITサービス
Issues	ソフトウェアの把握困難さ、定量的評価の限界	セキュリティ問題の多様化、一律のセキュリティ要件の限界	障害発生の回避困難、発生時の利害関係者間の紛争
コストに応じた対応に関する合意基準	Safety Integrity Level (SIL)	Evaluation Assurance Level (EAL)	Service Level Agreement
納得に基づく合意 / 説明責任の遂行	Safety case	Security Case/ Security Target / Certification Report	(Service report)
プロセスによる対応	Safety culture Safety lifecycle	Information security management	IT service management
関係国際規格	IEC61508, ISO26262	ISO/IEC15408, ISO/IEC27000	ISO/IEC12207/15288, ISO/IEC20000

■ 現在の標準化活動

■ ISO/IEC / JTC1 / SC7 (Software engineering) / WG7 (System/Software lifecycle)

ISO/IEC 15026 System and Software Assurance

- Part1: Consept and vocabulary
- Part2: Assurance case
- Part3: System ity level
- Part4: Assurance in the lifecycle

■ Safety case や Security Case の一般化としてのAssurance case の位置づけ

■ Safety integrity Level やEvaluation Assurance Levelの一般化としてのIntegrity level の位置づけ

2,名が編集委員(editor)として参加

■ IEC TC 56 (DEPENDABILITY)

IEC 60300Dependability Management

IEC 60300-1: Dependability management system
IEC 60300-2: Guidline for dependability management
....

■ Dependability の定義も含む

■ ただし、ハードウェアや機械指向

本プロジェクトから積極的にコメント発信

利用者指向ディペンダビリティの研究

National Institute of Advanced Industrial Science and Technology AIST

■ オープンシステムディペンダビリティ認証規格

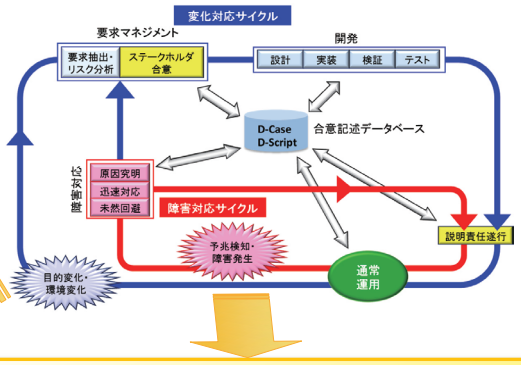
IEC 60300を策定している IEC TC56にて
New work-item proposal の提案を準備中

ねらい

■ Open system 概念の明確化
- Indeterminacy in elements and purpose
- Specification changes over time and stakeholder

■ Dependable open system lifecycle 実現手段の明確化
- Stakeholders' agreement formation
- Achievement of accountability
- Process for service continuity
  - Change accommodation process
  - Failure response process

DEOS Process



DEOS Process の適合性評価へ

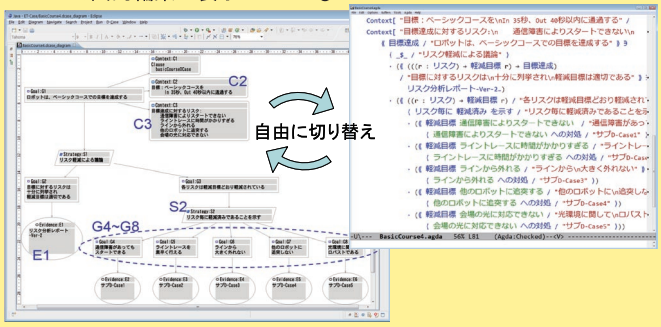
■ 適合性評価技術

■ D-Case/Agda 整合性検証・構築支援ツール

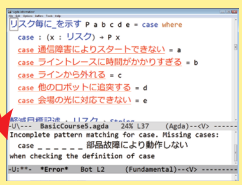
- 変化で生じるD-Case不整合箇所を迅速に検出、対応での変更箇所と全体との整合性を対応実施前に確実に検証
- D-Caseを「証拠データからゴールの証明を作るプログラム」として型検査によって機械的に検証
- プログラミング言語Agdaとその開発環境・証明支援系を使用

D-Case/Agda

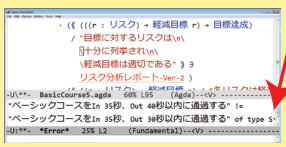
D-Case Editorで図的編集・表示　　Agdaで検証・構築支援・自動生成



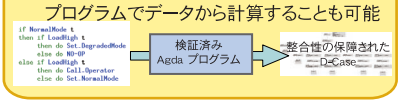
網羅性のチェック例
- C3で列挙したリスクをG4~G8の軽減策が網羅
- 新リスク 部品故障により動作しない が判明、追加してAgdaで検証し直すと...
- S2での分岐に抜けが生じることをAgdaが指摘



見落としがちな不整合のチェック例
- E1のリスク分析レポート-Ver-2は C2の目標 "...40秒以内に通過" に関するもの
- 目標が "30秒" に変更されたのに レポートを更新し忘れると... レポートと新目標との不整合をAgdaが指摘



パターン、自動生成
- D-Caseを直接書き下す代わりに、Agda プログラムでデータから計算することも可能



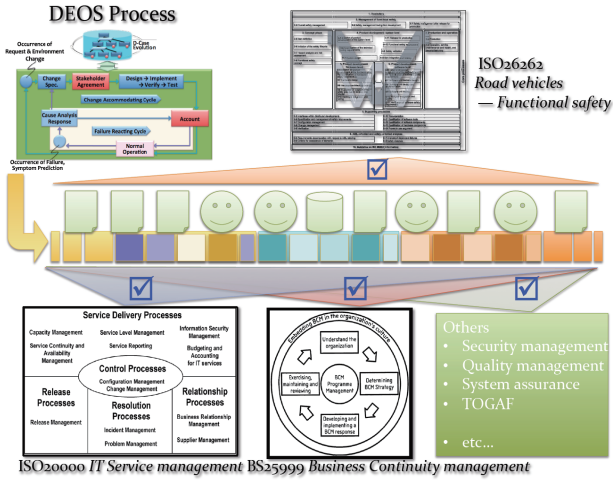
D-Caseをプログラムとしてみることにより、
- プログラミング技法を駆使したD-Case構造化
- ソフトウェア工学のD-Case更新管理への応用

等も可能に。

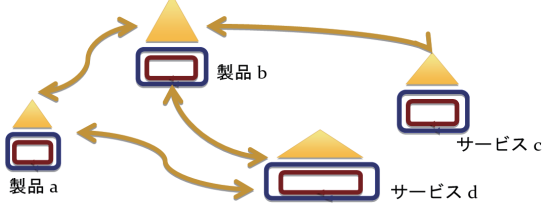
Agdaは、スウェーデンのChalmers大を中心に　開発が進められて来たソフトウェアシステムです。

■ 将来目指す方向

■ オープンシステムディペンダビリティという統一的な視点を与えることによりDEOS Process や D-Case を通じた適合性評価により、様々な分野の標準規格に対応



■ ネットワーク化された製品やサービスに対して、DEOS Process や D-Case のネットワークを通じたオープンシステムディペンダビリティ向上への貢献



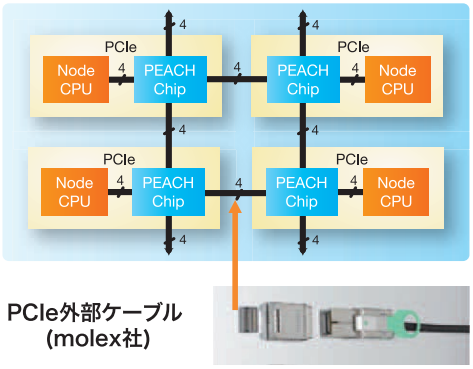
JST　CREST

高性能・省電力でディペンダブルな通信リンク PEARL

研究代表者：佐藤 三久（筑波大学）
研究分担者：朴　泰祐（筑波大学）
研究分担者：有本 和民（ルネサス エレクトロニクス）

■概　要

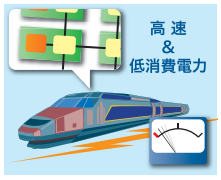
複数コア、複数ノードを用いることにより、冗長性を活かして運用時のディペンダビリティを高めることができます。このとき、ノード間通信にもディペンダブルな通信機構を用意しなければなりません。通信機構のディペンダビリティを高めるためには冗長構成が必須です。一方で、チェックポインティングやロギングなどには高い性能が必要であり、組込みシステムでは省電力も重要になります。そこで、通信リンクとして、PCで標準的に使われているPCI Expressをそのまま用いた、省電力高性能ディペンダブル通信機構 **PEARL** (PCI Express Adaptive and Reliable Link) を提案し、そのコミュニケータチップ **PEACH** (PCI Express Adaptive Communication Hub)を設計開発しています。



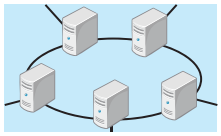
PCIe外部ケーブル
(molex社)

■PEARLを使うと ...

- 近距離の通信を高速かつ低消費電力で実現できます。
- 性能に余裕がある場合には、動作モードを変更して電力を削減することができます。

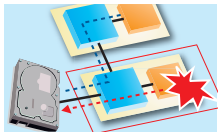


- ノードと PCI Express デバイスとの間を接続し、ネットワーク経由でデバイスをアクセスできます。



- リンクに故障が起こっても、別のリンクを使って通信を継続できます。

- ノードが故障しても、接続されたデバイスのフェイルオーバが可能です。



以下の点で制限があります。
※ノード数は数十程度、各リンク距離は数 m に制限されます。

■PEARLを使って通信するには ...

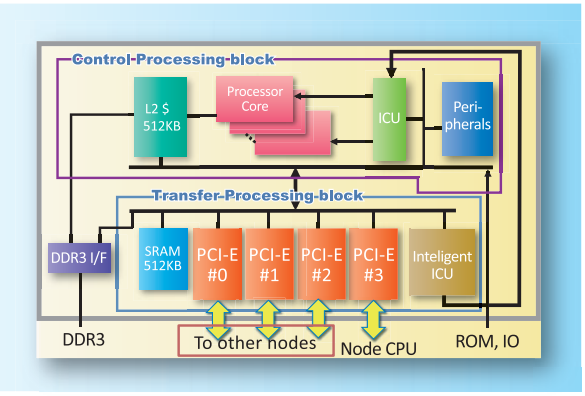
MCAPI (Multicore Communications API) を用いて記述することで、PEARL の性能を最大限に活かした Remote DMA 通信を提供します。

Socket ライブラリによる TCP/IP 通信も提供します。

■PEACHチップ

PEACH チップ仕様

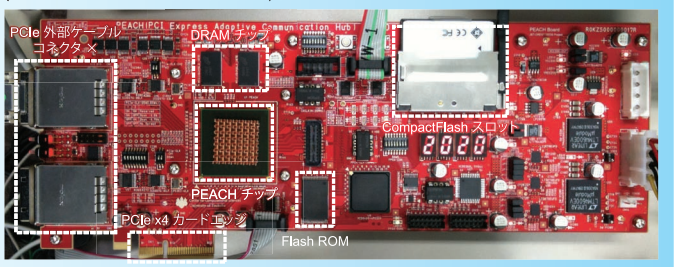
- CPU: M32R (4CPU, SMP 対応)
- PCI Express Revision 2，4 レーン (20Gbps)×4 ポート



■展示・デモ

開発に使用しているプロトタイプ環境の展示，およびプロトタイプ環境を用いてパケットの送信 - 中継 - 受信を行う通信デモを行います.

(PCIe コネクタ 1 個は子基板上に実装)



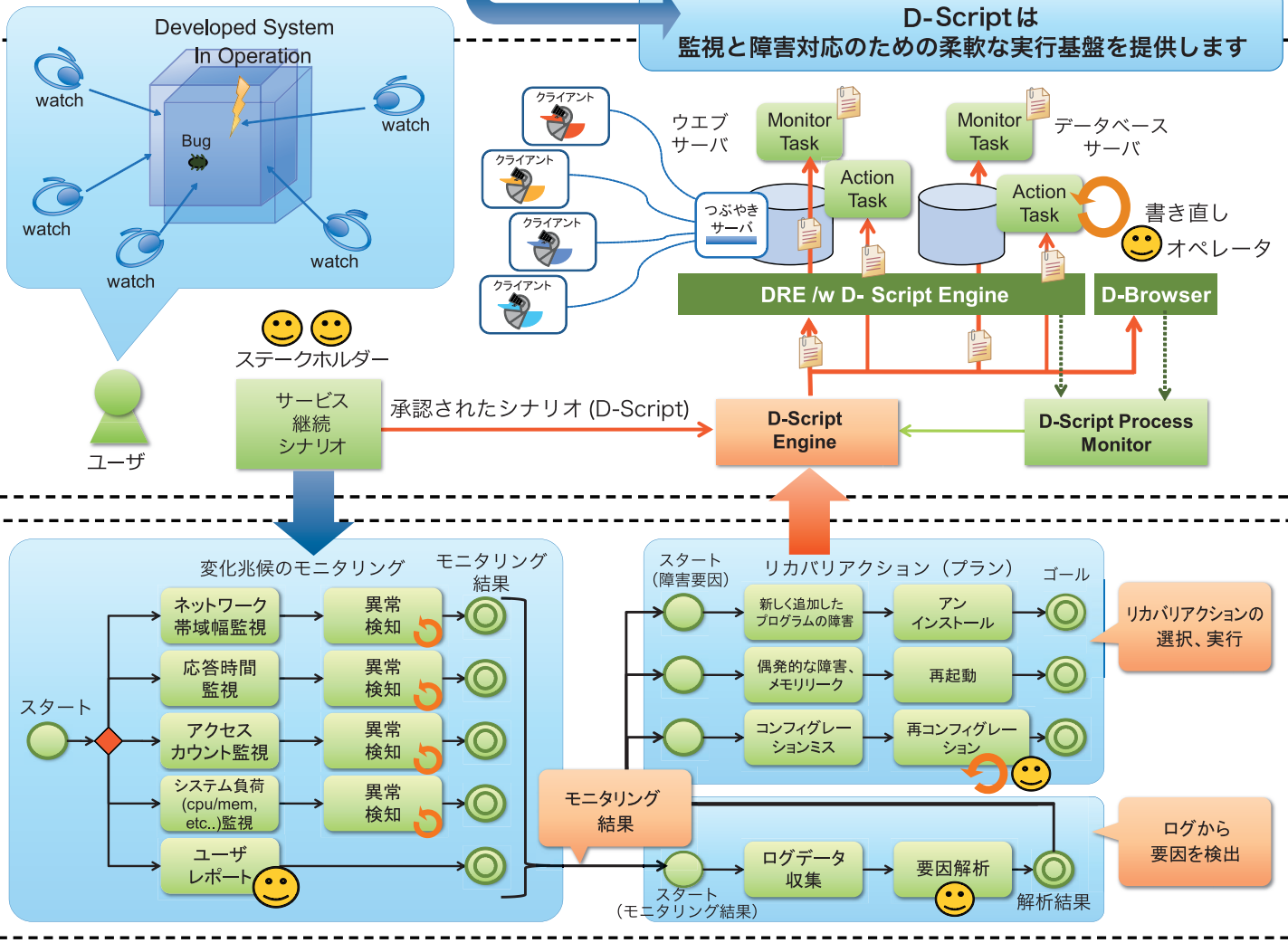
D-Script : Dependable Scripting Language
ディペンダブルスクリプト言語

研究代表者:倉光 君郎(横浜国立大学)

■D-Script：スクリプト言語による柔軟な障害対策実行

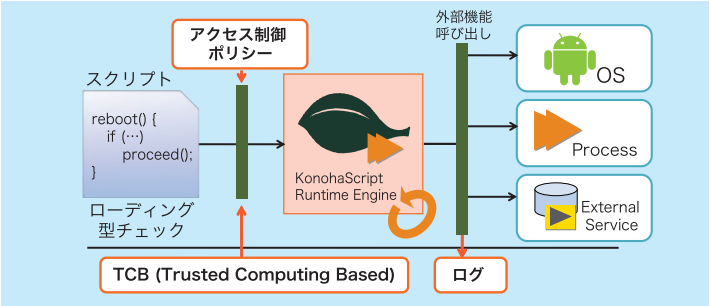
■オープンシステム配下のシステムやコンポーネントは、運用時において、様々な変化が生じます
■システムの運用時の変化を、様々な観点から監視する必要があります
■監視の結果により、最適な障害対応を柔軟に、かつ継続的に行う必要があります

D-Scriptは
監視と障害対応のための柔軟な実行基盤を提供します



■D-Script：スクリプト言語による柔軟な障害対策実行

静的型付けによるオブジェクト指向スクリプト言語:KonohaScript
URL: http://konohascript.org



■自動的な
ロギング

```
ins = new InputStream("/etc/passwd", "r");
```

fopen ()
libc
logging

・外部機能呼び出し
・Software
Accountability

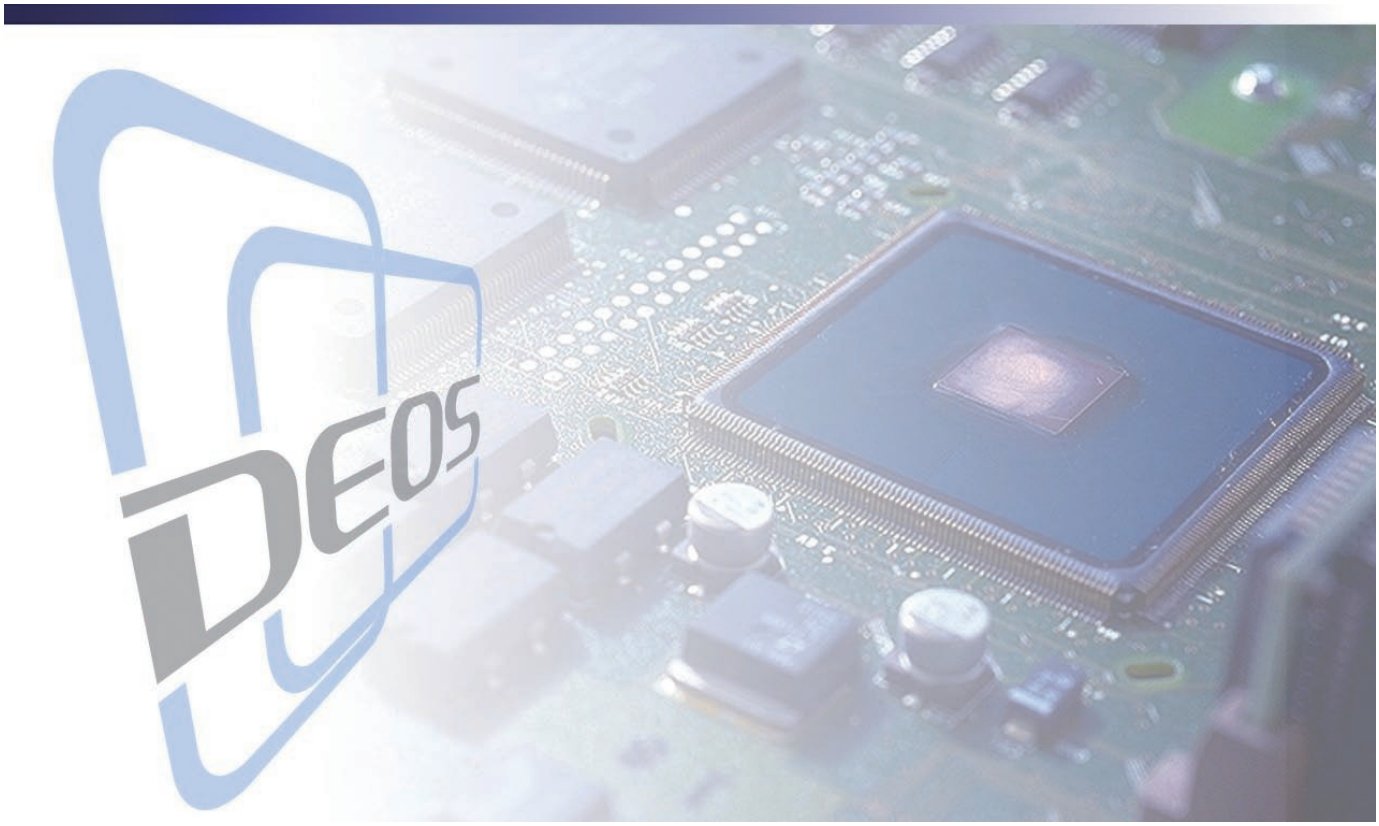
■アクセス制御

```
Class File {
  String read();
  @Restricted boolean write();
}
```

Error
write()

・権限管理(RBAC等)
・型システムベース
・アクセスが許可された
メソッドのみコンパイル
可能

JST
Konoha Project
http://konohascript.org
CREST

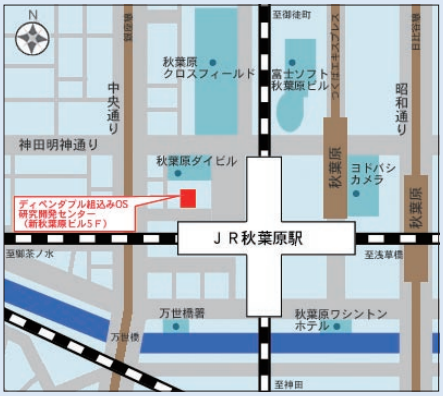


ディペンダブル組込みOS研究開発センター

e-mail center@dependable-os.net

URL http://www.dependable-os.net

DEOSセンター　検索



ディペンダブル組込みOS研究開発センター
＜案内図＞

JR秋葉原駅　電気街口を出てすぐ
（1Fに献血ルームあり）

〒101-0021
東京都千代田区外神田1-18-19
新秋葉原ビル5F
TEL.03-3526-6724
FAX.03-3256-7888