資源の有限性と分散性を意識したレジリエント情報システム

Bounded and Distributed Resource-Conscious Resilient Information Systems

飯島 正 †,
Tadashi IIJIMAudag†
† 慶應義塾大学　理工学部
†Faculty of Science and Technology, Keio Univ.

要旨

近年，レジリエンス [1] という言葉を耳にすることが多い．対人関係の心理学，人間の組織で形成されるシステム，生態系 (ecosystem) などのように，多くの要素が関わってくる中で，想定外の障害や，それに伴う損傷に関して,「回復力を持つ」という言葉で表現するのがしっくりくるようである．

内部の目標 (goal) や要求 (requirement) を自己管理 (self-management) しつつ，外界の変化に自己適応 (self-adaptation) していく，サブシステム（エージェントと呼ぶ）の自己組織的システム (SOS: Self-Organized System) として，レジリエント情報システムを概念的にモデル化する．

1. はじめに

近年，レジリエンス [1] という言葉を耳にすることが多い．対人関係の心理学，人間の組織で形成されるシステム，生態系 (ecosystem) などのように，多くの要素が関わってくる中で，想定外の障害や，それに伴う損傷に関して,「回復力を持つ」という言葉で表現するのがしっくりくるようである．

この概念は，単一の情報システムにとっても，情報システムのシステム（生態系）にとっても重要である．予測不能な想定外の事象に対し，それを回避したり，強靭な耐性を事前に与えることは難しい．耐故障性 (fault-tolerance) [2] や，頑健性 (robustness)，恒常性 (homeostasis) といった概念とも似た側面を持ちながら，決して同じではない．回復力という言葉からして自己修復性 (self-recovery) という概念は近いといえようが，必ずしも，元の状態に戻すということではなく，内部と外部を意識し外部要素の変化に適応する面を備えている．

内部の目標 (goal) や要求 (requirement) を自己管理 (self-management) しつつ，外界の変化に自己適応 (self-adaptation) していく，サブシステム（エージェントと呼ぶ）の自己組織的システム (SOS: Self-Organized System) として，レジリエント情報システムを概念的にモデル化する．自己管理する対象は，自身のもつ目標や要求ばかりではない．自己管理の対象は，自身が保有する資源も含まれ，それが有限であることを意識しつつ，その資源を何らかの最適化基準に基づいて，達成すべき目標に配分する．目標と資源の対によっては，その目標は部分的にしか達成されないということも，あるが，それは達成され維持される必要のなくなった目標が削除されたり，新たに資源が追加割り当てされることによって完全達成されることもある．

外界の変化に適応するということは，外部要因によって，内部の目標や要求を変化させ，更にその変化に追従することでもある．すなわち，目標や要求は，書き換えたり，読み出して分析したり，ということができる，一種のデータとして扱うことができなければならない．

また情報システムのシステム（生態系）を考えた場合，それら分散した情報システムの間で，一部の要求を委託したり，一部の資源を借用したり，といった融通も不可能ではない．それは，システム内部とシステム外部の境界を曖昧にするが，何らかの協調関係の契約の元であれば，生態系全体の存続に有効な手段である（もっとも互恵関係だけを仮定するわけではない）．特定の機能が欠損したり，特定の情報が欠落することがないように安全性・強靭性を意識した場合，分散した外部システムに機能や情報を多重化/冗長化させることも有効な生き残り戦略である．

本報告は，そうしたレジリエンスを備えた情報システムを，レジリエント情報システムと名付け，情報システムのシステム（生態系）の中で，システム境界の内部と外部の状態，自己の目標や要求，自己もしくは周辺に分散している外部システムの有する資源の状態を意識して，第一種オブジェクト (First-Class Object) として扱う情報システムとして，モデル化する[1].

[1]本報告の内容は，[13] の構想をレジリエント情報システムとして，更に発展させたものである．

但し，本報告におけるモデル化は，概念レベルに留め，実装フレームワークの構築は，今後の課題とする．実装フレームワークの中には，資源配分の最適化モジュールも含まれるが，ここでは，どれか一つの手法に限定せず，どのような最適化モジュールを利用するかはオープンとし，状況を自己認識して適切なものに差し替え可能 (plugable) とする．

2. 関連研究

Erik Hollnagel は，レジリエンス・エンジニアリングとして，人の組織を含むシステムにおけるレジリエンスを扱っている [3][4] が，ヒューマンファクターを中心としており，一般的な情報システムアーキテクチャという枠組からは多少離れた位置づけといえる．

耐故障性 (fault-tolerance) の概念は，レジリエンスと似ているが，必ずしも一致するものではない．ソフトウェアシステムの耐故障性は，ソフトウェアパターンとして収集されている [2]．多重化・冗長化は，ハードウェアシステムの太鼓焼成を向上させるために，標準的に用いられる手法であるが，同じくソフトウェアシステムにも用いることができる．本報告で述べる概念モデルにおいても利用される．

自己管理システムや自己適応システムに関する関連研究は数多くある（たとえば [5][6] ）．

システムズ・レジリエンス (`http://systemsresilience.org/index.html`) [7][8][9] [10] では，制約ベースの計算モデルを提案している．

3. レジリエント情報システムの概念モデル

レジリエント情報システムは，外界の変化から引き起こされる，システム境界内部の二つの変化に敏感でなければならない．

(1) ゴール (目標)/要求の変化

(2) 利用可能な (有限の) 資源の変化

本来，外界の状況を観測し，その入力データをもとに以下を実施する二つのモジュールが必要であるが，今回はモデル化の範囲外とする．

(a) 「ゴール (目標)/要求を変化させる」役割の要求変換モジュールと，

(b) 「利用可能な資源量を決定する」役割の資源監視モジュール

本報告では,「ゴール (目標)/要求の変化」が要求モデル（後述のゴール木）に反映され，利用可能な資源が検出できている状態からスタートする．

3.1. エージェント = ゴール＋機能

レジリエント情報システムでは，情報システムのシステム（生態系）の中で，システム境界の内部と外部の状態，自己の目標や要求，自己もしくは周辺に分散している外部システムの有する資源の状態を意識して，第一種オブジェクト (First-Class Object) として扱う情報システムとして，モデル化する．

レジリエント情報システムでは，まず，システム境界の内部にある，自己の目標や要求を第一種オブジェクト (First-Class Object) として扱う．すなわち操作対象（一種のデータ）として取り扱う．目標ないし要求の表現としては，ゴール木に，SysML の要求図で規定されているステレオタイプの一部を取り込む形で要求モデリング言語を検討している．

要求に対応するゴールと，それを満たす「解」としての機能とを，それぞれ自律的な分散要素（エージェント）で表現し，いわゆるマルチエージェントシステムとして，情報システムを柔軟かつ自己組織的に構成するアーキテクチャを考える．これによって，データとして参照できる要求モデルと対応するアーキテクチャを構成する．ゴールエージェントは一つ一つが何らかの機能要求に対応しており，その要求を満たす機能エージェントと対応付けられている．同じ機能要求に対して，必要な資源の量や性能等の非機能的品質が異なる「ゴール&機能」対が複数存在しうる．

有限の資源の間では，何らかの基準に基づいて，資源の配分がなされる．それら「ゴール&機能」対の間では，事前に定義されたエージェント協調プロトコルに基づいてゴールエージェント間で交渉するこ

とにより，資源割り当ての調整を行い，限られた資源の範囲で機能エージェントが要求を満たすように振る舞う．基本的なエージェント間の協調プロトコルとして，よく知られているものに，契約ネットプロトコル (Contract Net Protocol) があるが，こうしたプロトコルを使わずに，複数のエージェント群において，より大域的な最適化（満足化）を行うべく，遺伝的アルゴリズム (Genetic Algorithm) や，粒子群最適化 (Particle Swarm Optimization)，アントコロニー最適化手法など，メタヒューリスティックな最適化手法 [11] を採用し，いわゆるナップザック問題と同じような整数計画問題に帰着させてもよい．

情報システムのシステム（生態系）を考えた場合，機能や情報の委託は，個々の情報システムをマルチエージェントシステムとしてモデル化した場合，それらの間でのエージェントのマイグレーションに相当する．また，機能や情報の多重化や冗長化は，エージェントのクローニングに相当する．

4. ゴール木とマルチエージェントシステム

要求モデルをゴール木（拡張ゴール木）として与える．このゴール木の要素は，必要に応じてデータとして参照したり，追加削除するといった再構成を可能とする．

ゴール木を構成する各ゴールノードには，それを監視するとともに，関連する他のゴールノードと交渉することのできるエージェント（ゴールエージェント）を対応付ける．さらにゴールエージェントには，前節で述べたように，そのゴールを満たす「解」としての機能エージェントを対応付ける．

すなわち，情報システムを，ゴールエージェントと機能エージェントの組で表現することとなり，構成するエージェント群の選択と，相互作用（同期）によって全体の再構成を可能とする．機能エージェントは，さらにサブゴールのゴールエージェントと機能エージェントの組で表現される．機能エージェントは，それ自体がゴールエージェントと機能エージェントの組となっており，再帰的に階層構造を形成する (図 1)．そして情報システム全体は，次節で述べるシステム・ネットで統合される．

ゴールエージェントだけに着目すると，AND/OR 分割によるゴール–サブゴール分解が繰り返されることで木構造（ゴール木）を形成することになる．これによって，その情報システムに要求される機能全体を木構造上に表現することができる．この木構造には，その時点で，その情報システムが満たしている機能も含まれているが，優先順位を加味した選択の結果，候補として挙げられてはいても，その時点で含まれていないような機能も含まれていることがありうる．一般的には，通常，要求分析の結果，取捨選択して捨てられた機能に関しては，システムの設計・実装には含めることはない．しかし，ここでは，そうした余分な要素を持たせておく自体を特徴としている．それにより，あくまで「解」が与えられている範囲にとどまるとはいえ，その範囲内での要求の変化に適応できる可能性が生じる．

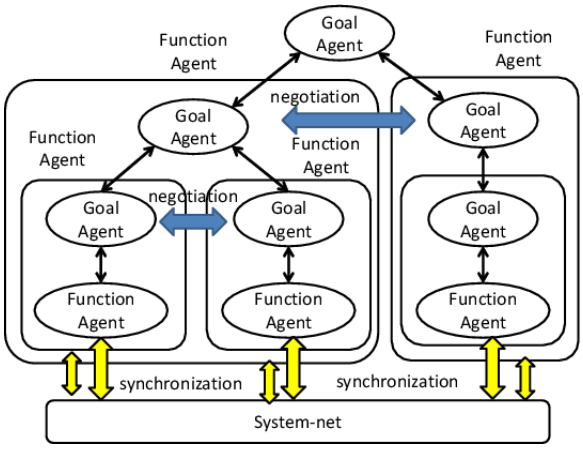

図 1: ゴールエージェントと機能エージェント

ゴールエージェントは，その階層構造に沿って，相互に交渉し，どのゴールを優先して実現するか，どの機能エージェントにどれくらいの資源を割り当てるかを調整する．その交渉に用いる協調プロトコルを共生プロトコル (Symbiotec Protocol) とよぶ．共生プロトコルを構成する基本的な協調プロトコルには，契約ネットプロトコル (CNP; Contract Net Protocol) を用いることもできる．トップゴールから順

に（葉の方向へ），AND 結合しているサブゴール群に向かっては実現を求められている機能を選択するための CFP(Call-for-Proposal) を送信する．一方，OR 結合しているサブゴール群に向かっては，機能は同等なので，そこで必要となる資源について選択するための CFP(Call-for-Proposal) を送信する．いずれのサブゴールも，プロトコルのイニシエータに対して入札を行い，順次，落札しながら，ゴール木を遡っていき，トップゴールに到達するまで繰り返す．落札基準は，交渉プロトコルのイニシエータとなるゴールエージェントに任せられることになる．

落札されたゴールエージェントは，その対応する機能エージェントを活性化させる．その結果，活性化された機能エージェントによって，情報システムが再構成されることになる．各機能エージェントの振る舞いは，オブジェクト指向ペトリネット [12] で表現されたワークフロー（プロセス）として規定することを想定しているが，ここでは詳細は省く．

共生プロトコルを含む協調プロトコルも，オブジェクト指向ペトリネットとして与えることで，エージェント間の協調的な振る舞いを，同期関係によって定義する．

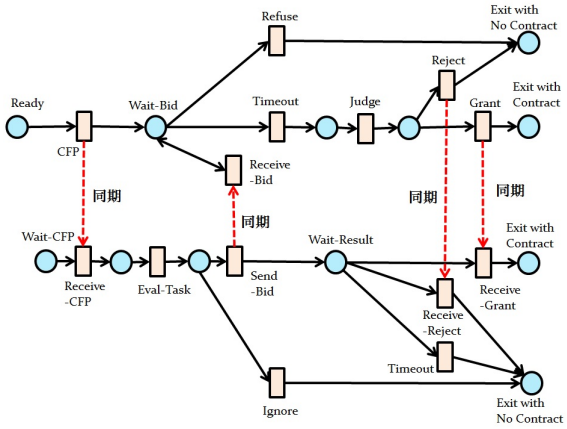

図 2: 契約ネットプロトコルのオブジェクト指向ペトリネットでの表現

情報システムを再構成するトリガーは，要求の変化として与えられる．要求の変化は，ゴール木の上で，関連するゴールエージェントを含む部分木に対して，作用し，その部分ゴール木のルートを起点として，再度，資源割り当てのための共生プロトコルを開始することで自己組織的な再構成を行う．

4.1. 自動的な再構成に伴う問題点と求められる機能

現時点では，第 2 節で示した基本的なシステム再構成機能に加えて，システムの信頼性を維持するためには，以下の 2 つの機能が必要であると認識している．

(1) 検証機能

(2) 説明機能

検証機能は，要求変化に応じた適応的再構成によって想定外の思わぬ不具合を混入させてしまわないようにするためである．また説明機能は，利用者が信頼してそのシステムを使い続けるためには，その時点での機能を理解し納得することがであることから重要である．以下ではそのそれぞれについて検討を加える．

4.1.1. 検証機能

機能エージェントの再構成を自動的に行う場合，想定外の思わぬ不具合を混入させてしまうことが起こってはならない．したがって，再構成したシステムを有効化する前には，毎回，全体的な整合性を検査する検証作業が必要である．必要な検証項目のリストをゴール木に対応付けて常に保守しておき，再構成のたびにモデル検査等の動作検証を行う．検証を通過した場合にのみ，その再構成を有効化するものとする．現時点では，ペトリネットの表現力を有限状態モデルのレベルに限定してモデル検査を適用することを検討中である．

4.1.2. 説明機能

自動的な再構成を行うということから，こうしたシステムでは，そのシステム自身がどのように動作するかについて，利用者へ説明する機能も必要である．ゴール木から有効化された機能エージェントに関連する部分木を抽出し，それを元に，説明を生成する．

5. おわりに

本報告は，レジリエント情報システムの概念を提唱し，情報システムのシステム（生態系）の中で，システム境界の内部と外部の状態，自己の目標や要求，自己もしくは周辺に分散している外部システムの有する資源の状態を意識して，第一種オブジェクト (First-Class Object) として扱う情報システムとして，モデル化した．但し，本報告におけるモデル化は，概念レベルに留め，実装フレームワークの構築は，今後の課題とする．

参考文献

[1] アンドリュー・ゾッリ, アン・マリー・ヒーリー (訳: 須川 綾子): “レジリエンス-復活力-あらゆるシステムの破綻と回復を分けるものは何か,” ダイヤモンド社 , 2013.

[2] Robert Hanmer: “Paterns for Fault Tolerant Software,” Wiley, 2007.

[3] Erik Hollnagel, Nancy Leveson, and David D. Woods (訳: 北村 正晴): “レジリエンスエンジニアリング – 概念と指針 単行本,” 日科技連出版社, 2012.

[4] Erik Hollnagel, David D. Woods, John Wreathall, and Jean Pari‘es (訳: 北村 正晴, 小松原 明哲): “実践レジリエンスエンジニアリング – 社会・技術システムおよび重安全システムへの実装の手引き,” 日科技連出版社, 2014.

[5] Jeremy S. Bradbury, James R. Cordy, Juergen Dingel, and Michel Wermelinger:: “A survey of self-management in dynamic software architecture specifications,” Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems(WOSS ’04), pp.28–33, 2014.

[6] Daniel Sykes, William Heaven, Jeff Magee, and Jeff Kramer: “From goals to components: a combined approach to self-management,” Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems (SEAMS ’08), pp.1–8, 2008.

[7] Kazuhiro Minami, Tenda Okimoto, Tomoya Tanjo, Nicolas Schwind, Hei Chan, Katsumi Inoue and Hiroshi Maruyama: “Formalizing the Resilience of Open Dynamic Systems,” Joint Agent Workshop and Symposium (JAWS), 2012.

[8] 丸山宏: “レジリエントな社会に向けて,” 日本ソフトウェア科学会第 29 回大会, 2012.

[9] 丸山宏, 井上克巳, 椿広計, 明石裕, 岡田仁志, 南和宏: “システムズ・レジリエンス,” 第 11 回情報科学技術フォーラム, 2012.

[10] Kazuhiro Minami and Hiroshi Maruyama. Resilience Engineering: “State of the Art and Research Challenges,” Forum on Information Technology (FIT), Information and Systems Society, 2012.

[11] 古川正志, 他: “メタヒューリスティックスとナチュラルコンピューティング,” コロナ社, 2012.

[12] Rüdiger Valk: “Object Petri nets? Using the nets-within-nets paradigm, ” LNCS 3098, pp.819–848, Spriner, 2004.

[13] 飯島 正: “第一級オブジェクトとしての要求モデル 〜 知識に基づくソフトウェアアーキテクチャアプローチの構想 〜,” 電子情報通信学会，知能ソフトウェア工学研究会, 信学技報, Vol. 114, No. 66, KBSE2014-11, pp. 59-63, 2014.