Dynamic Requirements Specification for Adaptable and Open Service-Oriented Systems

開放的で適応可能なサービス指向システムのための動的要求記述

Ivan J. Jureta, St´ephane Faulkner, Philippe Thiran
Information Management Research Unit, University of Namur, Belgium
iju@info.fundp.ac.be; sfaulkne@fundp.ac.be; pthiran@fundp.ac.be

Abstract.　概要

It is not feasible to engineer requirements for adaptable and open service-oriented systems (AOSS) by specifying stakeholders' expectations in detail during system development.
システム開発において、細部にわたりステークホルダの期待を満たすことによって柔軟で余地のあるサービス指向システムを実現することは、要求工学者にとって難しい（実現可能ではない）
Openness and adaptability allow new services to appear at runtime so that ways in, and degrees to which the initial functional and nonfunctional requirements will be satisfied may vary at runtime.
開放性と適応性は実行時に現れる新しいサービスと、実行時に変化する機能要求と非機能要求が満たすだろう程度を可能にする。
To remain relevant after deployment, the initial requirements specification ought to be continually updated to reflect such variation.
開発後に引き続き関係性をもたせるために、初期の要求記述はそのような変化を考慮するために継続的に最新の状態にすべきである。
Depending on the frequency of updates, this paper separates the requirements engineering (RE) of AOSS onto the RE
（開発は（？））更新の頻度に依存しているので、この研究論文ではAOSSの要求工学を次の要求工学に分割する。
for: individual services (Service RE), service coordination mechanisms (Coordination RE), and quality parameters and constraints guiding service composition (Client RE).
個々のサービス　　：Service RE
サービス連携メカニズム　：Coordination RE
サービス合成を導く制約と品質パラメータ：Client RE
To assist existing RE methodologies in dealing with Client RE, the Dynamic Requirements Adaptation Method (DRAM) is proposed.
Client REに対処するために、従来の要求工学方法論の助けとするため、動的要求適応法（DRAM）は提案されている。
DRAM updates a requirements specification at runtime to reflect change due to adaptability and openness.
動的要求適応法は、適応性と開放性が原因で起こる変化を反映するために実行時に要求仕様書を更新する。

1 Introduction　導入

To specify requirements, the engineer describes the stimuli that the future system may encounter in its operating environment and defines the system's responses according to the stakeholders' expectations. ※刺激と応答は一対で考えると良い。
要求を明確に述べるために、技術者は未来のシステムが未来の動作環境で直面するであろう外部からの信号を表現し、システムの応答をステークホルダの期待に従って定義する。

The more potential stimuli she anticipates and accounts for, the less likely a discrepancy between the expected and observed behavior and quality of the system.
潜在的要因（これから起こる新しい刺激）をシステムは、期待された動作と実際の動作とシステムの品質の間の矛盾は少ないと予測し、説明する。

Ensuring that the requirements specification is complete (e.g., [17]) becomes increasingly difficult as systems continue to gain in complexity and/or operate in changing conditions (e.g., [15, 10]).
システムが複雑さを増し、システムが変化する状況で動作するので、
要求仕様書が完全であることを保障することは、ますます困難になる。

Adaptable and open service-oriented systems (AOSS) are one relevant response to such complexity.
柔軟で余地のあるサービス指向システムは､そのように複雑なシステムに対する適切な返答の一つである。

They are open to permit a large pool of distinct and competing services orignating from various service providers to participate.
共有するために、さまざまなサービスプロバイダによって生じる、まったく異なり、競合したサービスの巨大なプールを可能にするために、それら(AOSS)は開放的である。

AOSS are adaptable—i.e., an AOSS coordinates service provision by dynamically selecting the participating services according to multiple quality criteria, so that the users continually receive optimal results (e.g., [7, 8]).
開放的で適応可能なサービス指向システムは適応可能である。
言い換えれば、AOSSは、ユーザが継続的に最善の結果を受け取れるように、多様な品質基準に従って共有するサービスを動的に選択することによって、サービス供給を連携させる。

A complete requirements specification for an AOSS would include the description of all relevant properties of the system's operating environment, and of all alternative system and environment behaviors.
AOSSのための完全な要求仕様書には、システムの動作環境の性質に関連性のあるすべての記述、そして代替系システムと環境問題のすべての記述が含まれる。

All services that may participate would thus be entirely known at development time.
共有し得る全てのサービスは、上で述べられたように、完全に開発段階で知られている。
（＝すでに存在するサービスを対象に開発する）

Following any established RE methodology (e.g., KAOS [4], Tropos [3]), such a specification would be constructed by moving from abstract stakeholder expectations towards a detailed

specification of the entire system's behavior.
実証された要求工学方法論を受けて、そのような仕様書(AOSSのための完全な要求仕様書)は、抽象ステークホルダの要求から詳細化された完全なシステムのふるまいの仕様書へと変化することによって構築される。
As we explain in Section 2, [applying such an approach and arriving at the extensive specification of an AOSS] is not feasible.
我々が第２章で説明するように、そのようなアプローチを適合することと、AOSSの詳細な仕様書に到達することは、実現不可能である。
In response, this paper introduces concepts and techniques needed to
それに応じて、この研究論文では、次にあげる必要とされる概念と技術を導入する。
(1) determine how extensive the initial specification ought to be and what parts thereof are to be updated at runtime to reflect system adaptation, and (2) know how to perform such updates.
第１に、システムの適応性に影響するため、初期の仕様書があるべき状態を判断し、どの部分が実行時に更新されるべきかを決める必要がある。
第２に、そのような更新を行う方法を知っている必要がある。
The specification can then be used to continually survey and validate system behavior.
仕様書は継続的にシステムのふるまいを検証することに利用することができる。
To enable (1), this paper separates the requirements engineering (RE) of AOSS depending on the frequency at which the requirements are to be updated (§2): RE executed for individual services or small sets of services (Service RE), RE of mechanisms for coordinating the interaction between services (Coordination RE), and RE of parameters guiding the runtime operation of the coordination mechanisms (Client RE).
１を可能にするために、この研究論文では、要求が更新されるべき箇所に頻繁に依存するAOSSの要求工学を分割する（第２章）。
■Service RE　　：要求工学は個々のサービスやサービスの小さなかたまりを実行する。
■Coordination RE：サービス間の相互作用を連携するための構造から成る要求工学
■Client RE　　：連携メカニズムの実行時作用を導くパラメータから成る要求工学
To address (2), this paper focuses on Client RE and introduces a method, called Dynamic Requirements Adaptation Method (DRAM) for performing Client RE for AOSS (§3).
２に対処するために、本研究論文ではClient REに焦点をあて、そしてAOSSのためのClient REを実行するために、動的要求適応法（DRAM）と呼ばれる手法を取り入れる。
We close the paper with a discussion of related work (§4), and conclusions and indications for future effort (§5).
我々は関連研究の考察をし、今後の取り組みのための結論と指摘を示し、研究論文を締めくくっている。

Motivation.　研究の動機

The proposal outlined in the remainder resulted from the difficulties encountered in engineering requirements for an experimental AOSS, call it TravelWeb, which allows users to search for and book flights, trains, hotels, rental cars, or any combination thereof.

この提案は、実験的なAOSSのための要求定義工学で経験する難しさによって生じる残りの部分を参照した。
それはTravelWebと呼ばれる、ユーザに電車やホテルやレンタカーを検索し予約をすること、またはそれらに関するあらゆる連携を許すものである。
Services which perform search and booking originate from the various service providers that either represent the various airlines and other companies, so that TravelWeb aggregates and provides an interface to the user when moving through the offerings of the various providers.
検索や予約をするサービスは、
TravelWebはいくつかのプロバイダから提供されるものを通して行動している時のユーザの接点の情報を収集でき、さらに提供できるように、
検索と予約のどちらも代理するいくつかの航空会社や、その他の会社といった、いくつかのサービスプロバイダが起源である。
Each provider can decide what options to offer to the user: e.g., in addition to the basics, such as booking a seat on an airplane, some airlines may ask for seating, entertainment, and food preferences, while others may further personalize the offering through additional options.
それぞれのプロバイダはどんなオプションをユーザに提供するのかを決めることができる。
例えば、飛行機や座席の予約のようなものが挙げられる。
基本的なものに加えて、航空会社は座席やもてなしや食の好みを求めてくるだろう。
他の航空会社は追加のオプションを通し、提供するものをより一層個人向けにするだろう。
We have studied elsewhere [7, 8] the appropriate architecture and service composition algorithms for TravelWeb.
我々はTravelWebのための適切なアーキテクチャとサービス合成アルゴリズムを他のどこかで学ぶべきだ。
Here, we focus on the engineering of requirements for such systems.
ここに、我々はそのようなシステムのための要求工学における焦点を置く。

2 Service, Coordination, and Client RE

To engineer the requirements for TravelWeb, a common RE methodology such as Tropos [3] would start with early and late requirements analyses to better understand the organizational setting, where dependencies between the service providers, TravelWeb, and end users would be identified, along with the goals, resources, and tasks of these various parties.
TravelWebのための要求工学を立案するために、Troposのような共通の要求工学の方法論は、サービスプロバイダとTravelWebとエンドユーザ間の依存関係が様々なパートナーの仕事や目標や資質とともに認識されるだろう組織環境をよりよく理解するために、朝早くから夜遅くまで要求分析を始めるだろう。

※Tropos：i* を要求工学分析工程として含むソフトウェア開発方法論
→アクタ間の依存関係を分析する

Architectural design would ensue to define the sub-systems and their interconnections in terms of data, control, and other dependencies.
構造化設計はサブシステムと、データや制御や依存関係の観点から相互連結を明確にすることで結果として起こるだろう。

Finally, detailed design would result in an extensive behavioral specification of all system components.
最後に、詳細設計は全てのシステムの構成要素の、広範囲に及ぶ挙動詳細をもたらすだろう。
While other methodologies, such as KAOS [4] involve a somewhat different approach, all move from highlevel requirements into detailed behavioral specifications. ※all ⇔ other の関係
他の方法論（KAOS法のようなものは多少異なったアプローチをしている）を含む、全ての方法論は、高水準の要求から挙動詳細仕様書まで進展させている。
The discussion below, however, concludes that such an approach is not satisfactory, because:
しかしながら、以下の議論はそのようなアプローチでは満足できない結論を出している。

※↓Service REに関する内容（話の流れから推測して）

1) TravelWeb is open.

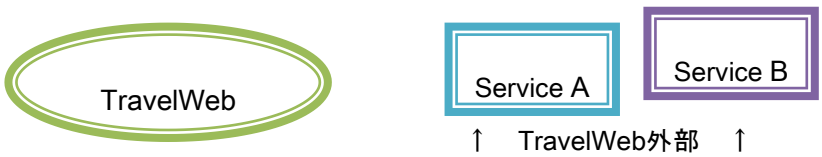
TravelWebは公開されています。

Various hotels/airlines/rental companies may wish to offer or retract their services.
さまざまなホテル、航空会社、レンタル会社は自身のサービスを申し出るか、取り消すことを望むだろう。
Characteristics of services that may participate in TravelWeb at runtime is thus unknown at TravelWeb development time.
実行時にTravelWebに関与するだろうサービスの特徴は、上で述べたようにTravelWebの開発段階では知られていない。

※TravelWebに関与するサービスのふるまいは、TravelWebで実行するまでわからない。
提供されるサービス＝ブラックボックスと考えられる。

＝　サービスの特徴は開発段階では知ることができない。
∵　個々のサービスは、TravelWebとは別の場所で開発されるから。

Individual services are likely to be developed outside the TravelWeb development team, before or during the operation of TravelWeb.
個々のサービスはTravelWebの運用の前後に、TravelWebの開発チームの外部で開発されたようだ。



※私が考えるに、TravelWebはWebサービスみたいなもの

It is thus impossible to proceed as described for the entire TravelWeb—instead, it is more

realistic to apply an established RE methodology locally for each individual service, and separately for the entire TravelWeb system, taking individual services as black boxes of functionality (i.e., not knowing their internal architecture, detailed design, etc.).
上で述べたように、TravelWeb全体のサービスを呼び出す関数などを表現することは不可能である。（ ＝　書いたコードの意図通りにTravelWebが実行することは不可能）
その代わり、機能的なブラックボックスとして個々のサービスは解釈し、
それぞれ個々のサービスのために局所的に、そして完全なTravelWebのシステムのために独立して、実証された要求工学方法論を利用することのほうが、もっと現実的である。

※↓Coordination REに関する内容（話の流れから推測して）

2) Resources are distributed and the system adapts.
資源は分配され、システムは状況に合わせて変化する。
All services may or may not be available at all times.
全てのサービスは常に利用できるかどうかは分からない。
Moreover, individual services are often not sufficient for satisfying user requests—that is, several services from distinct providers may need to interact to provide the user with appropriate feedback.
さらに、個々のサービスはユーザの要求を満足するにはしばしば十分ではないかもしれない。
異なったプロバイダからのいくつかのサービスは相互作用するために、ユーザに適切な意見を与えることを必要とするだろう。
Adaptability in the case of TravelWeb amounts to changing service compositions according to service availability, a set of quality parameters, and constraints on service inputs and outputs (see, [8] for details).
TravelWebの場合の柔軟性は、品質パラメータの一連、サービスの入出力に対する制約といった、サービスの可用性次第でサービス合成を変化することを意味する。(詳細は参考文献の８を見よ)
RE specific to the coordination of services carries distinct concerns from the RE of individual services.
サービス調整のための要求仕様書は、個々のサービスの要求工学から異なった関心事を支援する。

※↓Client REに関する内容（話の流れから推測して）

3) Quality parameters vary.
品質パラメータは変化する。
　　→動作条件を与えるための情報　⇒　品質パラメータ　＝　提供するサービスの内容？
Quality (i.e., nonfunctional) parameters are used by the service composer as criteria for comparing alternative services and service compositions.
品質(言い換えれば、非機能)パラメータは他のサービスとサービスの構成要素を比較するための基準としてサービス構築者によって使われる。
Quality parameters are not all known at TravelWeb development time, for different services can be advertised with different sets of quality parameters.
異なったサービスは異なった一連の品質パラメータとして売ることができるため、品質パラメー

タはTravelWebの開発時間において全く知られていない。
As the sets of quality parameters to account for in composing services change, (a) different sets of stakeholders' nonfunctional expectations will be concerned by various service compositions and (b) there may be quality parameters that do not have corresponding expectations in the initial specification.
サービスの変化を構成することを説明するための一連の品質パラメータであるので、

（a） 異なった一連のステークホルダの非機能要求はさまざまなサービスの構成要素によって懸念されるだろう。

（b） それらは最初の要求仕様書と一致する要求である必要がないような品質パラメータとなるだろう。

Observation (a) entails that initial desired levels of expectations may not be achieved at all times, making the initial specification idealistic.
(a)の見解は、常に最初の要求の望むレベルが最初の理想主義的な仕様書の作成段階で達成されない可能性を引き起こす。
Deidealizing requirements has been dealt through a probabilistic approach by Letier and van Lamsweerde [11] where requirements are combined with probability of satisfaction estimates.
(理想的でない＝de+idealizing ?)
現実的な要求はLetierとvan Lamsweerdeによる確率論的アプローチの至る所で扱われた。
その要求は満足のいく評価の確率を組み合わせたものである。
In an adaptable system, the probability values are expected to change favorably over time (see, e.g., our experiments on service composition algorithms for AOSS [7, 8]), so that updating the initial requirements specification to reflect the changes seems appropriate if the specification is to remain relevant after system deployment.
柔軟なシステムでは、もし仕様書はシステム開発後も関連性を保つとしたら、それを適すような変化を反映するための最初の要求仕様書は更新できるように、
確率変数（期待値？）が時間とともに順調に変化することを期待されている。
(参照、例：AOSSに対する我々のサービス合成アルゴリズムにおける経験)
Observation (b) relates to the difficulty in translating stakeholders' goals into a specification: as March observed in a noted paper [12], both individual and organizational goals (which translate into requirements) tend to suffer from problems of relevance, priority, clarity, coherence, and stability over time, all of which relate to the variability, inconsistency, and imprecision, among other, of stakeholder preferences.
(b)の見解は、ステークホルダの目標を仕様書に変えるという問題に関連している。
Marchのような有名な論文に見られる、個々や組織の両方のゴール(要求となる)は、
時間とともに、妥当性や優先度や鮮明さ(無曖昧性？) や一貫性や不変性や、可変性に関係するすべてのもの（ステークホルダの好みによるいくつかある矛盾や不正確さ）に悩まされる傾向がある。

Instead of assuming that the initial set of expectations is complete, the specification can be updated at runtime to reflect new system behaviors and to enable the stakeholders to modify requirements as they learn about the system's abilities and about their own expectations.
最初の期待が完全であると仮定する代わりに、新しいシステムのふるまいを反映することで、実行時に仕様書を更新することができる。
また、ステークホルダが、彼ら自身の期待とシステムの能力について学ぶにつれて、要求を修正できるようにすることで、実行時に仕様書を更新することができる。
Having established that updates are needed, we turn to the question of what to update.
更新する必要があることを確立し、我々は更新するものの問題のことを考える。
A requirements specification for an AOSS involves requirements that are of different variability over time.
AOSSのための要求仕様書は、時間とともに、異なる可変性のある要求に影響を与える。
Our experience with AOSS [7, 8] indicates that a particular combination of service-oriented architecture and service coordination algorithm enables adaptability, whereby the architecture and the algorithm act as a cadre in which various requirements can be specified.
AOSSにおける我々の経験は、サービス指向アーキテクチャの独特の連携とサービス合成アルゴリズムが柔軟性を可能にすることを示唆する。
そのために、アーキテクチャとアルゴリズムは、さまざまな要求を満足することができる構造として振る舞う。
Since adaptability does not require change in the architecture and algorithm, requirements on these two remain reasonably stable.
柔軟性はアーキテクチャやアルゴリズムの変化を必要としないので、それら２つの要求は無理なく安定したままで残る。

Service RE

This observation, along with the localization of service-specific RE to each individual service or small service groups leads to a separation of AOSS RE effort as follows:
この見解は、サービス仕様要求工学の位置を特定することとともに、それぞれ個々のサービス、もしくは小さなサービスの集まりは次のようなAOSSの要求工学の成果をもたらした。
Service RE involves the engineering of requirements for an individual service, or a set of strongly related services (e.g., those obtained by modularization of a complex service).
サービス要求工学は個々のサービス、または強く関係した一連のサービス（例：複雑なサービスのモジュール化によって得られるもの）に対する要求工学を伴う。
Depending on whether the service itself is adaptable, a classical RE methodology such as Tropos or KAOS can be applied.
サービス自身が柔軟であるかどうかにもよるが、TroposやKAOS法のように有名な(標準的な)要求工学方法論は応用することができる。
As the coordination mechanism selects individual services for fulfiling user requests, requirements on an individual service do not change with changes in service requests (inputs

and/or outputs and constraints on these and quality parameters change with variation in requests).
調整メカニズムはユーザの要求を満たすための個々のサービスを選択するので、
個々のサービスの要求はサービス要求の変化(要求のばらつきとともに変化する、入力および出力、さらにそれらの制約と品質パラメータ)とともに変化しない。

Coordination RE

Coordination RE takes services as self-contained functionality and focuses on the requirements for the coordination of services.
Coordination REはサービスを必要なものを完備した機能性と考え、サービスの調整のための要求に焦点をあてる。
In an AOSS, this typically involves the definition of the architecture to enable openness, service interaction, service selection, and service composition for providing more elaborate, composite services to fulfil user requests.
AOSSでは、一般的にはこれは、ユーザの要求を満たすためにもっと入念で各種の要素から成るサービスを提供するために、開放性やサービス相互作用、サービス選択、サービス合成を可能にするためのアーキテクチャの定義を伴う。
As noted above, these requirements vary less frequently than those elicited as a result of Client RE.
上で述べたように、それらの要求は、それらはClient REの結果のように誘発されたのに比べると、それほど頻繁ではないが変化する。

Client RE

Client RE assumes a coordination mechanism is defined and is guided by constraints to obey, and quality parameters to optimize (e.g., QoS, execution time, service reputation).
Client REが想定する調整メカニズムは、最適化するための品質パラメータ(例 : QoSや実行時間、サービスの品質)と従わなければならない制約によって導かれ、定義される。
This is the case after a service-oriented architecture is defined in combination with an algorithm for service composition (see, e.g., [7, 8]).
これはサービス指向アーキテクチャが､サービス合成のためのアルゴリズムを組み合わせて定義された後の事例である。
The aim at Client RE is to facilitate the specification of service requests at runtime.
Client REの目標は、実行時にサービス要求の記述を容易にすることである。
This involves, among other expressing constraints on desired outputs, quality criteria/parameters for evaluating the output and the way in which it is produced.
これには、いくつかある表現の中で、出力と作られ方の値を求めるために、目的とする出力や品質基準や品質パラメータを含む。
This can be performed by traditional RE methodologies.
これは、従来の要求工学方法論によって実行することができる。
In addition, Client RE ought to enable the definition of mechanisms for updating the service requests specification according to change in AOSS's behavior at runtime.
加えてClient REは、
実行時にAOSSのふるまいの変化に従って、サービス要求仕様書を更新するためのメカニズムの

定義を可能にするべきである。
The set of constraints and quality parameters is likely to vary as new services appear and other become unavailable.
一連の制約と品質パラメータは、新しいサービスが現れ、他方が利用できなくなるので変化するだろう。
Quality parameter values will vary as well, as the system adapts to the availability of the various services and change in stakeholders' expectations.
品質パラメータの値は同様に、システムが様々なサービスの可用性とステークホルダの期待の変化を適応させるので、変化するだろう。

3 Using DRAM at Client RE　Client REでDRAMを用いる

We arrived above at the conclusion that there are two tasks to perform at Client RE:
我々は、Client REを成し遂げるために、２つの仕事があるという結論(推論)にたどり着いた。
(a) specification of requirements that result in service requests, and (b) the definition of mechanisms for keeping [these requirements current with behaviors of the AOSS] and [degrees of quality it can achieve over the various quality parameters defined in the requirements].
(a)サービス要求の結果として必要なことの仕様書
(b)AOSSのふるまいに精通したサービス要求を保つことと、その要求で定義された様々な品質パラメータを実現し得る品質の程度を保つことによるメカニズムの定義
We focus now on Client RE, assume the use of an established RE methodology for accomplishing (a), and introduce the Dynamic Requirements Adaptation Method (DRAM) to perform (b).
我々は、今Client REに焦点をあて、(a)を成し遂げるために認められた要求工学方法論の使用を想定し、(b)を動作するために、DRAMを紹介する。
DRAM is thus not a standalone RE methodology—it does not indicate,
DRAMは上で述べたように単独の要求工学方法論ではない。それは指し示さない。
e.g., how to elicit stakeholder expectations and convert these into precise requirements.
例えば、ステークホルダに予期されるものの抽出や、これら(予期されるもの)を正確な要求に変換する仕方である。
Instead, DRAM integrates concepts and techniques for defining mappings between fragments of the requirements specification produced by an existing RE methodology and elements defining a service request (SReq).
その代わり、DRAMは、存在する要求工学方法論とサービス要求を記述する要素を用いて、生産された要求仕様書の断片間の対応付けを記述するために概念と技術を集約した＝ための方法論。
Mapping requirements onto SReqs aims to ensure that the stakeholders' expectations are translated into constraints and quality parameters understood by the AOSS.
サービス要求上の要求対応付けは、ステークホルダの期待がAOSSによって理解される制限と品

質パラメータの中で翻訳されることを確実に保証することを目的とする(保証するように向ける)。

Mapping in the other direction—from SReqs onto requirements—allows the initial (also: static) requirements specification to be updated to reflect runtime changes in the system due to adaptability and openness.

他の方向での対応付け——要求上のサービス要求から——は、適応性と開放性が原因でシステムの実行時の変化に反映するために最初(また、静的な)の要求記述を更新することを許す。

The specification obtained by applying DRAM on the initial, static requirements specification is referred to as the dynamic requirements specification.

最初の、または静的な要求仕様書に関して、DRAMに適用することによって得られた仕様書は、動的な要求仕様書と呼ばれる。

…

※　省略　※

…

4 Related Work　関連研究

Engineering requirements and subsequently addressing completeness concerns for AOSS has only recently started to receive attention in RE research.

AOSSによって、要求工学とその次に完全性に取り組む関心事が、最近になってやっと、要求工学の研究に注目を集め始めた。

Berry and colleagues [1] argue in a note that, while much effort is being placed in enabling adaptive behavior, few have dealt with how to ensure correctness of software before, during, and after adaptation, that is, at the RE level.

Berryとその同僚の有名な議論の中で、多くの成果が順応行動を可能にするとしている。
少数の成果は、要求工学の段階で、適応前後や適応中のソフトウェアの正しさをどのように保証するのかに対処する。

They recognize that RE for such systems is not limited to the initial steps of the system development process, but is likely to continue in some form over the entire lifecycle of the system.

彼らはそのようなシステムのための要求工学はシステムの開発プロセスの初期段階で限定されないと評価するが、システムのライフサイクル全体からいくつか継続するようだ。

Zhang and Cheng [19] suggest a model-driven process for adaptive software; they represent programs as state machines and define adaptive behaviors usually encountered in adaptable systems as transitions between distinct state machines, each giving a different behavior to the system.

ZhangとChengは柔軟な(順応できる)ソフトウェアのためのモデル駆動型プロセスを提案した。
彼らは、状態機械としてプログラムを表現し、
それぞれ異なったふるまいをシステムに与えている異なった状態機械間の遷移として、柔軟なシステムが大抵経験する順応できるふるまいを定義する。

Being situated more closely to the design phase of development than to RE, Zhang and Cheng's process has been related [2] to the KAOS RE methodology by using A-LTL instead of temporal logic employed usually in KAOS.
要求工学よりも開発の設計段階の方が密接な位置にあるとしているが、
ZhangとChengのプロセスは、KAOS法で通常用いられた時相倫理の代わりに、A-LTLを使うことにより、KAOS要求方法論に関係があるといえる。
In the extended KAOS, a requirement on adaptation behavior amounts to a goal refined into two sequentially ordered goals, whereby the first in the sequence specifies the conditions holding in the state of the system before adaptation while the second goal gives those to hold in the state after adaptation.
KAOS法の延長では、適応行動に関する要求は、洗練され順序づけられた連続した２つのゴールを意味する。それによって、シーケンスの最初では、適応前のシステムの状態で状態を保つことを明記する。それと同時に、２つめのゴールはそれらに適応後の状態を保つことを課す。
This paper differs in terms of concerns being addressed and the response thereto.
本論文は、取り組みと応答に加え、利害関係に関して異なる。
The suggested separation onto Service, Coordination, and Client RE for AOSS usefully delimits the concerns and focus when dealing with AOSS.
AOSSに対してService RE, Coordination RE, Client RE上に提案する分離が、AOSSを扱う時に役立つように、利害関係と焦点の境界を定める。
The notion of dynamic requirements specification, along with the associated concepts and techniques is novel with regards to the cited research.
動的要求記述の考えは、関連概念と関連技術とともに、前述の研究に関しては奇抜である。

5 Conclusions and Future Work　　結論と今後の課題

This paper presents one approach to addressing the difficulties in the RE of AOSS.
本研究論文では、AOSSの要求工学の問題に対処するためのひとつのアプローチを提示する。
We argued that the RE of AOSS involves the specification of requirements that may vary at runtime.
我々は、AOSSの要求工学が、実行時に変化する可能性のある要求仕様書に影響を与えるということを主張した。
We consequently identified the most variable class of AOSS requirements and proposed DRAM, a method for specifying these within dynamic requirements specifications.
それ故に我々は、AOSS要求で最も変化しやすいクラスと、動的要求仕様書の中でそれらを明確に述べるための方法として提案されたDRAMとを識別した。
The method has the benefit that it can be combined to any RE methodology.
その方法（DRAM？）は、いくつかの要求工学方法論を組み合わせることで、役立つ。
Its principal limitation at this time is the lack of automated means for defining or facilitating the definition of update rules.
現段階では、これ（DRAM?）の主な制限としては、更新ルールの定義を定義すること、または

それを容易にすることに対して、自動化手段の欠如が挙げられる。
Automation of the DRAM process by reusing results in defeasible logic programming is the focus of current work.
実行可能でない（defeasible＝de+feasible?）論理プログラムをもたらす再利用によって、DRAM プロセスの自動化は最新の研究の焦点となる。

References　参考文献

1. D. M. Berry, B. H. Cheng, J. Zhang. The four levels of requirements engineering for and in dynamic adaptive systems. REFSQ'05.
2. G. Brown, B. H. C. Cheng, H. Goldsby, J. Zhang. Goal-oriented Specification of Adaptation Semantics in Adaptive Systems. SEAMS@ICSE'06.
3. J. Castro, M. Kolp, J. Mylopoulos. Towards requirements-driven information systems engineering: the Tropos project. Info. Sys., 27(6), 2002.
4. A. Dardenne, A. van Lamsweerde, S. Fickas. Goal-directed requirements acquisition. Sci. Comp. Progr., 20, 1993.
5. N. R. Jennings. On Agent-Based Software Engineering. Artif. Int., 117, 2000.
6. I. J. Jureta, S. Faulkner, P.-Y. Schobbens. Justifying Goal Models. RE'06.
7. I. J. Jureta, S. Faulkner, Y. Achbany, M. Saerens. Dynamic Task Allocation within an Open Service-Oriented MAS Architecture. AAMAS'07. To appear.
8. I. J. Jureta, S. Faulkner, Y. Achbany, M. Searens. Dynamic Web Service Composition within a Service-Oriented Architecture. ICWS'07. To appear.
9. Y. Kalfoglou, M. Schorlemmer. Ontology Mapping: The State of the Art. Dagstuhl Seminar Proceedings, 2005.
10. J. O. Kephart, D. M. Chess. The vision of autonomic computing. Computer, 36(1):41–52, 2003.
11. E. Letier, A. van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. ACM Sigsoft Softw. Eng. Notes 29(6), 2004.
12. J. March. Bounded Rationality, Ambiguity, and the Engineering of Choice. The Bell J. Econonomics, 9(2), 1978.
13. M. P. Papazoglou, D. Georgakopoulos. Service-Oriented Computing. Comm. ACM, 46(10), 2003.
14. G. R. Simari, R. P. Loui. A mathematical treatment of defeasible reasoning and its implementation, Artif. Int., 53, 1992.
15. D. Tennenhouse. Proactive Computing. Comm. ACM, 42(5), 2000.
16. A. van Lamsweerde, E. Letier. Handling Obstacles in Goal-Oriented Requirements Engineering. IEEE Trans. Softw. Eng., 26(10), 2000.
17. A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. RE'01.
18. J. Zhang, B. H. C. Cheng. Specifying adaptation semantics. WADS'05.
19. J. Zhang, B. H. C. Cheng. Model-Based Development of Dynamically Adaptive Software. ICSE'06.