Definition 1.

Dynamic requirements specification S is < R, ℛ, Q, P, U, A>,

動的要求仕様書S は、R, ℛ, Q, P, U, A の要素を持つ。

where:

R is the static requirements specification (Def.2);

Rは，静的要求記述書である。（定義２）

ℛ the set of service requirements (Def.3);

ℛは、一連のサービス要求（定義３）

Q the set of quality parameters (Def.4);

Qは、一連の品質パラメータ（定義４）

P the preferences specification (Def.5);

Pは、優先権の仕様書（定義５）

U the set of update rules (Def.6); and

Uは、一連の更新ルール（定義６）

A the argument repository (Def.7).

Aは、独立変数の格納場所（定義７）

The aim with DRAM is to build the dynamic requirements specification.

DRAMの目的は、動的要求仕様書を作り上げることである。

Members of R are specifications of nonfunctional and functional requirements, taking the form of,

Rには、次のような形となって現れる、機能要求と非機能要求の仕様書である。

e.g., goals, softgoals, tasks, resources, agents, dependencies, scenarios, or other, depending on the RE methodology being used.

例：ゴール、ソフトゴール、タスク、リソース、エージェント、依存関係、シナリオ、など、要求工学方法論によって用いられるもの

Service requests submitted at runtime express these requirements in a format understandable to service composers in the AOSS.

サービス要求は、特定のそれらの要求を、AOSSのサービス構築者に理解できる形式で、実行時に提案された。

Nonfunctional requirements from R are mapped onto elements of Q and P, whereas functional requirements from R onto service request constraints grouped in ℛ

Rからの非機能要求は、QとPの要素で対応づけられた。

それに対し、Rからの機能要求は、ℛで分類したサービス要求制約で対応付けられた。

[As equivalence between fragments of R and R, ℛ, Q, P can seldom be claimed], a less demanding binary relation is introduced:
R、ℛ、Q、P の断片間の同値は、めったに獲得することができないので、
少数の要求する２変数の関係が導かれる。
the justified correspondence "≙," between two elements in S indicates [that there is a justification for believing [that the two elements correspond in the given AOSS, at least until a defeating argument is found [[[which breaks the justification]]].
正当化された文書である≙、
Sの２つの要素間は　～を意味する、示唆する、暗示する/
確信することによる正当化である。/　２つの要素が、任意のAOSSで合致する/
少なくとも議論が終わるまでは、発見される。/
正当化を破る（更新する）ことを
In other words, the justified correspondence establishes a mapping between instances of concepts and relationships in the language [in which] [members of R are written] and the language [in which] [members of ℛ, Q, P are written].
言い換えれば、正当化された文書は、Rで書かれた言語とℛ, Q, Pで書かれた言語の中で、
概念と関係のインスタンス間を対応付けることを成立させる。
The preferences specification P contains information [needed to manage conflict] and subsequent negotiation [over quality parameters [that cannot be satisfied simultaneously to desired levels]].
優先権の仕様書Pは、所望のレベルを同時に満たすことができない品質パラメータをめぐる話し合いと、その後の交渉を管理するために必要な情報が含まれる。
Update rules serve to continually change the contents of R according to system changes at runtime.
更新ルールは、継続的に実行時にシステムの変化に応じてRの内容を変更するのに役立つ。
Finally, the argument repository A contains knowledge, arguments, and justifications used to construct justified correspondences and at other places in S, as explained below.
ようやく、独立変数格納場所であるAは、以下に説明するように、
Sの他の場所と正当化された文書は構築に用いられた知識と議論と正当化を含む。
S is continually updated to reflect change in how the service requests are fulfilled.
Sは継続的にサービス要求を満足させる方法の変化を反映するために更新される。
Updates are performed with update rules:
更新は、更新ルールを用いて行われる。
an update rule will automatically (or with limited human involvement) change the R according to the quality parameters, their values, and the constraints on inputs and outputs characterizing the services composed at runtime to satisfy service requests.
更新ルールは、自動的にまたは人の関与で、

サービス要求を満足するために実行時に構成されたサービスを特徴付ける入出力における品質パラメータ、それらのルール、その制約に従いRを変更する。

An update rule can thus be understood as a mapping between fragments of R and those of ℛ, Q, P.

更新ルールは、上で述べたように（このように）、ℛ とQとP のそれらとRの断片間の対応付けとして、理解することができる。

Consequently, an update rule is derived from a justified correspondence.

それ故に、更新ルールは、正当化された文書から導かれる。

It is according to the constraints on inputs/outputs and quality parameter values [observed at runtime [that fragments of requirements will be added or removed to R]].

それ（更新ルール）は、実行時に観測される要求の断片はRを取り除かれたり、追加されたりするだろう、入出力と品質パラメータのルールの制約に従う。

Update rules work both ways, i.e., change in R is mapped onto [service requests, and the properties of services participating in compositions are mapped onto fragments of R].

更新ルールは両方をもたらす。

すなわち、Rの変化は、Rの断片で対応付けられる、合成に関わるサービス特性とサービス要求で対応づけられる。

Building fully automatic update rules is difficult for it depends on the precision of the syntax and semantics of languages used at both ends,

完全に自動更新ルールを作り上げることは、両極端で使われる意味言語と構文言語の正確さ次第なので、難しい。

i.e., the specification language of the RE methodology [which produces R and the specification language employed to specify input/output constraints on services and quality parameters].

すなわち、品質パラメータとサービスにおける入出力の制約を記述するために用いられた仕様化言語と、Rを作る要求工学方法論の仕様化言語

Due to a lack of agreement on precise conceptualizations of key RE concepts (e.g., [17]), DRAM makes no assumptions about the languages employed for writing R, ℛ and Q.

主要な要求工学の概念の、明確な概念化に関する協定の欠如を理由に、

DRAMは、R とℛ とQを書くために作られた言語について何も仮定しない。

Hence the assumption [that languages at both ends] are ill-defined, and the subsequent choice of establishing a “justified” correspondence (i.e., a defeasible relation) between specification fragments.

それ故に、両端の言語の仮定と、それに続く仕様書の断片間の“正当化された”文書（すなわち、実行不可能な関係）を規定することの選択は、明確に定義されていない。

An unfortunate consequence is [that update rules in many cases cannot be established automatically—a repository of update rules is built during testing and at runtime.

運の悪い結果は、多くの場合、更新ルールは自動的に作られることができないことである。

すなわち、更新ルールの格納場所は、実行時やテストをしている間に作られるから。

S integrates the necessary means for constructing update rules:

Sは、更新ルール構成するために必要な手段を統合する。

to build justified correspondences between elements of R and ℛ, Q, P, arguments are built and placed in the argument repository A.

R とℛ とQ とP の要素間の正当化された文書を作るために、独立変数は作られ、Aの独立変数の保管場所に入れられる。

Update rules are automatically extracted from justified correspondences.

更新ルールは、自動的に正当化された文書から抽出される。

As competing services will offer different sets of and values of [quality parameters at service delivery, and as not all will be always available, trade-offs performed by the AOSS need to be appropriately mapped to R.

競合するサービスは、サービス提供における品質パラメータの値の違いやセットの違いを提供するので、

全て（のサービス）がRを適切に対応付けるために必要なAOSSによって提供される取引をいつもできるようになるわけではない。

Moreover, stakeholders may need to negotiate the quality parameters and their values.

さらに、ステークホルダは、それらの値と品質パラメータを交渉するために必要とするかもしれない。

P performs the latter two roles.

Pは、後者の２つのルールを実行する。

DRAM proceeds as follows in building the dynamic requirements specification (concepts and techniques referred to below are explained in the remainder).

DRAMは、動的要求記述を作ることを引き継ぐこととして進める。

（概念と技術は以下の残りの部分で説明される）

Building the dynamic requirements specification with DRAM

DRAMで動的要求仕様書を作る

1. Starting from the static requirements specification R (Def.2), select a fragment r ∈ R of that specification [that has not been converted into a fragment in ℛ (Def.3), Q (Def.4), and/or P (Def.5)].

   静的要求仕様書R（定義２）から始めること

   ℛ 、Q およびP の要素から変換されない仕様書のRの要素 r を選択する。

2. Determine the service requirement and/or quality parameter information [that can be extracted from r as follows]:

   サービス要求およびサービス品質の、次のような r から抽出することができる情報を決定する。

(a) If r is a functional requirement (i.e., it specifies a behavior to perform), focus is on

building a justified correspondence (see, Def.6 and Technique 1) between r and elements of service requirements.
もしｒが機能要求であれば（それは実行するためのふるまいを記述する）、焦点はサービス要求と要素とｒ間で正当化された文書（定義６と技術１参照）を作ることにある。

Consider, e.g., the following requirement:
例えば、次の要求を考えることとする。

Each user of TravelWeb expects a list of available flights for a destination to be shown within 5 seconds after submitting the departure and destination city and travel dates.

TravelWebの各ユーザは、出発地と目的地の町と旅行日を提示した後、５分以内に見られるようにするために飛行機の予約のリストを期待する。

available(depC, depD, arrC, arrD, flight)∧correctFormat(depC, depD, arrC, arrD)
⇒ $\Diamond_{5s}$shown(searchResults, flight)

available(出発地C、出発地D、目的地C、目的地D、飛行機)　※予約したいもの？
∧　correctFormat(出発地C、出発地D、目的地C、目的地D)　※抽出した情報？
⇒ $\Diamond_{5s}$shown（検索結果、飛行機）

Starting from the above functional requirement:
機能要求上から始めること

Ⅰ. Identify the various pieces of data [that are to be used (in the example:depC, depD, arrC, arrD, flight)] and those [that are to be produced (searchResults)]according to the requirement.

使われるための（例えば、出発地C、出発地D、目的地C、目的地D、飛行機など）データの様々な要素や、要求に従って、提示（提供）するためのデータの様々な要素（検索結果）を特定する。
※要するに、入力に必要なものと出力に必要なものは何かをここで決める。

Ⅱ. Find services [that take the used data as input] and [give produced data at output] (e.g., FlightSearch Serv, s.t. {depC, depD, arrC, arrD, flight} ⊆ Ⅰ ∧ searchResults∈O).

入力として使われたデータを扱うサービスや、出力として提供されたデータを与えるサービスを検索する。
例：{ 出発地C、出発地D、目的地C、目的地D、飛行機}⊆ Ⅰ　∧　検索結果 ∈ O
となるような飛行機検索サービスが存在する。
※(入力に使われるデータが、入力に含まれる）∧（検索結果が、出力データを含む）というサービスが存在する。

Ⅲ. Determine [whether] the service requirements available on inputs justifiably corresponds to the conditions on input data in the requirement, and [perform the same for output data (i.e., check if there is a justified correspondence between input/output service requirements and conditions in the relevant requirements in R—i.e., use Def.6 and Technique 1)].
要求中の入力データの状態に正当に一致する、入力で利用できるサービス要求か

どうか、また、出力データで同じことを実行する（すなわち、Rに関連性のある要求の状態と入出力のサービス要求間で正当化された文書があるかどうかをチェックする。　—すなわち、定義６と技術１を使う。）かどうかを決定する。

想定する入力データ　　実際の入力データ
（つまり要求中の）

If constraints do not correspond (justified correspondence does not apply), map the conditions from the requirement in R into constraints on inputs and/or outputs, and write them down as service requirements.
もし、制約が一致しなかったら（正当化された文書が当てはまらない）、
入力および出力の制約にRの要求からの状態を対応づけ、それら（の対応付け）をサービス要求として書き記す。

If there is no single service [that satisfies the requirement (i.e., step 2(a) Ⅰ above fails)], refine the requirement (i.e., brake it down into and replace with more detailed requirements)—to refine, apply techniques provided in the RE methodology.
もし、それが要求を満たす単一のサービス（すなわち、above fails手順２-(a) Ⅰの逆でなかったら、要求を改善する。（すなわち、要求を分解し、より詳細化された要求に置き換える）
—要求工学方法論で提供されている技術をあてはめたり、改善したりするために

Ⅳ. Use step 2b to identify the quality parameters and preferences related to the obtained service requirement.
獲得したサービス要求に関係している優先権と品質パラメータを特定するために、ステップ２（ｂ）を使う。

(b) If r is a nonfunctional requirement (i.e., describes how some behavior is to be performed, e.g., by optimizing a criterion such as delay, security, safety, and so on), the following approach is useful:
もし、ｒが非機能要求であるなら（すなわち、いくつかのふるまいが実行されるためにはどのように表現するのか…例：遅延時間やセキュリティ、安全性などのような基準を最適化することによって実行される）、次のアプローチを用いる。

Ⅰ. Find quality parameters (Def.4) [that describe the quality [at which] the inputs and outputs mentioned in [a particular service requirement are being used and produced]].
特定のサービス要求が提供され、使われている中に記載されている入力と出力の品質を表現する、品質パラメータ（定義４）を探し出す。

In the example cited in the DRAM process, [the delay between the moment input data] is available and [the moment it is displayed to [the user can be associated to a quality parameter [which measures the said time period].
前述のDRAM中の例では、データを入力する間の遅延時間は、有効であり、

その一瞬は、ユーザが、時間周期と呼ばれる測定単位で品質パラメータを関連付けできるように表示される。

Ⅱ. Following Def.4, identify the various descriptive elements for each quality parameter.
以下の定義４では、それぞれの品質パラメータのためにさまざまな記述的（何が事実か事実でないかを客観的に検証する）な要素を特定する。

Use R as a source for the name, target and threshold value, and relevant stakeholders.
名前やターゲットやしきい値や関連性のあるステークホルダのための情報源として、Rを使用する。

If, e.g., Tropos is employed to produce R, softgoals provide an indication for the definition of quality parameters.
もし、例えばTroposがRを作るために採用されたとしたら、
ソフトゴールは、品質パラメータの定義のための目安を提供する。

Ⅲ. For each quality parameter that has been defined, specify priority and preferences.
定義されたそれぞれの品質パラメータのために、優先度と優先権を明確に述べる。
Initial preferences data for trade-offs comes from test runs.
妥協のための最初の優先権のデータは、試運転によってもたらされる。

3. Write down the obtained $r \in \mathcal{R}$, $q \in Q$, and/or $p \in P$ information, along with arguments and justifications used in mapping r into r and/or q.
r および q の中で、 r の対応付けに用いられる正当化する理由と根拠(独立変数？)に沿って、
獲得したr（Rに含まれる）と q（Qに含まれる）および p（Pに含まれる）の情報を記録する。
Each justified correspondence obtained by performing the step 2. above is written down as an update rule $u \in U$.
上記のステップ２を実行することによって獲得したそれぞれの正当化された文書は、更新ルール u（Uに含まれる）として記録される。
4. Verify [that the new arguments added to A do not defeat justifications already in A]; revise the old justifications if needed.
Aに加えられる新しい引数が、Aですでに正当性を破ることを確認する。
必要に応じて、古い正当性を修正する。

Definition 2. 定義２

The static requirements specification R is the high-level requirements specification obtained during RE before the system is in operation.
静的な要求仕様書Rは、システムが実施される前の要求定義活動中に獲得された高度な要求仕様書である。

R is obtained by applying a RE methodology, such as, e.g., KAOS [4] or Tropos [3].
Rは、(例えばKAOSやTroposのような)要求工学方法論を適合することによって獲得される。

The meaning of “high-level” in Def.2 varies accross RE methodologies:
定義２の“高度な”とは、要求工学方法論により、異なるということを意味する。

if a goal-oriented RE methodology is employed, R must contain the goals of the system down to the operational level, so that detailed behavioral specification in terms of, e.g., state machines, is not needed.
ゴール指向要求工学方法論を利用する場合、例えば、状態機械に関して、詳細な挙動詳細が必要とされないので、Rは、システムのゴールを動作レベルに至るまでを含まなければならない。
If, e.g., KAOS is used, the engineer need not move further than the specification of goals and concerned objects, that is, can stop before operationalizing goals into constraints.
例えばKAOS法が用いる場合、
エンジニアは、関連オブジェクトとゴールの仕様書より遠くに移動する必要がなくなる。
すなわち、エンジニアは制約中のゴールを操作可能にする前に停止することができる。
If Tropos is used, the engineer stops before architectural design, having performed late requirements analysis and, ideally, formal specification of the functional goals.
Troposを用いる場合、エンジニアは、末期の要求分析と、
概念的に機能的ゴールの形式仕様を実行している設計段階の前に停止する。

Example 1.　例 1

When a RE methodology with a specification language grounded temporal first-order logic is used[[1]], the following requirement r ∈ R for TravelWeb states [that all options [that a service may be offering to the user] should be visible to the first time user]:
一時的な一階論理に基づいた仕様化言語を備えた要求工学方法論が利用されたとき、
サービスがユーザに提供する可能性のあるすべてのオプションとなるTravelWebの状態のための次の要求 r は、初めてのユーザ（初心者）のために明白でなければならない。

1stOpt ≡ (hasOptions(servID) ∧ firstTimeUser(servID, userID) ⇒
$\Diamond_{1s}$showOptions(all, servID, userID))

1stOpt(備わっているオプション(サービスID)∧初心者(サービスID,ユーザID)) ⇒
$\Diamond_{1s}$提示するオプション（全てのオプション,サービスID,ユーザID）

---------- 注釈 ----------

[[1]]
Assuming, for simplicity, a linear discrete time structure, one evaluates the formula for a given history (i.e., sequence of global system states) and at a certain time point.
The usual operators are used:

for a history H and time points i, j, (H, i) $\models \circ \phi$ iff (H, next(i)) $\models \phi$; (H, i) $\models \Diamond \phi$

iff $\exists j > i$, (H, j) $\models \phi$; (H, i) $\models \Box \phi$ iff $\forall j \geq i$, (H, j) $\models \phi$.

Mirror operators for the past can be added in a straightforward manner.

Operators for eventually ◇ and always □ can be decorated with duration constraints, e.g., $\Diamond_{\leq 5s} \phi$ indicates that ϕ is to hold some time in the future but not after 5 seconds. To avoid confusion, note that → stands for implication, while $\phi \Rightarrow \psi$ is equivalent to $\Box(\phi \rightarrow \psi)$.
For further details, see, e.g., [16].

---------- 注釈 ----------

Definition 3. 定義３

A service requirement r ∈ R is a constraint on service inputs or outputs [that appears in at least one service request] and [there is a unique r ∈ R such that there is a justified correspondence between it and r].
あるサービス要求ｒ（Rに含まれる）は、少なくとも１つのサービス要求に現れるサービスの入力または出力に対する制約がある。
また、固有のｒはｒとそれ（何を指す？）との間に正当化された文書が存在する。

Example 2. 例２

Any service [that visualizes to the TravelWeb user the options [that other services offer] when booking] obeys the following service requirement:
予約する時に、他のサービスが提供するオプションをTravelWebのユーザに視覚化する任意のサービスは、次のサービス要求に従う。

Υ = (**input** :servID ∉ userID.visited ∧ servID.options ≠ ϕ;
output :thisService.show = servID.options)

Υは、(入力：サービスID、そのサービスIDのオプション
出力：このサービスを視覚化したもの、サービスIDのオプション)

Definition 4. 定義４

A quality parameter q ∈ Q is a metric expressing constraints on how the system (is expected to) performs.
品質パラメータｑは、どのようにシステムが実行するために期待されているかといった測定基準を表現している制約である。

q = <Name, Type, Target, Threshold, Current, Stakeholder>,

qは名前、型、ターゲット(目標)、 しきい値、最新性(最新値？)、ステークホルダを持つ。

where

Name is the unique name for the metric;
名前は、測定基準のための固有名詞である。
Type indicates the type of the metric;

型は、測定基準の型を意味する。
Target gives a unique or a set of desired values for the variable;
ターゲットは、 変数のための一連の目標値または特定の目標値を与える。←何に？
Threshold carries the worst acceptable values;
しきい値は、最悪時の許容値を持っている。
Current contains the current value or average value over some period of system operation;
最新値は、システム運用の期間を越えた平均値や現在の値を含む。
and
Stakeholder carries names of the stakeholders [that agree on the various values given for the variable].
ステークホルダは、変数のために与えられたさまざまな値について一致するステークホルダの名前を持っている。

Example 3.　例３

The following quality parameters can be defined on the service from Example 2:
次の品質パラメータは、例２からサービスで定義されることができる。

q1 = <ShowDelay, Ratio, 500ms, 1s, 780ms, MaintenanceTeam>

q1は、視覚化した遅延時間(名前),割合(型),500ms(目標値), 1s(しきい値), 780ms(最新値), メンテナンスチーム(ステークホルダ)を持つ。

q2 = <OptionsPerScreen, Ratio, {3,4,5}, 7, (all), UsabilityTeam>

q2は、スクリーン毎のオプション(名前), 割合(型), {3,4,5} (目標値), 7(しきい値), すべて(最新値),ユーザビリティチーム(ステークホルダ)を持つ。

q3 = <OptionsSafety, Nominal, High, Med, Low, MaintenanceTeam>

q3は、オプションの安全性(名前),Normal(型),High(目標値), Med(しきい値), Low(最新値), メンテナンスチーム(ステークホルダ)を持つ。

q4 = <BlockedOptions, Ratio, 0, ($\geq$ 1), 0, MaintenanceTeam>

q4は、ブロックしたオプション(名前) , 割合(型), 0 (目標値), 1以上(しきい値), 0(最新値), メンテナンスチーム(ステークホルダ)を持つ。

As quality parameters usually cannot be satisfied to the ideal extent simultaneously, the preference specification contains information on priority and positive or negative interaction relationships between quality parameters.
通常、品質パラメータは、同時に仮想上の空間を満足させることができないので、
優先権の仕様書は、優先権に関する情報と確かな情報、または品質パラメータ間の悪い相互作用関係に関する情報を含む。
Prioritization assists [when negotiating trade-offs], [while interactions indicate trade-off directions between parameters].
優先順位付けは、交渉が矛盾するときに役立ったり、相互作用がパラメータ間の方向を変

換すること意味するときに役立ったりする。

Definition 5.　定義５

The preferences specification is the tuple P = <$\succ$,$P^{\succ}$,$P^{\pm}$ >.

優先権の仕様書は、組Pである。

“$\succ$” is a priority relation over quality parameters.

“$\succ$”は、品質パラメータを越えた優先関係である。

The set $P^{\succ}$ contains partial priority orderings,

集合$P^{\succ}$は、一部優先の順序付けを含む。

specified as (qi $\succ$ qj , **Stakeholder**) $\in$ $P^{\succ}$

「ステークホルダに対して、qi の方がqj より優先度が高い」として、明記される。

where qi carries higher priority than qj ,

q1 は、qj より高い優先度を持つことを意味する。

and **Stakeholder** contains the names of the stakeholders agreeing on the given preference relation.

ステークホルダは,与えられた優先権の関係について一致するステークホルダの名前を含む。

Higher priority indicates [that a trade-off between the two quality parameters [will favor the parameter with higher priority]].

高い優先度は、高い優先度をもつパラメータを支持するだろう２つの品質パラメータ間の妥協点(矛盾？どっち？) を意味する。

The set P± contains interactions.

集合P±は、相互作用を含む。

An interaction indicates [that a given variation of the value of a quality parameter results in a variation of the value of another quality parameter].

相互作用は、他の品質パラメータの値について変化をもたらす、品質パラメータの値の与えられた変化を意味する。

An interaction is denoted (q1 $\overset{b_1 \Rightarrow b_2}{\longleftrightarrow}$ q2)@ϕ.

相互作用は、(q1 $\overset{b_1 \Rightarrow b_2}{\longleftrightarrow}$ q2)を意味する。

q1 $\overset{b_1 \Rightarrow b_2}{\longleftrightarrow}$ q2 indicates [that changing the value of the quality parameter q1 by] or [to b1 necessarily leads the value of the parameter q2 to change for or to b2].

q1 $\overset{b_1 \Rightarrow b_2}{\longleftrightarrow}$ q2は、品質パラメータ q1 の値が変化すること、つまり、b1が必然的に、b2に

変わるために、またはb2に変化するために、パラメータq2の値を導くことによって、変化することを意味している。
As the interaction may only apply when particular conditions hold, an optional non-empty condition ϕ can be added to indicate when the interaction applies.
相互関係は、特定の状況を保持する時にだけ利用される可能性があるので、
任意の非空状態ϕは、相互関係を適応する時に表示するために、加えられることができる。
The condition is written in the same language as service requirements.
状態は、サービス要求として同じ言語で記述される。
When the relationship [between the values of two quality parameters] can be described with a function, we give [that functional relationship instead of b1 $\Rightarrow$ b2].
２つの品質パラメータの値間の関係は、機能によって描写されることができるとき、
我々は、b1ならばb2の代わりに、関数関係を与える。

Example 4.　例４　↓↓↓以降

Starting from the quality parameters in Ex.3, the following is a fragment of the preferences specification:
例３の品質パラメータから始める。次のものは、優先権の仕様書の断片である。

p1 = (OptionsPerScreen $\overset{+1\Rightarrow+60ms}{\longleftrightarrow}$ ShowDelay) @ (OptionsPerScreen > 4)

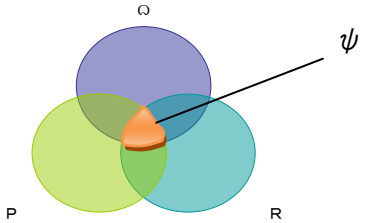
p1は、スクリーン毎のオプション $\overset{+1\Rightarrow+60ms}{\longleftrightarrow}$ 視覚化した遅延時間

※スクリーン毎のオプションは４より大きい

p1 indicates [that increasing the number of options per screen by 1 increases the delay to show options to the user by 60ms, [this only if the number of options to show] is above 4].
p1は、～を意味する。
これがオプションの数字が４より大きいならば、・・・

Definition 6.　定義6

A justified correspondence exists between $\phi \in R$ and $\psi \in \mathcal{R} \cup Q \cup P$,
正当化された文書はϕとψ($\mathcal{R}$ とQとPの共通部分)の間に存在する。



i.e., $\phi \triangleq \psi$ iff there is a justification <P, $\phi \triangleq \psi$ >.

すなわち、正当化の理由<P,$\phi \triangleq \psi$>があるその時かつその時限り(必要十分条件)
$\phi \triangleq \psi$であるという。

Recall from the above [that the justified correspondence] is a form of mapping [in which very few assumptions] are made on the precision] and [formality of the languages being mapped].

上述の正当化された文書から振り返ることは、仮定がほとんどないことは、対応付けられた言語の形式と正確さを実現することにおける、対応付けの形式である。

This entails the usual difficulties (as those encountered in ontology mapping, see, e.g., [9]) regarding conversion automation and the defeasibility of the constructed mappings, making DRAM somewhat elaborate to apply in its current form.

これは、通常の難しさ（オントロジーマッピングで現れるような,例：参考文献９参照）
変更自動化と構造化マッピングの実現不可能性に関しての難しさ、
現在の形で適応するために少々DRAMを念入りに作ることの難しさを伴う。

Defeasibility does, however, carry the benefit of flexibility in building and revising mappings.

実現不可能性は、しかしながら、マッピングを修正したり、作ったりすることで柔軟性の利点を支援する。

Definition 7.　定義 7

A justification < P, c > is an argument [that remains undefeated after the justification process]. [[2]]

正当化する理由である Pとc は、正当化するプロセスの後に負けないままでいるための根拠(独立変数？)である。

---------- 注釈 ----------

[[2]]

Some background [14]:

Let A a set of agents (e.g., stakeholders) and the first-order language L defined as usual. Each agent a $\in$ A is associated to a set of first-order formulae K_a which represent knowledge taken at face value about the universe of discourse, and Δ_a which contains defeasible rules to represent knowledge which can be revised.

Let K $\equiv \cup_{a \in A} K_a$, and $\Delta \equiv \cup_{a \in A} \Delta_a$.

“|～” is called the defeasible consequence and is defined as follows.

Define $\Phi = \{\phi 1, \ldots, \phi n\}$ such that for any $\phi i \in \Phi$, $\phi i \in K \cup \Delta^{\downarrow}$.

A formula ϕ is a defeasible consequence of Φ(i.e., Φ |～ ϕ) if and only if there exists a sequence B1, . . . ,Bm such that ϕ = Bm, and, for each Bi 2 {B1, . . . ,Bm}, either Bi is an axiom of L, or Bi is in Φ, or Bi is a direct consequence of the preceding members of the sequence using modus ponens or instantiation of a universally quantified sentence.

An argument < P, c > is a set of consistent premises P supporting a conclusion c.

The language in which the premises and the conclusion are written is enriched with the binary relation $\hookrightarrow$. The relation $\hookrightarrow$ between formulae α and β is understood to express that "reasons to believe in the antecedent α provide reasons to believe in the consequent β".
In short, $\alpha \hookrightarrow \beta$ reads "α is reason for β" (see, [14] for details).
Formally then, P is an argument for c, denoted <P, c>, iff:
(1) K ∪ P |∼ c (K and P derive c);

(2) K ∪ P $\nvdash \bot$ (K and P are consistent); and

(3) $\nexists$P´ ⊂P , K ∪ P´ |∼ c (P is minimal for K).

---------- 注釈 ----------

Up to this point, the concepts needed in DRAM have been introduced.
このポイント次第で、DRAMで必要とされる概念は、導入される。
The remainder of this section describes the techniques in DRAM [that use the given concepts in the aim of constructing the dynamic requirements specification].
このセクションの残りの部分は、動的要求仕様書を構築するという目標に与えられた概念を用いる、DRAMの技術を表現する。

Technique 1. 技術 1

The justification process [14] consists of recursively defining and labeling a dialectical tree T <P, c> as follows:
正当化するプロセス(参考文献14)は、
次のようなラベル付けし、再帰的に定義された弁証法的な木T<P,c>から成る。

1. [A single node containing the argument <P, c> with no defeaters] is by itself a dialectical tree for <P, c>.
   敗者を伴わない独立変数<P,c>を含んでいる単一のノードは、<P,c>のための弁証的な木のみの単独である。
   This node is also the root of the tree.
   このノードは、木の根（ツリーのルート）でもある。
2. Suppose that <P1, c1> , . . . , <Pn, cn> each defeats [[3]] <P, c>.
   それぞれの<P,c>を駄目にするノードとして、<P1,c1>～<Pn,cn>を仮定する。
   Then the dialectical tree T < P, c > for < P, c > is built [by placing < P, c > at the root of the tree] and [by making this node the parent node of roots of dialectical trees rooted respectively in < P1, c1 > , . . . , < Pn, cn >].
   <P,c>のための弁証的な木Tのノード<P,c>は、

木の根でノード<P,c>を収納することによって築かれ、
また、<P1,c1>～<Pn,cn>のそれぞれに根付いた弁証的な木の根の現在のノードを作ることによっても、築かれる。

3. When the tree has been constructed to a satisfactory extent by recursive application of steps 1) and 2) above, label the leaves of the tree undefeated (U).
ツリーが、上述のステップ１と２の再帰的なアプリケーションによって満たしている範囲を構築するとき、木の葉のラベルは、(U)に屈しない。
For any inner node, [label it undefeated [if and only if] every child of that node] is a defeated (D) node.
内部のノードにとって、それに屈しないラベルの必要十分条件とは、ノードのすべての子が(D)のノードに屈しないことである。
An inner node will be a defeated node [if and only if] it has at least one U node as a child.
内部のノードは、子として少なくとも１つのUのノードを持つことで必要十分条件として、屈しないノードとなるだろう。
Do step 4 below after the entire dialectical tree is labeled.
弁証的な木がラベル表示された後、下記のステップ４を行う。
4. <P, c > is a justification (or, P justifies c) iff the node < P, c > is labelled U.
<P,c>は正当化されている(または、Pはcを正当化する)。
ノード<P,c>はUでラベル表示されているノードであることが必要十分条件である。

Example 5. 例５

Fig.1 contains the dialectical tree for the justified correspondence 1stOpt $\triangleq$ r, where
図１は、正当化された文書1stOpt $\triangleq$ r にとって弁証的な木を含む。
r is from Ex.1 and r from Ex.2.
r は例２から, r は例１からの要素である。
To simplify the presentation of the example, we have used both formal and natural language in arguing.
例の提案説明を簡単にするために、我々は議論で形式言語と自然言語の両方を用いる。
More importantly, notice [that the correspondence 1stOpt $\triangleq$ r] is unjustifed, as it is defeated by an undefeated argument containing information on a quality parameter and a fragment of the preferences specification.
より重要なことは、文書1stOptの通知が不当であることである。
それは、優先権の仕様書の断片と品質パラメータにおける情報を含んでいる独立変数によって免れることができる。
A justified correspondence such as,
次のようなものは正当化された文書である。
e.g., firstTimeUser(servID, userID) $\triangleq$ servID $\notin$ userID.visited, becomes an update rule,
i.e., (firstTimeUser(servID, userID) $\triangleq$ servID $\notin$ userID.visited) $\in$ U.

例えば、
初心者ユーザ(サービスID,ユーザID)・・・・が、更新ルールとなる。
すなわち、(初心者ユーザ(サービスID,ユーザID)・・・・・)

Having established [that justified correspondence], the service requirement is taken to correspond to the given initial requirement until the justified correspondence is defeated.
正当化された文書を確立することは、
サービス要求が正当化された文書が免れるまでに与えられた最初の要求に一致することを追求することである。

[Elements of the argument repository] correspond to the argument structure shown in Fig.1.
独立変数の格納場所の要素は、図１に見られる独立変数の構造に一致する。