UN SISTEMA BASADO EN CONOCIMIENTO PARA LA ELABORACIÓN DE DIETAS

MODELADO CONCEPTUAL EN COMMONKADS

Jorge Hermo González
David Rodríguez Bacelar
Dana Gallent Iglesias

Horario de prácticas: Jueves 8:30 - 10:00
Login de entrega: jorge.hermo.gonzalez

Desarrollo de Sistemas Inteligentes
Universidade da Coruña
Curso 2022

Índice

1. Modelo de Conocimiento.

1.1. Fase de Identificación

Este documento trata sobre la tarea 4 del formulario OM-3: la elaboración de la dieta.

1.1.1. Glosario

- Slot: Casilla de la dieta que modela una comida de un día concreto (desayuno, almuerzo, comida, merienda o cena)
- Insuficiencia renal: Afección en la cual los riñones dejan de funcionar y no pueden eliminar los desperdicios y el agua adicional de la sangre, o mantener en equilibrio las sustancias químicas del cuerpo.

1.1.2. Escenarios

Paciente con problemas de riñón y alergia al pescado, padece insuficiencia renal. Hombre, 64 años y 85kg. El recuento de los últimos días indica que suele realizar 3 comidas diarias. Le gusta mucho la pasta.

Dieta recomendada para el primer día:

- Desayuno: leche desnatada con una tostada de pan integral, aceite de oliva y una manzana.
- Comida: un plato de espaguetis con tomate natural, de segundo plato patata cocida con brócoli, de postre una naranja.
- Cena: arroz con mejillones y espinacas, postre plátano.

1.1.3. Elementos reutilizables

N/A.

1.2. Fase de Especificación

1.2.1. Metodología empleada

La plantilla de Configuration Design se adapta correctamente a nuestro problema. Sin embargo, necesitaremos obtener de alguna manera los alimentos concretos de una base de datos externa, por lo que se ha modificado la plantilla y el pseudocódigo. Estos cambios se especificarán posteriormente.

Hemos decidido entonces seguir una metodología *Middle-out*, partiendo del conocimiento inferencial y construyendo, posteriormente, el conocimiento del dominio y el conocimiento de la tarea.

1.2.2. Plantilla anotada

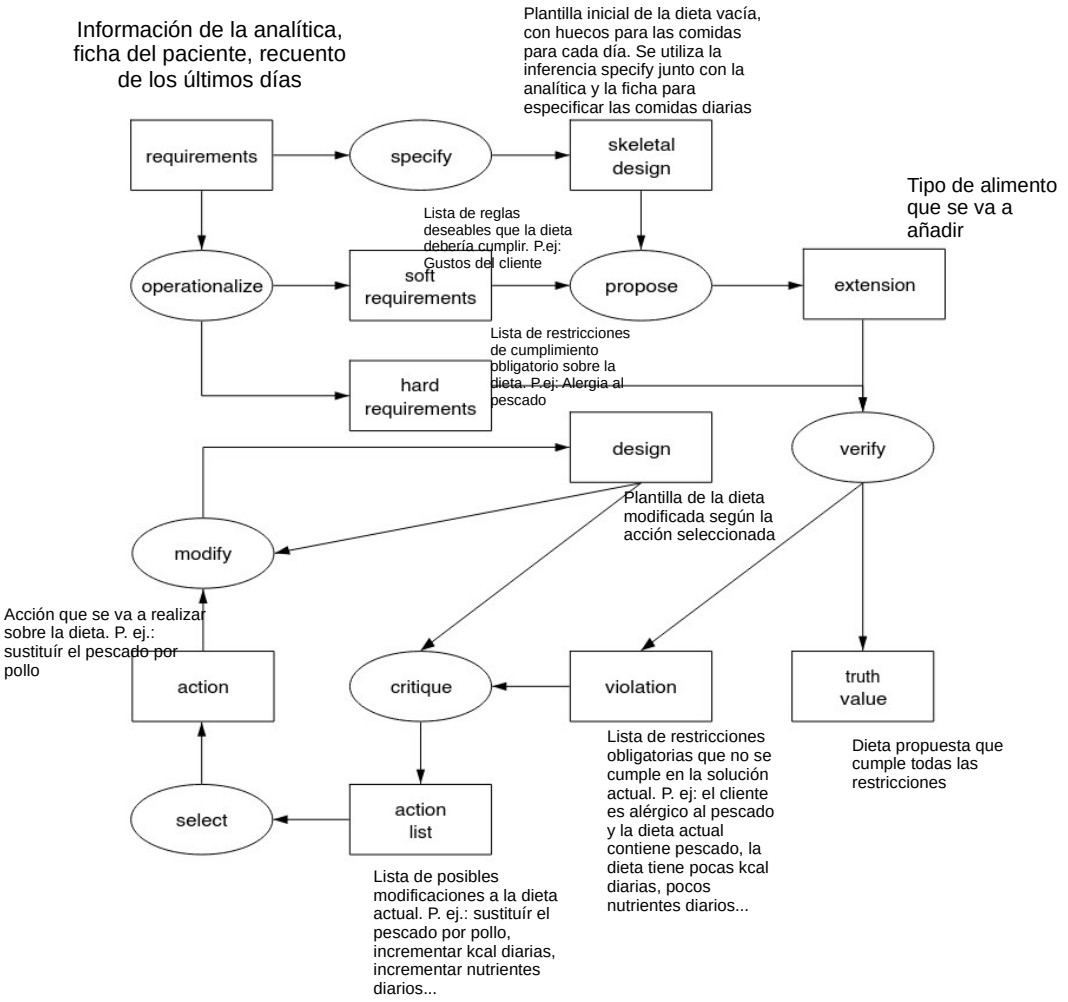

Figura 1: Plantilla Configuration Design anotada

1.2.3. Esquema inicial del dominio

Boceto inicial del esquema del dominio, antes de realizar el mapeo de las inferencias.

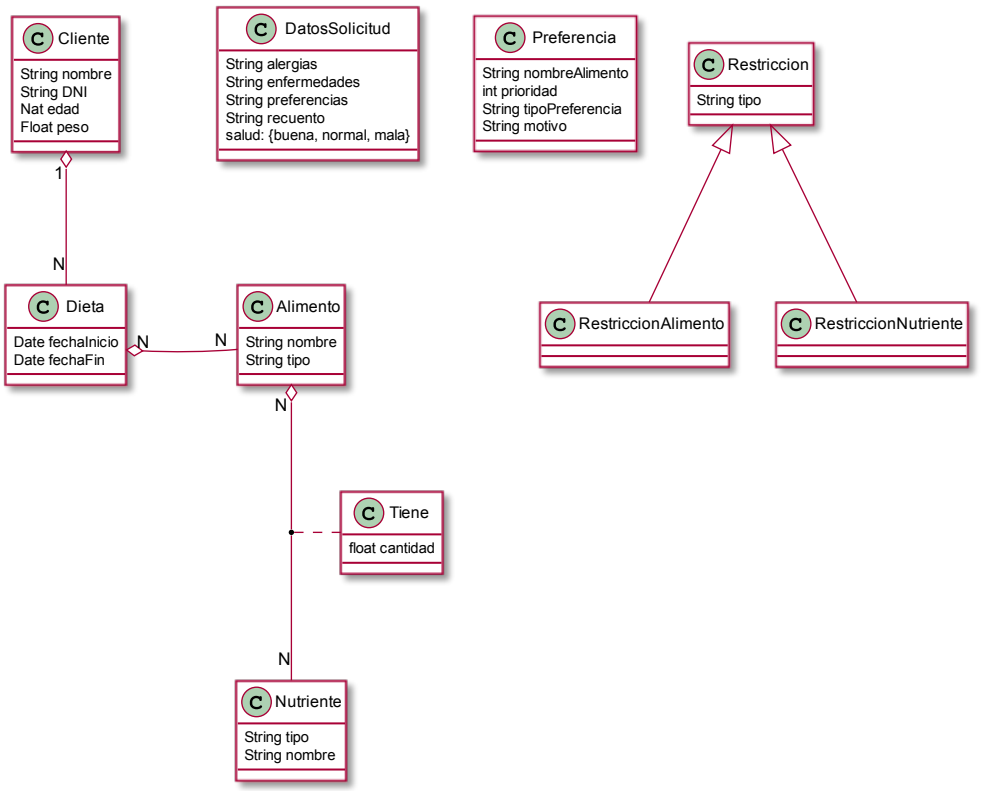

Figura 2: Esquema del dominio inicial

1.2.4. Estructura inferencial y Mapeo

Se han realizado modificaciones sobre la plantilla original. En concreto, se ha cambiado qué se hace con *Extension*, ya que la inferencia *Propose* solo propone un tipo de alimento y necesitamos una función de transferencia *Obtain* para poder obtener un alimento concreto de la base de datos que sea del tipo especificado; este alimento obtenido se modela en *Design Atom*. Posteriormente, el *Design Atom* se tendrá que añadir al slot seleccionado.

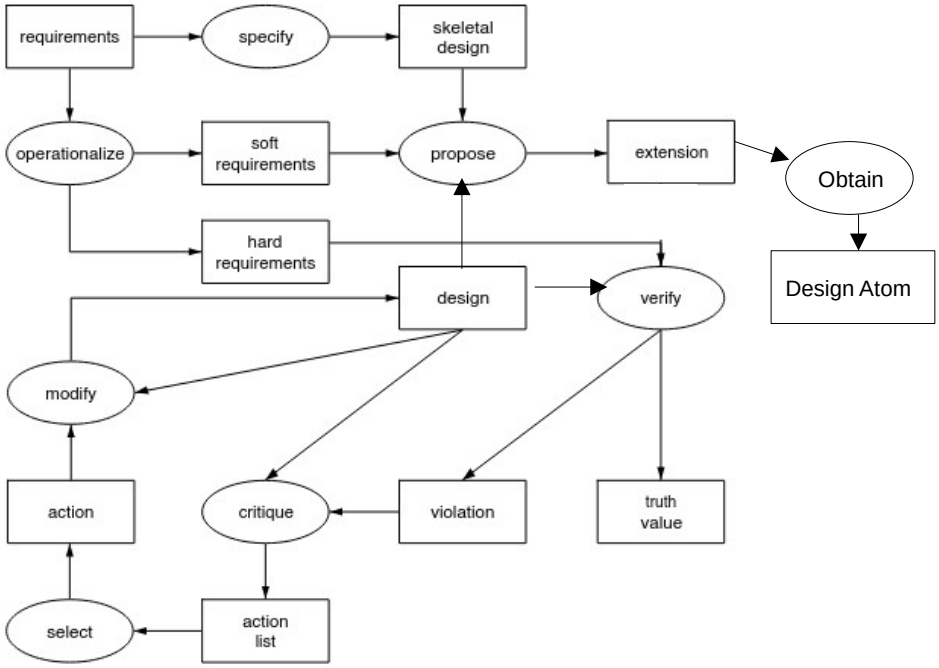

Figura 3: Gráfico de la estructura inferencial

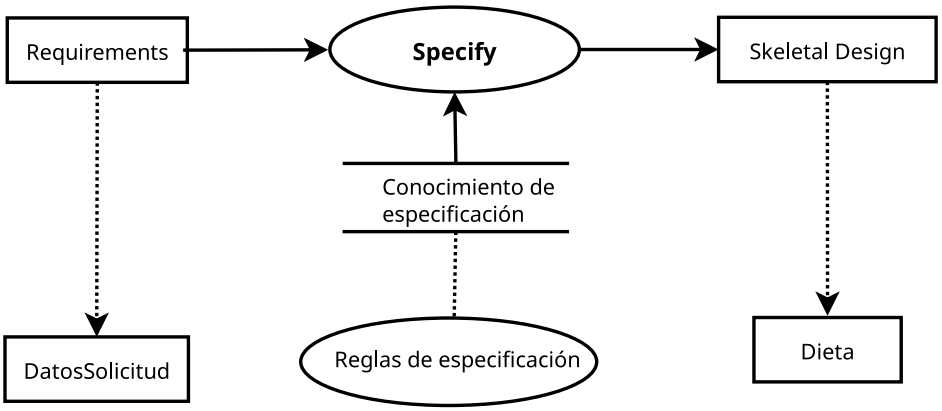

Figura 4: Inferencia Specify

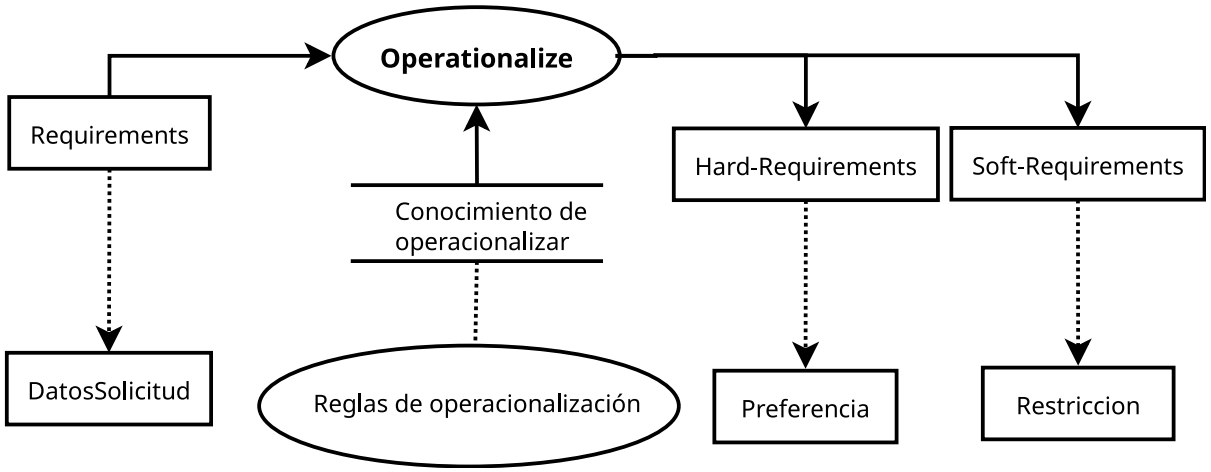

Figura 5: Inferencia Operationalize

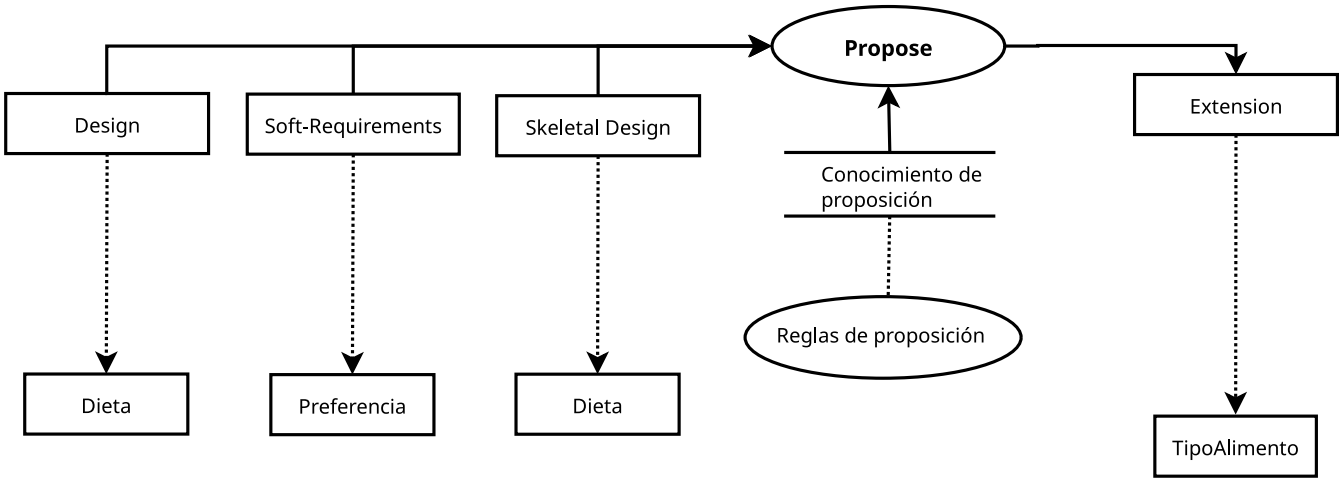

Figura 6: Inferencia Propose

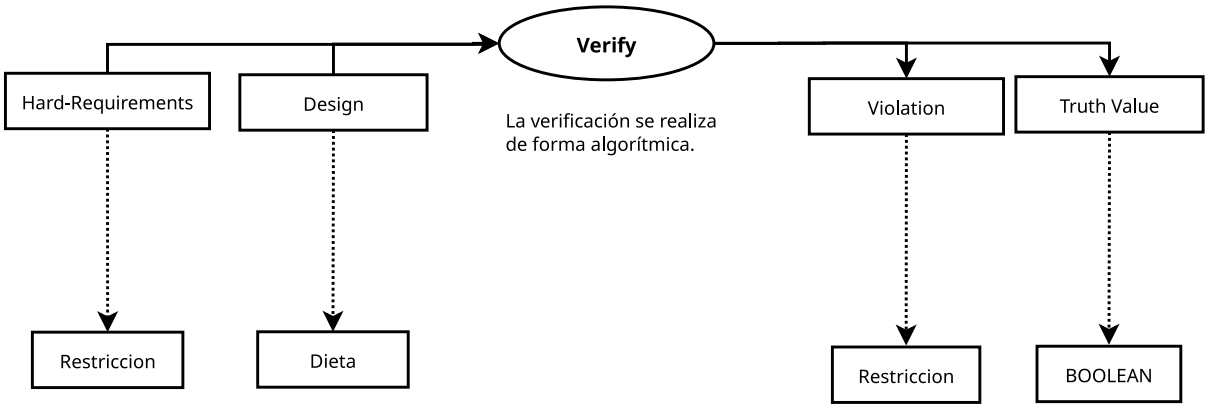

Figura 7: Inferencia Verify

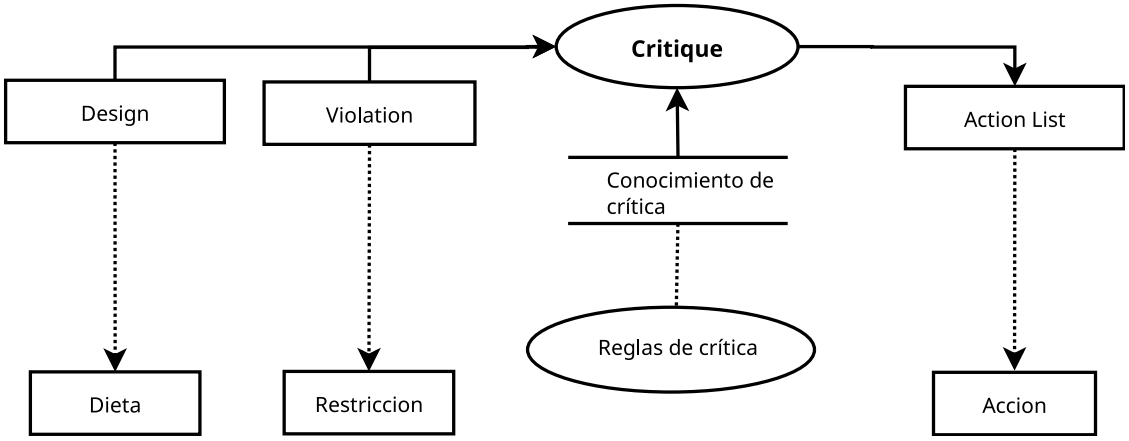

Figura 8: Inferencia Critique

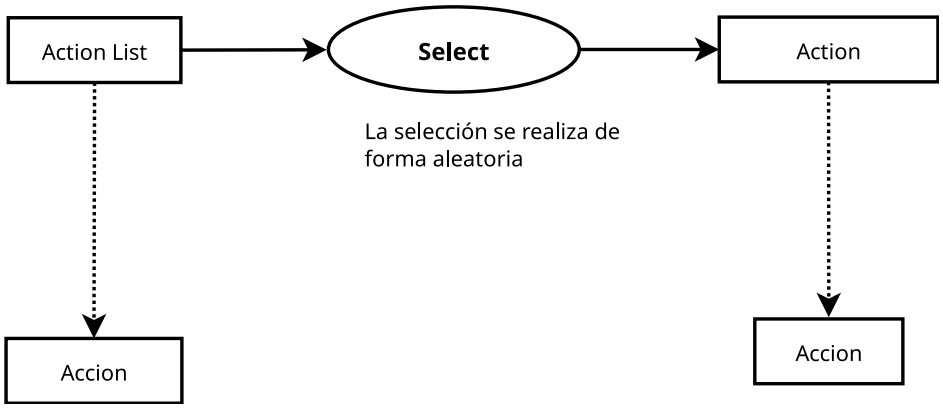

Figura 9: Inferencia Select

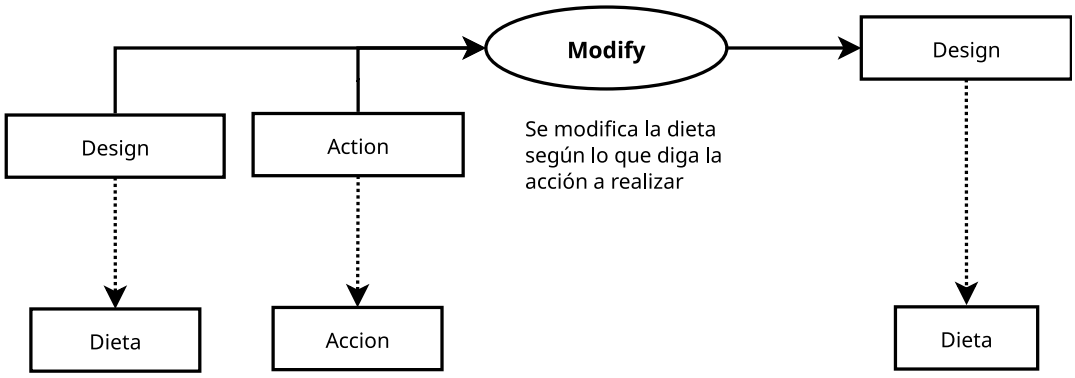

Figura 10: Inferencia Modify

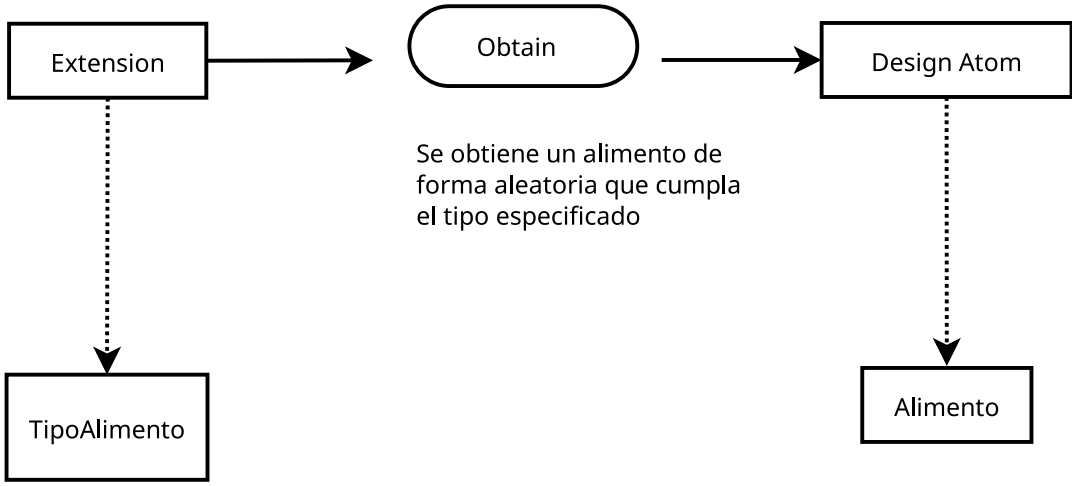

Figura 11: Función de transferencia Obtain

1.2.5. Tarea

```
TASK elaboracion-dieta;
    ROLES:
        INPUT: requirements: "Datos de la solicitud del cliente";
        OUTPUT: design: "La dieta resultante";
END TASK elaboracion-dieta;

TASK-METHOD propose-and-revise;
    REALIZES: elaboracion-dieta;
    DECOMPOSITION:
        INFERENCES: operationalize, specify, propose, verify, critique,
        select, modify;
        TRANSFERENCES: obtain;
    ROLES:
        INTERMEDIATE:
            soft-requirements: "Preferencias del usuario sobre alimentos";
            hard-requirements: "Restricciones que tiene que cumplir la dieta";
            skeletal-design: "Esqueleto de la dieta, solo contiene los slots vacios";
            violation: "Restriccion que incumple la dieta";
            truth-value: "Valor booleano que indica si la dieta es correcta";
            action-list: "Lista de acciones para poder reparar las restricciones que
            action: "Accion para reparar alguna de las restricciones que incumple la
            extension: "Tipo de alimento que se va a añadir en el slot";
            design-atom: "Alimento concreto que se va a añadir en el slot"
    CONTROL-STRUCTURE:
        operationalize(requirements -> hard-requirements
            + soft-requirements);
        specify(requirements -> skeletal-design);
        WHILE NEW-SOLUTION propose(skeletal-design + design
                + soft-requirements -> extension) DO

            obtain(extension -> design-atom)
            design := design ADD design-atom;

            verify(design + hard-requirements
                -> truth-value + violation);
            IF truth-value == false
            THEN
                critique(violation + design -> action-list);
                REPEAT
                    select(action-list -> action);
                    modify(design + action -> design);
                    verify(design + hard-requirements
```

```
                        -> truth-value + violation);
                UNTIL truth-value == true;
                END REPEAT
            END IF
        END WHILE
END TASK-METHOD propose-and-revise;
```

1.2.6. Esquema del dominio final

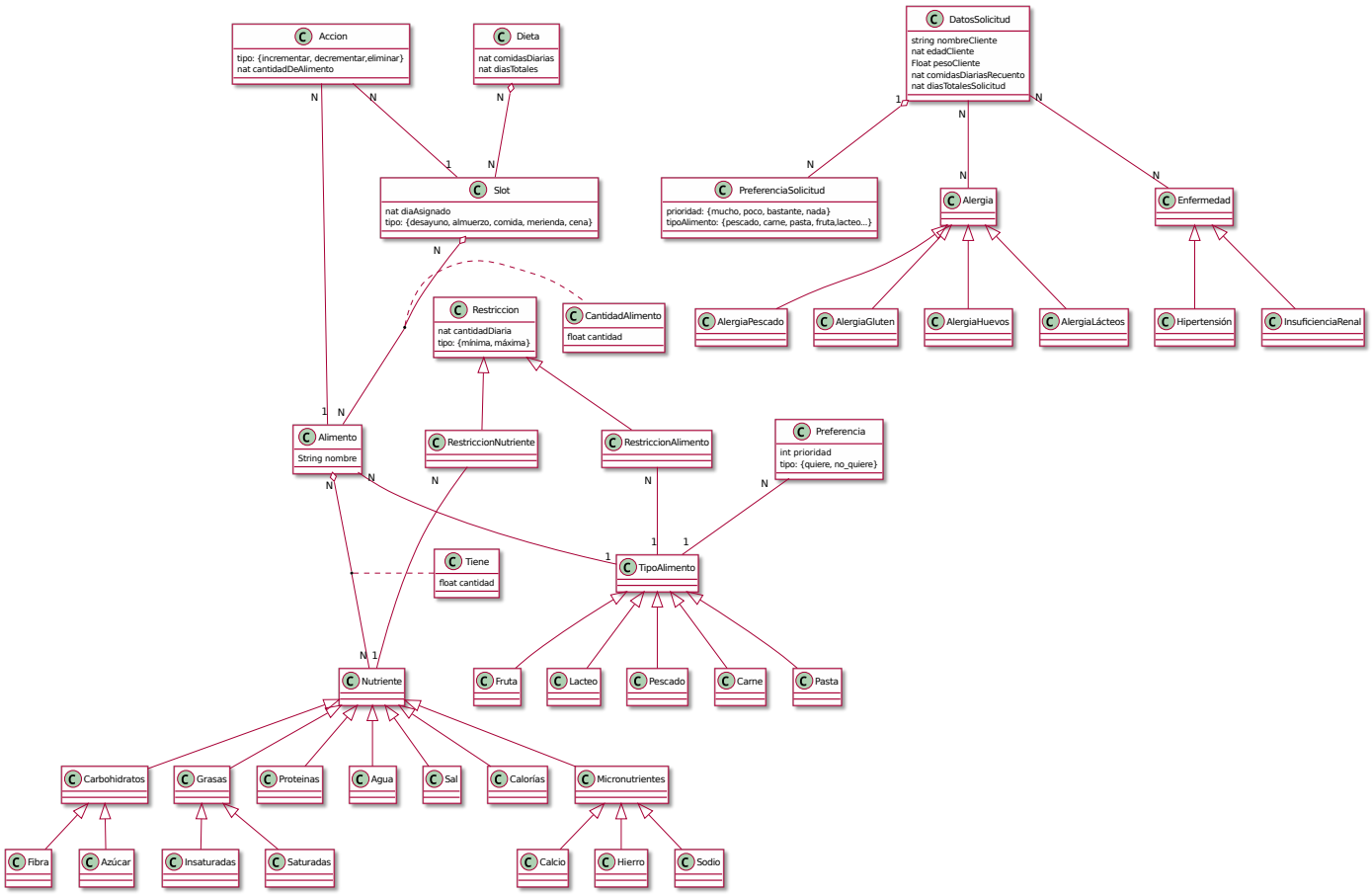

Figura 12: Esquema del dominio final

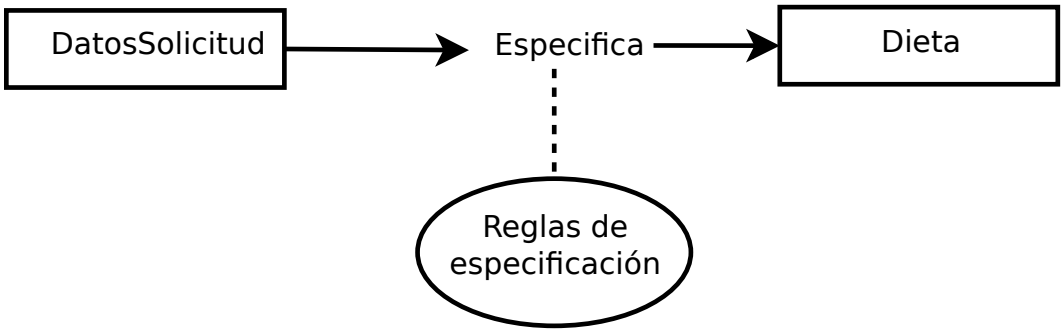

Figura 13: Reglas de especificación

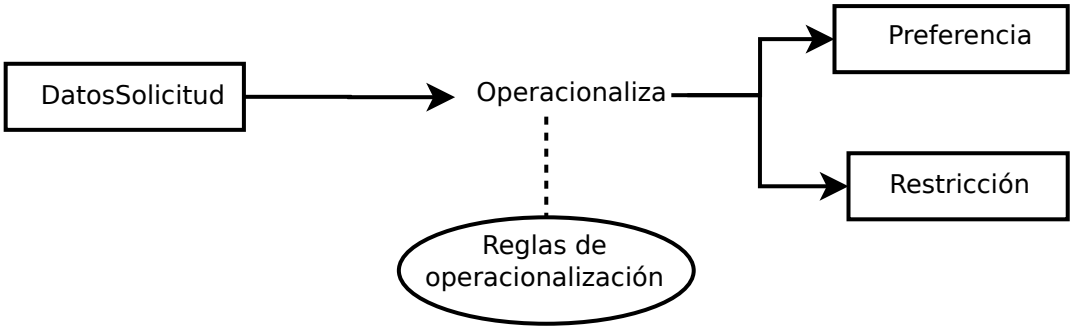

Figura 14: Reglas de operacionalización

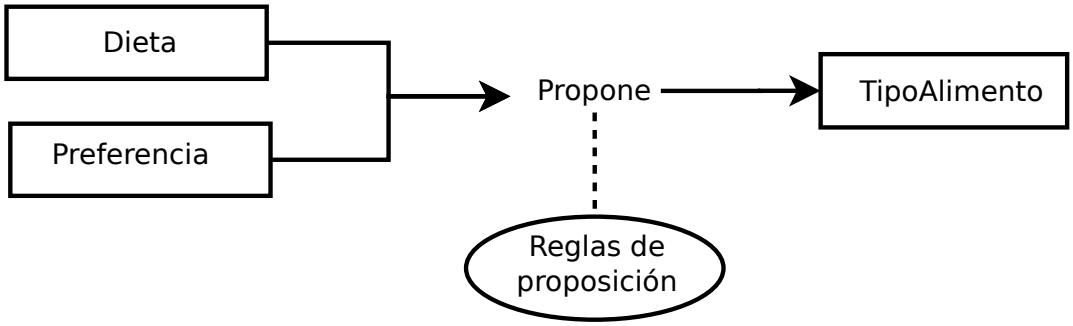

Figura 15: Reglas de proposición

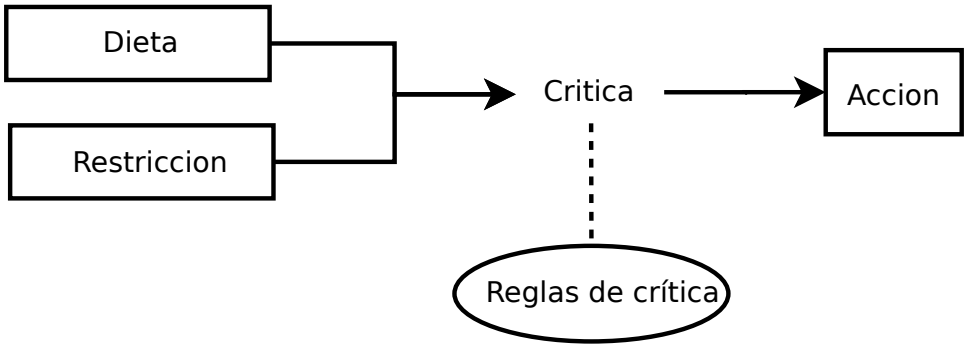

Figura 16: Reglas de crítica

1.3. Fase De Refinamiento

1.3.1. Validación

- **Paso 1**

  - **Interacción Usuario-Sistema**: Juan, paciente con problemas de riñón y alergia al pescado, insuficiencia renal. Hombre, 64 años, 85kg. El recuento de los últimos días indica que suele realizar 3 comidas diarias. Le gusta mucho la pasta. Quiere una dieta para 2 días.
  - **Comportamiento del modelo**:
    DatosSolicitud.nombreCliente = Juan
    DatosSolicitud.edadCliente = 64 años
    DatosSolicitud.pesoCliente = 85 kg
    DatosSolicitud.diasTotalesSolicitud = 2
    DatosSolicitud.comidasDiariasRecuento = 3
    DatosSolicitud.Enfermedad = InsuficienciaRenal
    DatosSolicitud.Alergia = AlergiaPescado

  - **Explicación**: Se guarda en DatosSolicitud toda la información que entra al sistema.

- **Paso 2**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Operationalize
    - Preferencia.tipo = quiere
      Preferencia.prioridad = 1
      Preferencia.tipoAlimento = Pasta
    - RestriccionNutriente.tipo = mínima
      RestriccionNutriente.cantidadDiaria = 2L
      RestriccionNutriente.Nutriente = Agua
    - RestriccionNutriente.tipo = mínima
      RestriccionNutriente.cantidadDiaria = 63.75gr
      RestriccionNutriente.Nutriente = Proteína
    - RestriccionNutriente.tipo = máxima
      RestriccionNutriente.cantidadDiaria = 85gr
      RestriccionNutriente.Nutriente = Proteína
    - RestriccionNutriente.tipo = máxima
      RestriccionNutriente.cantidadDiaria = 2gr
      RestriccionNutriente.Nutriente = Sodio
    - RestriccionNutriente.tipo = máxima
      RestriccionNutriente.cantidadDiaria = 2.5gr
      RestriccionNutriente.Nutriente = Potasio
    - RestriccionNutriente.tipo = mínima
      RestriccionNutriente.cantidadDiaria = 2550kcal
      RestriccionNutriente.Nutriente = Calorías
    - RestriccionNutriente.tipo = máxima
      RestriccionNutriente.cantidadDiaria = 3400kcal
      RestriccionNutriente.Nutriente = Calorías
    - RestriccionAlimento.tipo = máxima
      RestriccionAlimento.cantidadDiaria = 0
      RestriccionAlimento.TipoAlimento = Pescado
  - **Explicación**: El sistema, mediante la inferencia *Operationalize*, genera las preferencias (soft-requirements) y las restricciones (hard-requirements) a partir de la información almacenada en DatosSolicitud, aplicando reglas con conocimiento lógico y variable en función de dicha información.

- **Paso 3**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Specify
    - Dieta.comidasDiarias = 3
      Dieta.diasTotales = 2
    - Dieta.Slots[1].diaAsignado = 1
      Dieta.Slot.tipo = desayuno
      Dieta.Slot.Alimentos = [ ]
    - Dieta.Slots[2].diaAsignado = 1
      Dieta.Slot.tipo = comida
      Dieta.Slot.Alimentos = [ ]
    - Dieta.Slot[3].diaAsignado = 1
      Dieta.Slot.tipo = cena
      Dieta.Slot.Alimentos = [ ]
    - ...
  - **Explicación**: Se ejecuta la inferencia *Specify* y se crea un esqueleto de una dieta con los slots vacíos.
- **Paso 4**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Propose
    - TipoAlimento = Fruta
  - **Explicación**: Primero, la inferencia *Propose*, trata el Slot del día 1 que es de tipo desayuno. Como el usuario no tiene niguna preferencia sobre frutas o lácteos se selecciona uno de los dos aleatoriamente, ya que en el desyuno es obligatorio que haya al menos una fruta o un lácteo.
- **Paso 5**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Obtain
    - Alimento = Manzana

    - Dieta.Slots[1].Alimentos = [Manzana (100gr)]
  - **Explicación**: El sistema accede a la base de datos y obtiene un alimento aleatorio a partir del TipoAlimento que le llega como entrada a la inferencia *Obtain*. Posteriormente, se añade el alimento recuperado al slot que se está tratando.

- **Paso 6**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Verify
    - TruthValue = False
    - RestriccionNutriente.tipo = mínima
      RestriccionNutriente.cantidadDiaria = 2550kcal
      RestriccionNutriente.Nutriente = Calorías
  - **Explicación**: En la inferencia *Verify*, el sistema verifica la configuración actual de la dieta y devuelve que dicha configuración no es correcta, junto a una restricción que incumple.

- **Paso 7**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Critique
    - Accion.tipo = incrementar
      Accion.cantidadDeAlimento = 100gr
      Accion.Slot = Dieta.Slots[1]
      Accion.Alimento = Manzana
  - **Explicación**: En la inferencia *Critique*, el sistema elabora una lista de acciones en base a la restricción incumplida. En este caso, se puede aplicar una única acción ya que solo existe un alimento en el slot.

- **Paso 8**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Select
    - Accion = ActionList[1]
      (Accion.tipo = incrementar
      Accion.cantidadDeAlimento = 100gr
      Accion.Slot = Dieta.Slots[1]
      Accion.Alimento = Manzana)

  - **Explicación**: En *Select*, se elige aleatoriamente una de las acciones de la lista de entrada, que será la que se aplique a la dieta. En este ejemplo, ya que la lista consta de una sola acción, será esa la que se escoja.

- **Paso 9**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Modify
    - Dieta.Slots[1].Alimentos = [Manzana (200gr)]
  - **Explicación**: Se modifica el slot de la dieta indicado por la acción. En este caso, la cantidad del alimento *Manzana* aumenta en 100gr.
- **Paso 10**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Propose
    - TipoAlimento = Pescado
  - **Explicación**: En la segunda iteración, la inferencia *Propose* recomendará un tipo de alimento para el siguiente slot de la dieta. Como en nuestro caso de ejemplo el cliente realiza 3 comidas al día, el slot será de tipo *comida*. Además, y a pesar de que exista una preferencia de *Pasta*, como en nuestro sistema sigue existiendo un componente aleatorio a la hora de elegir el tipo de alimento, este acaba recomendando *Pescado*.
- **Paso 11**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Obtain
    - Alimento = Salmón
    - Dieta.Slots[1].Alimentos = [Salmón (100gr)]
  - **Explicación**: El sistema selecciona aleatoriamente un alimento de tipo *Pescado* de la base de datos.
- **Paso 12**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Verify

- TruthValue = False
- RestriccionNutriente.tipo = máxima
  RestriccionNutriente.cantidadDiaria = 0
  RestriccionNutriente.Nutriente = Pescado

- **Explicación**: Como nuestro cliente es alérgico al pescado, *Verify* devuelve la restricción que establece que no puede tomar pescado.

- **Paso 13**

  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Critique

    - Accion.tipo = eliminar
      Accion.Slot = Dieta.Slots[2]
      Accion.Alimento = Pescado

  - **Explicación**: Como la restricción hace referencia al *Pescado*, y la restricción indica que la cantidad diaria máxima es de 0, la única acción que genera *Critique* es la de eliminar dicho alimento.

- **Paso 14**

  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Select

    - Accion = ActionList[1]
      (Accion.tipo = eliminar
      Accion.Slot = Dieta.Slots[2]
      Accion.Alimento = Pescado)

  - **Explicación**: Se vuelve a elegir la única acción posible.

- **Paso 15**

  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Modify

    - Dieta.Slots[2].Alimentos = [ ]

  - **Explicación**: Se aplica la acción sobre el slot especificado, eliminando el alimento *Pescado*.

- **Paso 16**

- **Interacción Usuario-Sistema**: N/A
- **Comportamiento del modelo**:
  Propose
  - TipoAlimento = Pasta
- **Explicación**: Ahora, la inferencia volverá a proponer un alimento para la comida, ya que no hemos podido añadirlo anteriormente. A diferencia de la iteración anterior, nuestro sistema recomendrá el tipo de alimento *Pasta* (tras la selección aleatoria), siguiendo la preferencia del cliente. Aunque la selección fue aleatoria, al ser la *Pasta* una preferencia del cliente, se le da una mayor probabilidad de ocurrencia.

- **Paso 17**
  - **Interacción Usuario-Sistema**: N/A
  - **Comportamiento del modelo**:
    Obtain
    - Alimento = Espaguetis
    - Dieta.Slots[2].Alimentos = [Espaguetis (100gr)]
  - **Explicación**: El sistema selecciona aleatoriamente un alimento de tipo *Pasta* de la base de datos.

A partir de este punto, se seguiría generando la dieta hasta que ninguna restricción fuese incumplida, de una forma similiar a los pasos especificados anteriormente.

1.3.2. Bases de conocimientos

Reglas de especificación

- SI DatosSolicitud.edadCliente <= 12 O DatosSolicitud.edadCliente >= 80 ESPECIFICA Dieta.comidasDiarias = 5
- SI DatosSolicitud.edadCliente > 12 Y DatosSolicitud.edadCliente < 65 Y DatosSolicitud.comidasDiariasRecuento = n ESPECIFICA Dieta.comidasDiarias = max(3, n)
- SI DatosSolicitud.edadCliente >=65 Y DatosSolicitud.edadCliente < 80 Y DatosSolicitud.comidasDiariasRecuento = n ESPECIFICA Dieta.comidasDiarias = max(4, n)

- SI DatosSolicitud.diasTotalesSolicitud = n ESPECIFICA Dieta.diasTotales = n Y Dieta.Slot.diaAsignado <= n

Reglas de operacionalización

A: Indica el tipo de alimento; se utiliza esto en vez de poner el tipo concreto para comprimir las reglas.

- SI PreferenciaSolicitud.prioridad = mucho OPERACIONALIZA Preferencia.tipo = quiere Y Preferencia.prioridad = 1
- SI PreferenciaSolicitud.prioridad = nada OPERACIONALIZA Preferencia.tipo = no_quiere Y Preferencia.prioridad = 1
- SI PreferenciaSolicitud.tipoAlimento = *A* OPERACIONALIZA Preferencia.TipoAlimento = *A*
- SI DatosSolicitud.Enfermedad = Hipertensión OPERACIONALIZA RestriccionNutriente.{tipo, cantidadDiaria, Nutriente} = {máxima, 5g, Sal}
- SI DatosSolicitud.edadCliente >= 14 OPERACIONALIZA RestriccionNutriente.{tipo, cantidadDiaria, Nutriente} = {mínima, 2L, Agua}
- SI DatosSolicitud.Enfermedad = InsuficienciaRenal OPERACIONALIZA RestriccionNutriente.{tipo, cantidadDiaria, Nutriente} = {mínima, 0.75g * DatosSolicitud.pesoCliente, Proteína}
- SI DatosSolicitud.Enfermedad = InsuficienciaRenal OPERACIONALIZA RestriccionNutriente.{tipo, cantidadDiaria, N putriente} = {mínima, 30kcal * DatosSolicitud.pesoCliente, Calorías}
- SI DatosSolicitud.Alergia = AlergiaPescado OPERACIONALIZA RestriccionAlimento.{tipo, cantidadDiaria, TipoAlimento} = {máxima, 0, Pescado}

Reglas de proposición

Explicación de la notación a utilizar:

- seleccionar(p, conj1, conj2): Indica que se seleccionará aleatoriamente un tipo de alimento del conj1 con una probabilidad de p, o un alimento del conj2, con probabilidad de 1-p. Esto se hace para poder modelar que algunas veces se seleccione un alimento u otro, pero dando más probabilidad de aparición a los alimentos que gusten más, y menos probabilidad a los que gusten menos.

- seleccionarAleatorio(conj): Indica que se seleccionará aleatoriamente de forma equiprobable un tipo de alimento del conjunto.
- *A*: Indica un tipo de alimento concreto, para poder comprimir las reglas y que no se expandan mucho. Podría sustituírse por valores concretos.

Las primeras dos reglas sirven para que en la comida y en la cena haya al menos una vez pasta, carne o pescado, teniendo en cuenta primero las preferencias. Las otras dos reglas sirven para que en el desayuno, merienda y almuerzo haya al menos una vez un lacteo o una fruta.

- SI Dieta.Slot.tipo ES {comida, cena} Y Dieta.Slot.alimentos.tipoAlimento NO HAY {Pescado, Carne, Pasta} Y Preferencia.{tipo, tipoAlimento, prioridad} = {quiere, *A*, 1} Y *A* ES {Carne, Pasta, Pescado} PROPONE TipoAlimento = seleccionar(0.6, {*A*}, {Carne, Pasta, Pescado} - A)
- SI Dieta.Slot.tipo ES {comida, cena} Y Dieta.Slot.alimentos.tipoAlimento NO HAY {Pescado, Carne, Pasta} Y NO HAY Preferencia.tipoAlimento = {Carne, Pasta, Pescado} PROPONE TipoAlimento = seleccionarAleatorio({Pasta, Carne, Pescado})
- SI Dieta.Slot.tipo ES {desayuno, almuerzo, merienda} Y Preferencia.{tipo, tipoAlimento, prioridad} = {no_quiere, Fruta, 1} Y Dieta.Slot.alimentos.tipoAlimento NO HAY {Fruta, Lácteo} PROPONE TipoAlimento = Lácteo
- SI Dieta.Slot.tipo ES {desayuno, almuerzo, merienda} Y Dieta.Slot.alimentos.tipoAlimento NO HAY {Fruta, Lácteo} Y NO HAY Preferencia.tipoAlimento = {Fruta, Lácteo} PROPONE TipoAlimento = seleccionarAleatorio({Fruta, Lácteo})

Reglas de Crítica

- SI Restriccion.tipo = máxima y Restriccion.cantidadDiaria > 0 CRITICA Accion.tipo = decrementar
- SI Restriccion.tipo = máxima y Restriccion.cantidadDiaria = 0 CRITICA Accion.tipo = eliminar
- SI Restriccion.tipo = mínima CRITICA Accion.tipo = incrementar
- SI RestriccionAlimento.tipoAlimento = Fruta Y RestriccionAlimento.cantidadDiaria > 0 CRITICA Accion.cantidadDeAlimento = 100gr

Si la restricción es sobre un alimento, la acción se ejecutará sobre ese alimento del Slot. Si la restricción es sobre un nutriente, entonces se ejecutará sobre un alimento aleatorio del Slot.

Inferencia Select Para la inferencia *Select* no serán necesarias reglas; Se elegirá una acción de forma aleatoria, ya que a priori, no podemos saber cuál va a tener mejores efectos sobre el resultado.

Inferencia Verify La verificación de una dieta se puede realizar de forma algorítmica, se tendría que recorrer todos los slots de la dieta y comprobar que, para cada conjunto de slots que se refieran al mismo día, no se incumpla ninguna restricción sobre un alimento o una dieta. Si alguna restricción se incumple, entonces el *Truth Value* será falso y se devolverá dicha restricción a la salida de *Violation*. En caso de que no se incumpla ninguna restricción, *Truth Value* sería verdadero.

Inferencia Modify La modificación también se realiza de forma algorítmica; simplemente aplica sobre la dieta la acción seleccionada.

1.4. Plan de comunicación general

Primero, el usuario comunica al sistema los datos de la solicitud. Después, el sistema especifica los requisitos que tiene que cumplir la dieta y el esqueleto de la dieta. A continuación, el sistema propone un tipo de alimento a incluír y recuperará un alimento de ese tipo de la base de datos. Finalmente, si la dieta generada es correcta, la devuelve al usuario.

- Diagrama de diálogo:

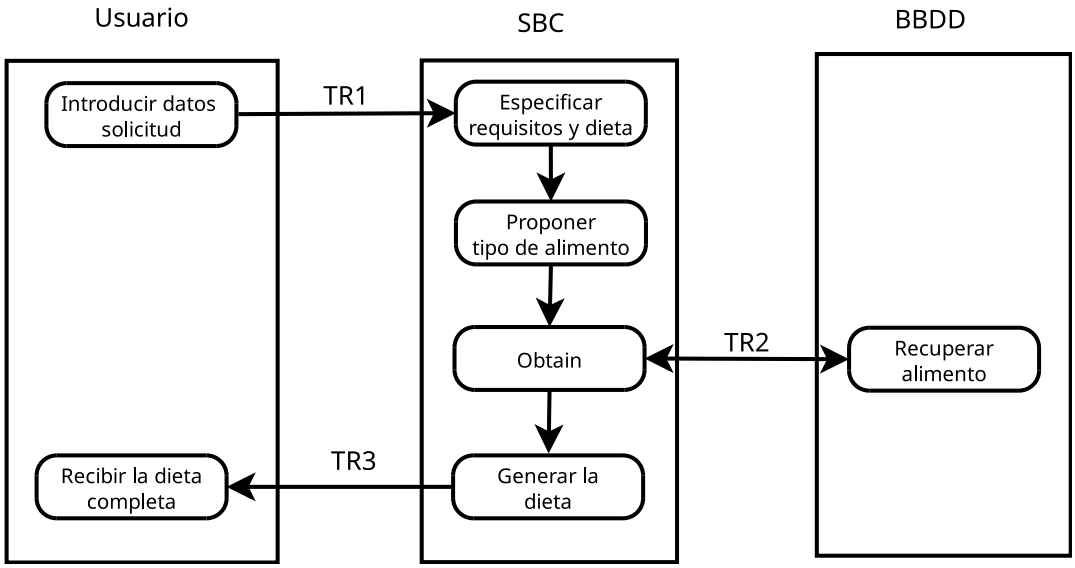

Figura 17: Diagrama de diálogo

- Información de control de las transacciones en forma de diagrama de transición:

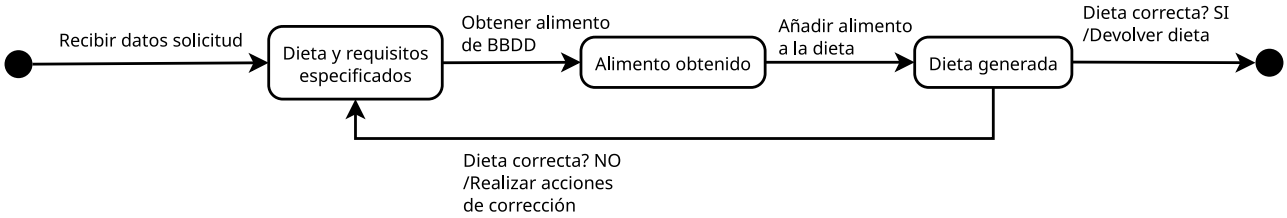

Figura 18: Diagrama de transición

1.5. Descripción de las Transacciones

Formularios CM-1 para cada una de las transacciones identificadas en la Figura 17.

Nombre de la transacción	TR1: Introducir datos de la solicitud en el sistema
Objetos de información	Los datos de la solicitud del paciente
Agentes involucrados	Emisor: Usuario Receptor: SBC
Plan de comunicaciones	Ver figuras 17 y 18
Restricciones	Para iniciar la transacción, el SBC debe estar listo
Especificación del intercambio de información	Es del tipo ORDER

Nombre de la transacción	TR2: Obtener alimento del tipo deseado de BBDD
Objetos de información	El tipo de alimento y el alimento concreto a añadir en la dieta
Agentes involucrados	Emisor: SBC Receptor: BBDD
Plan de comunicaciones	Ver figuras 17 y 18
Restricciones	El SBC debe estar listo y los alimentos concretos deben estar disponibles en la BBDD
Especificación del intercambio de información	Es del tipo REQUIRE

Nombre de la transacción	TR3: Devolver dieta elaborada
Objetos de información	La dieta completa
Agentes involucrados	Emisor: SBC Receptor: Usuario
Plan de comunicaciones	Ver figuras 17 y 18
Restricciones	La dieta elaborada tiene que ser correcta
Especificación del intercambio de información	Es del tipo REPORT

1.6. Especificación de las transacciones

Los modelos de comunicación son sencillos, por lo que no es necesario realizar el formulario CM-2