Modeling Insurance Products with DMN

wilfried.kurth@axa-winterthur.ch

DecisionCamp 2020

Modeling Insurance Products with DMN

Content

1. General Insurance Conditions (GIC)
   - → From continuous text
   - → To structured form
2. DMN Decision Table Approach
   - → Business Knowledge Model (BKM) with a Decision Table
   - → Limits of the Decision Table Approach
3. DMN Relation Approach
   - → Business Logic for Determining required Claims Input
   - → Business Logic for Determining Benefit Promise
4. What DMN is good for
   - → Process Automation
   - → For all Kind of Business Logic!

General Insurance Conditions (GIC)

Benefit Promise as continuous Text

- Structuring continuous text:
  - → Determine properties and corresponding values

1) **What goods are insured?**
The insurance covers the household contents (…).

Household contents are defined as all movable objects for personal use (…).

The household contents also include (…) property belonging to guests (not including money assets).

2) **What risks are insured?**
Fire
Natural Forces
Water
Simple Theft
Burglary
Robbery

3) **Where does the insurance apply?**
At home, i. e. at the locations given in the policy (…).

Elsewhere worldwide for household contents kept temporarily at any other location (…).

4) **What compensation is insured?**
Household contents are insured at replacement value up to the (…) sum insured.

Money assets include: cash, traveler checks, (…).
a. Compensation is restricted to CHF 5'000 for normal storage
b. Compensation is restricted to CHF 20'000 for storage in safety container (…)
c. The insurance does not cover money assets against simple theft at home or outwards
d. The insurance does not cover money assets kept in movable structures

The compensation for simple theft out-and-about is restricted to the agreed sum insured.

Object Category
Usage
Ownership
Claim event
Place
Max. sum
Object Category
Safekeeping
Storage

Structured GIC

Benefit Promise in tabular Form

Properties

Values from the GIC

	Claim							Compensation	
Product module	**Object category**	**Claim event**	**Usage**	**Ownership**	**Place**	**Storage**	**Safekeeping**	**Max. sum**	**max. percent**
Standard	household contents	fire, natural forces, water, burglary, robbery	personal					insured sum	100 % of insured sum
Standard	household contents	simple theft	personal		at home			insured sum	100 % of insured sum
Standard	household contents	simple theft			outwards			0	0
Standard	money assets	fire, natural forces, water, burglary, robbery	personal	owned		building, carry along	normal	5'000	100 % of insured sum
Standard	money assets	fire, natural forces, water, burglary, robbery	personal	owned		building	cash box	20'000	100 % of insured sum
Standard	money assets	simple theft						0	0
Standard	money assets		on business					0	0
Standard	money assets			third party				0	0
Standard	money assets					vehicle		0	0

covered

excluded

covered

excluded

Empty cells: there is no value or respectively any value is possible
Several values per cell: values apply alternatively

GIC as DMN Decision Table

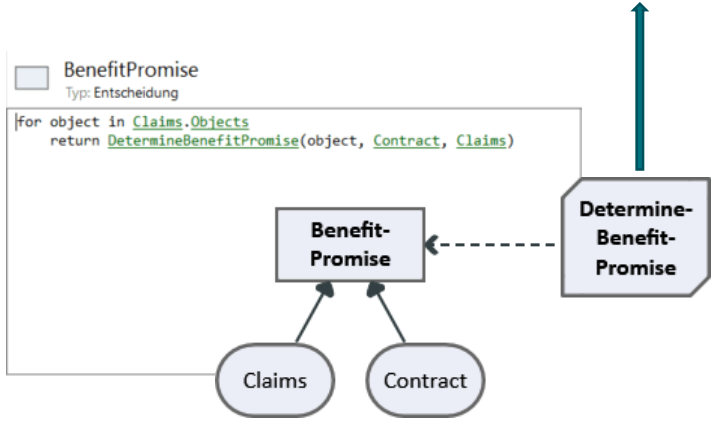
Determine benefit promise in case of damage

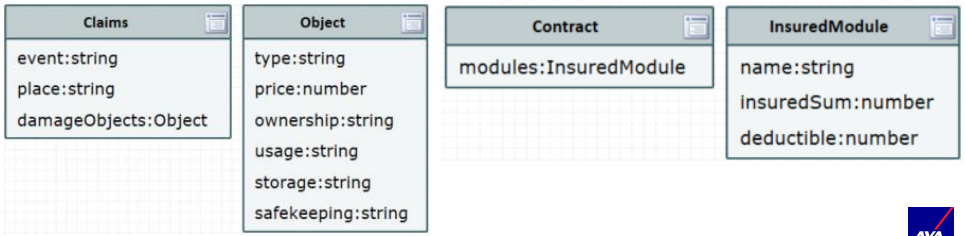
- A claim may have many claim objects
  - → A statement for every object is needed
  - => Iterative BKM invocation

- A contract may have many product modules
  - → Every module may provide coverage for a claims
  - => list contains function for contract module
  - => filter function for insured sum

The decision table approach works only for this one use case!

Enlargement on slide 12





Limits of the Decision Table Approach

Applying different logic based on the same representation of the GIC

- GIC as decision table => only one conclusion can be drawn:
  - → Question: What is the benefit promise in case of damage according to the GIC?
  - → Answer: Compensation with max. sum and max. percent

- What if an additional conclusion is needed?
  - → Question: What information is required to report a household-claim?
  - → Answer: List of attributes with its possible values

- The facts of the GIC have to be depict in a form that allow different queries:
  - → Use Case 1: Determining the required information
  - → Use Case 2: Determining the benefit promise

GIC as DMN Relation

1. Determine attributes and values

- The GIC are represented as a relation
- There is a separate BKM which provides the decision logic for this first use case
- The decision logic is only invoked once

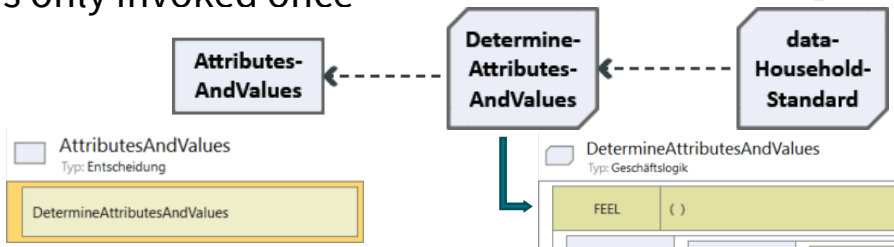
dataHouseholdStandard
Typ: Geschäftslogik

Enlargement on slide 14

FEEL (insuredSum)

GIC

	module	objectType	event	usage	ownership	place	storage	safekeeping	maxSum	maxPercent
1	"standard"	"household contents"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	null	null	null	null	insuredSum	100
2	"standard"	"household contents"	["simple theft"]	"personal"	null	["at home"]	null	null	insuredSum	100
3	"standard"	"household contents"	["simple theft"]	null	null	["outwards"]	null	null	0	0
4	"standard"	"money assets"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	["owned"]	null	["building", "carry along"]	"normal"	5000	100
5	"standard"	"money assets"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	["owned"]	null	["building"]	"cash box"	20000	100
6	"standard"	"money assets"	["simple theft"]	null	null	null	null	null	0	0
7	"standard"	"money assets"	null	"on business"	null	null	null	null	0	0
8	"standard"	"money assets"	null	null	["third party"]	null	null	null	0	0
9	"standard"	"money assets"	null	null	null	null	["vehicle"]	null	0	0



DetermineAttributesAndValues
Typ: Geschäftslogik

Decision Logic

FEEL ( )

propertyList

modules
Modules

dataHouseholdStandard

insuredSum
number
null

```
get entries(modules[1]).key[item != "module" and item != "maxSum" and item != "maxPercent"]
```

valueLists

modules
Modules

dataHouseholdStandard

insuredSum
number
null

allValuesLists

```
for header in propertyList
    return for datensatz in modules
        return get value(datensatz,header)
```

```
for value in allValuesLists return distinct values(flatten(value[item != null]))
```

```
 *** AttributesAndValues ***
Evaluation status: SUCCEEDED
Result:
'{propertyList=[safekeeping, ownership, usage, place, storage, event, objectType],

  valueLists=[[normal, cash box],
              [owned, third party],
              [personal, on business],
              [at home, outwards],
              [building, carry along, vehicle],
              [fire, natural forces, water, burglary, robbery, simple theft],
              [household contents, money assets]]}'
```

GIC as DMN Relation

2. Determine benefit promise in case of damage

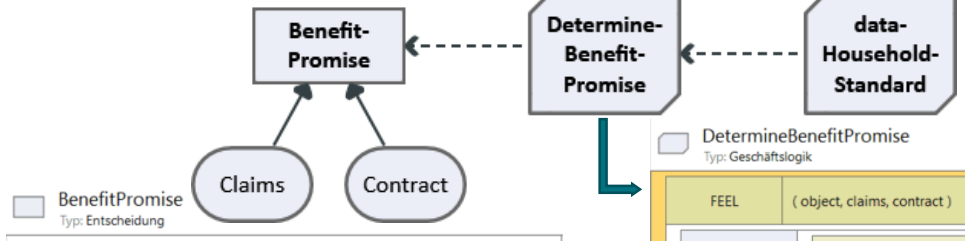
- The relation for the GIC is still the same
- There is a another BKM which provides the decision logic for this second use case
- The decision logic is invoked iteratively for each damaged object

dataHouseholdStandard
Typ: Geschäftslogik

FEEL (insuredSum)

GIC

	module	objectType	event	usage	ownership	place	storage	safekeeping	maxSum	maxPercent
1	"standard"	"household contents"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	null	null	null	null	insuredSum	100
2	"standard"	"household contents"	["simple theft"]	"personal"	null	["at home"]	null	null	insuredSum	100
3	"standard"	"household contents"	["simple theft"]	null	null	["outwards"]	null	null	0	0
4	"standard"	"money assets"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	["owned"]	null	["building", "carry along"]	"normal"	5000	100
5	"standard"	"money assets"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	["owned"]	null	["building"]	"cash box"	20000	100
6	"standard"	"money assets"	["simple theft"]	null	null	null	null	null	0	0
7	"standard"	"money assets"	null	"on business"	null	null	null	null	0	0
8	"standard"	"money assets"	null	null	["third party"]	null	null	null	0	0
9	"standard"	"money assets"	null	null	null	null	["vehicle"]	null	0	0



BenefitPromise
Typ: Entscheidung

```
for object in Claims.damageObjects
    return DetermineBenefitPromise(object, Claims, Contract)
```

Simulation with test data
➔ Slide 15

DetermineBenefitPromise
Typ: Geschäftslogik

FEEL (object, claims, contract) **Decision Logic**

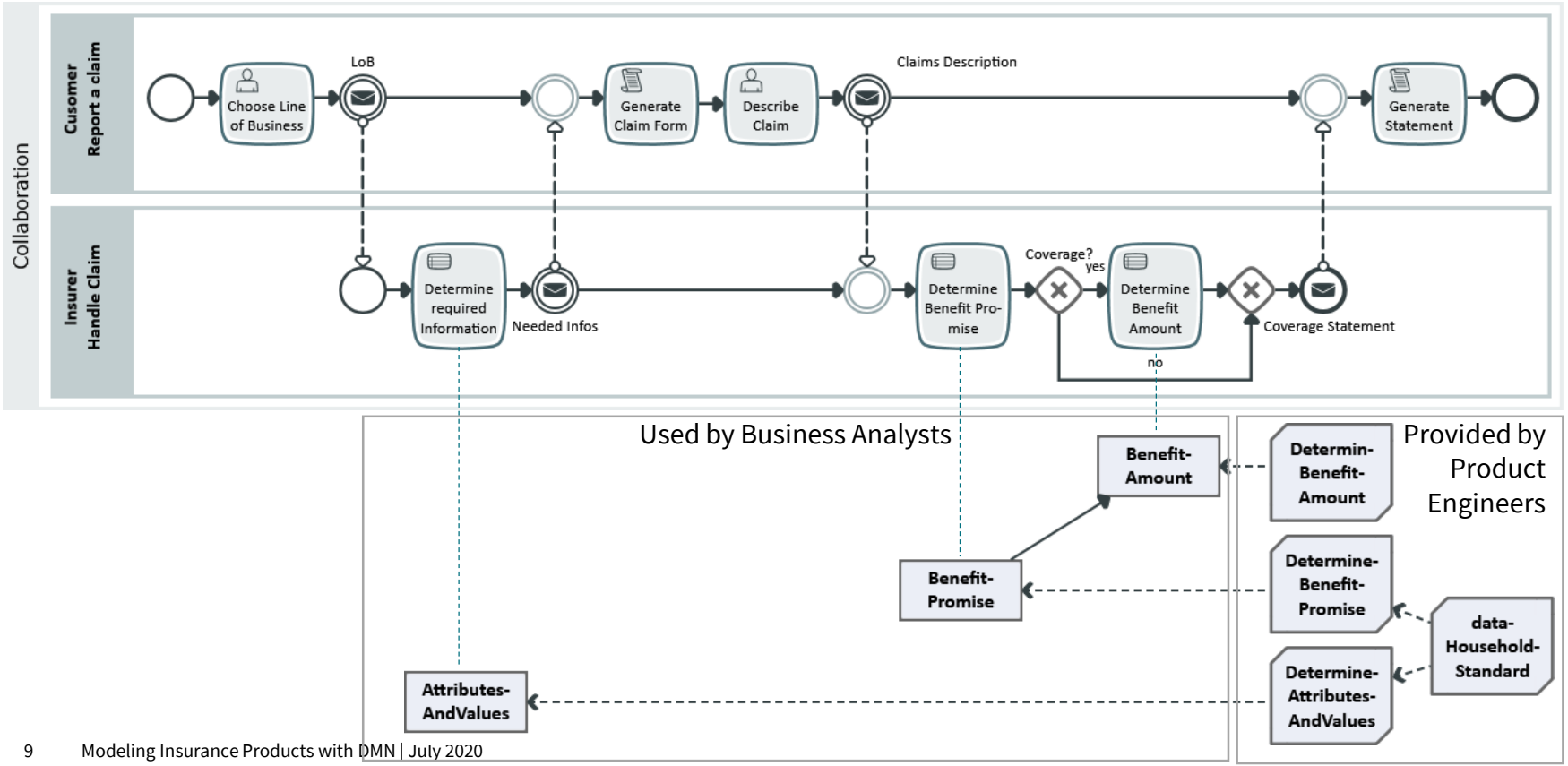
hhProduct *Modules*	dataHouseholdStandard
	insuredSum *number* : `contract.modules[name="standard"].insuredSum[1]`

benefitRow	`hhProduct[ list contains(contract.modules.name, module)`
	`and object.type = objectType`
	`and list contains(event, claims.event)`
	`and (usage = null or object.usage = usage)`
	`and (ownership = null or list contains(ownership, object.ownership))`
	`and (place = null or list contains(place, claims.place))`
	`and (storage = null or list contains(storage, object.storage))`
	`and (safekeeping = null or safekeeping = object.safekeeping)]`

benefitRow	

What DMN is good for

Process Automation (An examplary Claims Handling Process)



What DMN is good for

All kind of business logic!

- No more bother with missing business logic in business application systems
  - → DMN decisions fill the gaps – business logic is independent of systems
- No more fear of loss of business logic when systems are replaced
  - → DMN decisions stay with the Business
- No longer: First gathering requirements, then implementing systems
  - → Modeling processes, modeling decisions, running systems
    (ok, a little bit of coding still needs to be done)

Appendix

GIC as DMN Decision Table

DetermineBenefitPromise
Typ: Geschäftslogik

FEEL | (object, contract, claims)

Auswertungsstrategie: Eindeutig

	contract.Modules.name	object.typ	claims.event	object.usage	object.ownership	claims.place	object.storage	object.safekeeping	maxSum	maxPercent
1	list contains (?, "standard")	"household contents"	"fire", "natural forces", "water", "burglary", "robbery"	"personal"	-	-	-	-	contract.Modules[name="standard"].insuredSum	100
2	list contains (?, "standard")	"household contents"	"simple theft"	"personal"	-	"at home"	-	-	contract.Modules[name="standard"].insuredSum	100
3	list contains (?, "standard")	"household contents"	"simple theft"	-	-	"outward"	-	-	0	100
4	list contains (?, "standard")	"money assets"	"fire", "natural forces", "water", "burglary", "robbery"	"personal"	"owned"	-	"building", "carry along"	"normal"	5000	100
5	list contains (?, "standard")	"money assets"	"fire", "natural forces", "water", "burglary", "robbery"	"personal"	"owned"	-	"building"	"cash box"	20000	100
6	list contains (?, "standard")	"money assets"	"simple theft"	-	-	-	-	-	0	0
7	list contains (?, "standard")	"money assets"	-	"on business"	-	-	-	-	0	0
8	list contains (?, "standard")	"money assets"	-	-	"third party"	-	-	-	0	0
9	list contains (?, "standard")	"money assets"	-	-	-	-	"vehicle"	-	0	0

Decision Simulation and Result

Input Data: multi objects per claim and multi modules per contract

Claims

```
{
  "TestDataSets": [
    {
      "Name": "Testdatensatz",
      "Content": {
        "Claims": {
          "event [string]": "fire",
          "place [string]": "at home",
          "damageObjects": [
            {
              "type [string]": "household contents",
              "price [number]": 25000,
              "ownership [string]": "third party",
              "usage [string]": "personal",
              "storage [string]": "building",
              "safekeeping [string]": "normal",
              "count [number]": 2
            },
            {
              "type [string]": "money assets",
              "price [number]": 7900,
              "ownership [string]": "owned",
              "usage [string]": "personal",
              "storage [string]": "building",
              "safekeeping [string]": "normal",
              "count [number]": 1
            }
          ]
        }
      }
    }
  ]
}
```

Contract

```
{
  "TestDataSets": [
    {
      "Name": "Testcase: 2 Modules",
      "Content": {
        "Contract": {
          "Modules": [
            {
              "name [string]": "standard",
              "insuredSum [number]": 250000,
              "deductible [number]": 200
            },
            {
              "name [string]": "bike",
              "insuredSum [number]": 5000,
              "deductible [number]": 50
            }
          ]
        }
      }
    }
  ]
}
```

Simulation Results

```
*** HouseholdStandard ***
Evaluation status: SUCCEEDED
Result:
'[{maxPercent=100, maxSum=5000},
  {maxPercent=100, maxSum=[250000]}]'
================================
```

Back to slide 5

GIC as DMN Relation

dataHouseholdStandard
Typ: Geschäftslogik

FEEL | (insuredSum)

	module	objectType	event	usage	ownership	place	storage	safekeeping	maxSum	maxPercent
1	"standard"	"household contents"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	null	null	null	null	insuredSum	100
2	"standard"	"household contents"	["simple theft"]	"personal"	null	["at home"]	null	null	insuredSum	100
3	"standard"	"household contents"	["simple theft"]	null	null	["outwards"]	null	null	0	0
4	"standard"	"money assets"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	["owned"]	null	["building", "carry along"]	"normal"	5000	100
5	"standard"	"money assets"	["fire", "natural forces", "water", "burglary", "robbery"]	"personal"	["owned"]	null	["building"]	"cash box"	20000	100
6	"standard"	"money assets"	["simple theft"]	null	null	null	null	null	0	0
7	"standard"	"money assets"	null	"on business"	null	null	null	null	0	0
8	"standard"	"money assets"	null	null	["third party"]	null	null	null	0	0
9	"standard"	"money assets"	null	null	null	null	["vehicle"]	null	0	0

Back to slide 7

Decision Simulation and Result

Input Data: Multi objects per claim and multi modules per contract

Claims

```
{
  "TestDataSets": [
    {
      "Name": "Testdatensatz",
      "Content": {
        "Claims": {
          "event [string]": "fire",
          "place [string]": "at home",
          "damageObjects": [
            {
              "type [string]": "household contents",
              "price [number]": 25000,
              "ownership [string]": "third party",
              "usage [string]": "personal",
              "storage [string]": "building",
              "safekeeping [string]": "normal",
              "count [number]": 2
            },
            {
              "type [string]": "money assets",
              "price [number]": 7900,
              "ownership [string]": "owned",
              "usage [string]": "personal",
              "storage [string]": "building",
              "safekeeping [string]": "normal",
              "count [number]": 1
            }
          ]
        }
      }
    }
  ]
}
```

Contract

```
{
  "TestDataSets": [
    {
      "Name": "Testcase: 2 Modules",
      "Content": {
        "Contract": {
          "Modules": [
            {
              "name [string]": "standard",
              "insuredSum [number]": 250000,
              "deductible [number]": 200
            },
            {
              "name [string]": "bike",
              "insuredSum [number]": 5000,
              "deductible [number]": 50
            }
          ]
        }
      }
    }
  ]
}
```

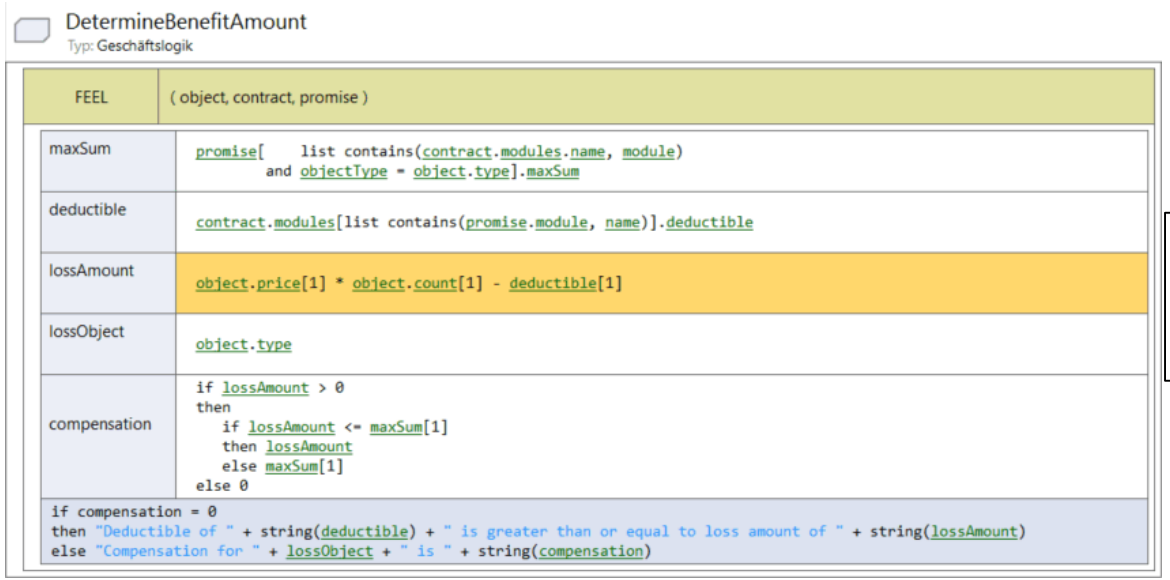
Simulation Results

```
*** BenefitPromise ***
Evaluation status: SUCCEEDED
Result:
'[{module=standard,
   objectType=household contents,
   event=[fire, natural forces, water, burglary, robbery],
   usage=personal,
   ownership=null,
   place=null,
   storage=null,
   safekeeping=null,
   maxSum=250000,
   maxPercent=100},

  {module=standard,
   objectType=money assets,
   event=[fire, natural forces, water, burglary, robbery],
   usage=personal,
   ownership=[owned],
   place=null,
   storage=[building, carry along],
   safekeeping=normal,
   maxSum=5000,
   maxPercent=100}]'
```

Back to slide 8

BKM DetermineBenefitAmount



DetermineBenefitAmount

Typ: Geschäftslogik

FEEL	(object, contract, promise)
maxSum	promise[list contains(contract.modules.name, module) and objectType = object.type].maxSum
deductible	contract.modules[list contains(promise.module, name)].deductible
lossAmount	object.price[1] * object.count[1] - deductible[1]
lossObject	object.type
compensation	if lossAmount > 0 then if lossAmount <= maxSum[1] then lossAmount else maxSum[1] else 0
if compensation = 0 then "Deductible of " + string(deductible) + " is greater than or equal to loss amount of " + string(lossAmount) else "Compensation for " + lossObject + " is " + string(compensation)	

```
 *** BenefitAmount ***
Evaluation status: SUCCEEDED
Result:

'[Compensation for household contents is 49800,
  Compensation for money assets is 5000]'
```

Structure Definitions