Validating a Validation Technology
Towards a Prototype Validation Experiment

Rainer Knauf

Technical University of Ilmenau
School of Computer Science and Automation
Ilmenau, Germany

Setsuo Tsuruta
Hirokazu Ihara

Tokyo Denki University
School of Information Environment
Tokyo, Japan

Torsten Kurbad

Knowledge Media Research Center
Tübingen, Germany

Avelino J.Gonzalez

University of Central Florida
Dept. of Electrical and Computer Engineering
Orlando, FL, USA

Validating a Validation Technology
Towards a Prototype Validation Experiment

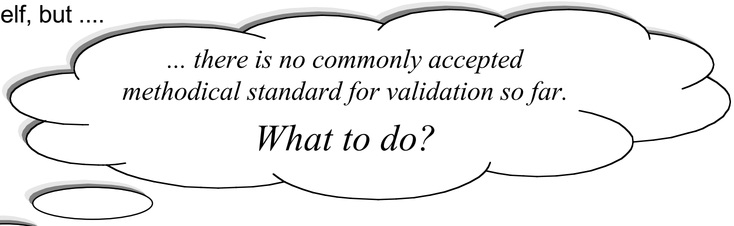
1. Motivation
2. The validation framework
   A short overview on the Turing Test
3. Basic concepts to involve validation knowledge
   An overview on *VKB* and *VESA*
4. The database structure of *TestMeToo*
5. The framework and the involvement of *VKB* and *VESA*
6. Towards a prototype application
7. Implementation Aspects
8. Summary, conclusion, actual and upcoming research and development

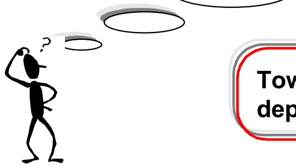
1 Motivation

Motivation for ***VKB*** and ***VESA*** ⇒ see PRDC slides

The trustfulness of an evaluation result corresponds to the dependability of the evaluation concept.

⇒ Validating validation concepts is at least as essential as validation of the systems itself, but ....





Towards such standard we need indications for the dependability of our methods and concepts!

2 The validation framework: A short overview on the TURING Test

Step # 1: **Test case generation**
Generate and optimize a set of test cases ***[test data , expected output]*** *that meets the competing requirements (1)* ***coverage*** *and (2)* ***efficiency***

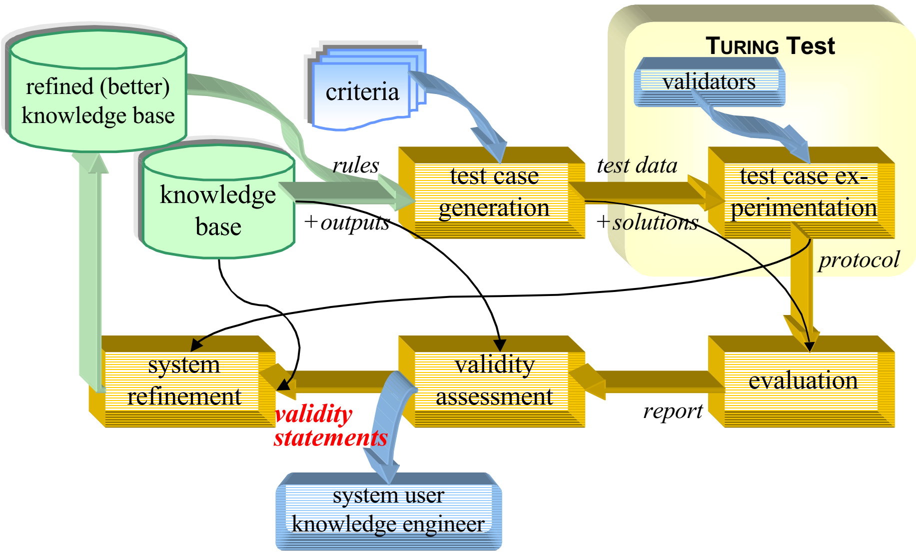
Step # 2: **Test case experimentation**
Exercise the test data by both the system under investigation and a panel of validating experts as a TURING Test - like experiment

Step # 3: **Evaluation**
Interpret experimentation results & report test case associated invalidities

Step # 4: **Validity assessment**
Analyze reported results and conclude validity assessments associated with (1) test cases, (2) outputs, (3) rules, and (4) the entire system

Step # 5: **System refinement**
Formally reconstructing the rule base so that it infers best rated solutions

The TURING Test validation framework so far



Unfortunately, there is a long list of general disadvantages and doubts

- *The quality of the results (validity statements and refined system) is highly influenced by the quality of current human input*
- *Human interaction is time consuming*
- *Human workload is expensive*
- *Human experts are not always available*
- *Human experts may not be willing when we want them to*
- *In this context, the commercial applicability of this technology might be doubtful*

Fortunately, there are our Japanese colleagues, who developed ideas to reduce this human factor

- *a Validation Knowledge Base (VKB) to model the „collective intelligence" of an expert panel*
- *a Validation Expert Software Agent (VESA) concept to model the „individual intelligence" of a certain expert*

3 Basic concepts to involve validation knowledge: *VKB* and *VESA*

Objective: Limit the workload of experts to break the limitation of dependability

3.1 TSURUTA's VKB Idea

knowledge transfer by sharing the experts' validation knowledge with

- other experts
- knowledge engineers
- computers

by

- memorizing a commented validation protocol and
- developing a Validation Knowledge Base from it.

The basic original idea is collecting collects validation (meta-) knowledge in a

- a ***Validation Data Base*** (***VDB***), which collects all upcoming data within the validation process and
- a ***Validation Knowledge Base*** (***VKB***), which analyzes, selects, pre-processes and memorizes relevant historical data of the VDB.

⇒ see PRDC slides

Here, we utilize this idea to model collective expertise of an expert panel.

3.2 TSURUTA's and UEHARA's *VESA* Idea

Objective

Modeling an individual expert's validation knowledge within a Validation Software Expert Agent (***VESA***)

Underlying basic assumptions

- Experts who provide similar solutions to test cases and similar ratings to other experts' solutions might have a similar „knowledge structure"
  - ⇒ A particular expert might be modeled by an agent that provides the response of another human expert, who had a maximum similarity to him in the past
- „knowledge structures" do change over time
  - ⇒ the degree of similarity depends on both
    - the ratio of same reflections (solutions, ratings) and
    - the „age" of this identical behavior

Here, we utilize this idea to model individual expertise of an expert

What is a *VESA*?

- autonomous software agent corresponding to a particular human expert
- gains personal validation knowledge mainly from personal data such as (not always best) solutions, ratings, etc. of its corresponding human expert validator
- can be considered to be a model that represents the validation experience and behavior of a group or an organization of validation experts

The learning issue of *VESA*

- learns from test inputs and the associated answers, their certainties and their ratings provided by the human validators
- increases its validation competence through validation knowledge gained by various sessions over time
- become more intelligent as well as more adaptive to wider (similar but slightly different) applications

Advantage of *VESA* over humans

- it also gains the validation knowledge of other validators
- Anonymity and impartiality is kept even if they get information from other experts: They do not need the name of each expert, but rather an *ID* to distinguish whether or not the information belongs to the same expert.

Software Agents vs. Validation Expert Software Agents (*VESA*)

- Generally, ***VESA*** adopts some basic concepts and assumptions of this idea from Software Technology
- In particular, advanced ideas like
    - the issue of learning,
    - the issue of cooperation and competition, and
    - the issue learning
        - about/from other agents and the world or
        - by communication and understanding

  are far away from the fundamental agent concept introduced here.

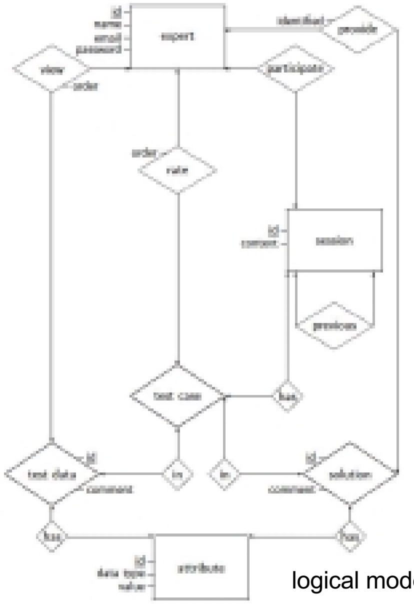
4 The database structure of *TestMeToo*

All available data: to each case

- the (input) test data t_j
- a list of all solvers E_K
- a list of all raters E_I
- associated optimal (best rated) solution sol_{Kj}^{opt}
- the ratings provided by the rating experts r_{IjK}
- the certainties of these ratings c_{IjK}
- a session time stamp τ_S
- an informal description of the context D_C

represented as an 8-tuple

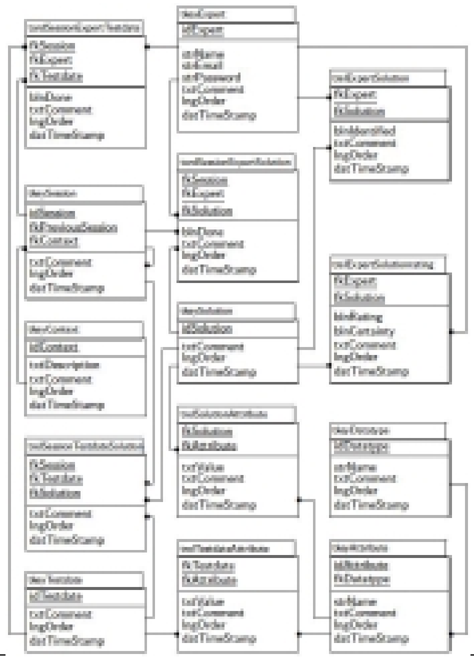
$$[\, t_j, E_K, E_I, sol_{Kj}^{opt}, r_{IjK}, c_{IjK}, \tau_S, D_C \,]$$



logical model

The physical database structure of *TestMeToo*

... looks like that:



5 The framework and the involvement of VKB and VESA

Sources of test cases

1. ***ReST*** : test cases especially computed for the actual situation by
   a) generating a ***quasi exhaustive set of test cases*** (***QuEST***) based on an analysis of the rule structure and the input/output behavior they perform
   b) reducing ***QuEST*** down to a ***reasonable set of test cases*** (***ReST***) based on topical validation criteria and their application dependent ranking and rating
2. The historical cases in ***VKB***

What do these cases look like?

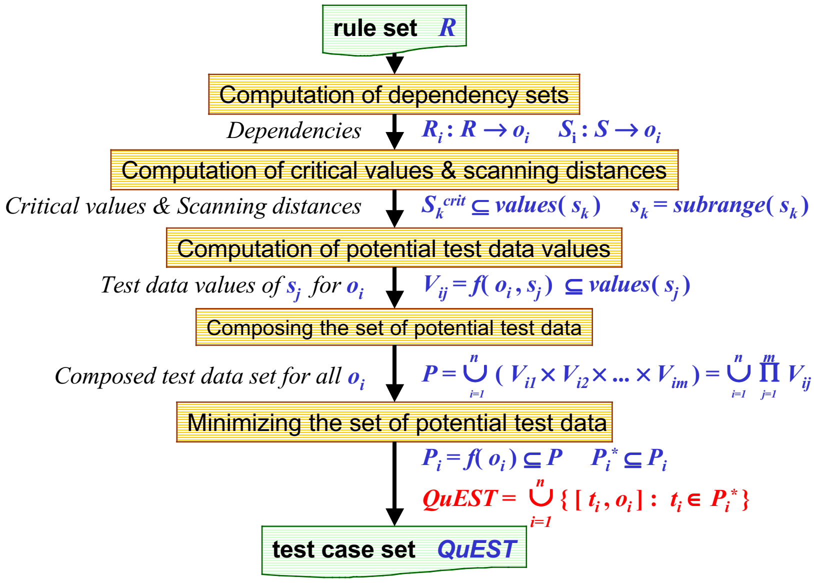
1. ***ReST*** is a set of pairs $[\, t_j , sol_{ij} \,]$ consisting of the test cases input (the test data) t_j and a solution sol_{ij} that is provided to t_j by the test case solver e_i, who is either
   a) a human expert of the panel $e_i \in \{e_1, ..., e_n\}$ or
   b) the system under evaluation e_{n+1}. *(initially the only available solution)*
2. ***VKB*** contains 8-tuples $[\, t_j , E_K , E_I , sol_{Kj}^{opt} , r_{IjK} , c_{IjK} , \tau_S , D_C \,]$.
   Cases in the format $[\, t_j , sol_{ij} \,]$ can be extracted by the projection of the appropriate elements $[\, \Pi_1 (VKB) , \Pi_4 (VKB) \,]$.

5.1 Generating *QuEST*

General Approach

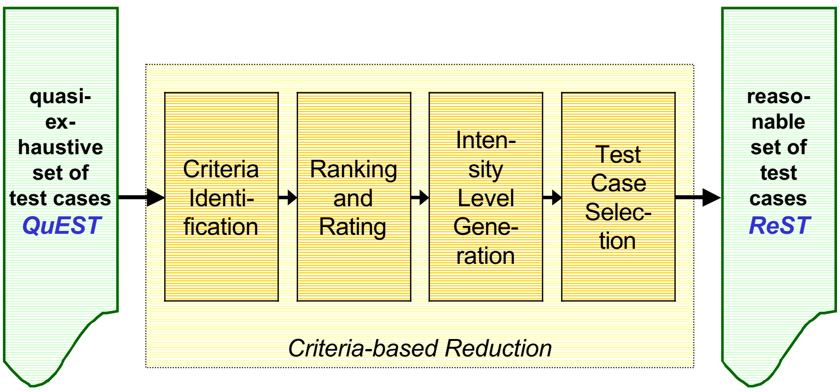
1. Break down the range of an input into sub-ranges where its values are considered to be equivalent in terms of its effects on the outputs
2. Compute an initial set of potential test data P based upon all computable combinations of values that surround these sub-ranges
3. Sort these data into several sets P_i of data according to their system's output o_i , i.e. form partitions of P
4. Filter all P_i by eliminating those that are subsumed by others, i.e. exclusively surrounded by test cases with the same system's output

Particular Steps of Generating QuEST



5.2 Forming *ReST from QuEST*

General Approach



Criteria Identification

- **Conceptual Criteria**
  - Domain Related Criteria ***DRC***
    criticality, complexity, sensitivity, domain coverage, ...
  - Input Related Criteria ***IRC***
    criticality, sensitivity, characteristics, ...
  - Output Related Criteria ***ORC***
    probability, criticality, sensitivity, costs, robustness, ...
- **Human Criteria**
  - Expert Related Criteria ***ERC***
    competence, credibility, availability, ...
  - Validator Related Criteria ***VRC***
    objectivity, competence, independence, neutrality, ...
  - User Related Criteria ***URC***
    acceptable performance, maintainability, effectiveness, usability, ...

Ranking & Rating

- **ranking**

  describes proportions among various criteria
- **rating**

  describes the degree of a criterion's influence on the investigated domain for the considered output

In both procedures, ranking and rating the assessments are quantified by natural numbers of a range from zero (**0**) to a top level assessment r_{max}.

To ensure the entire expressiveness of such a quantification, in both procedures, ranking and rating the complete range should be used: Both

- the zero level assessment **0** and
- the top level assessment r_{max}

should occur at least once.

Procedure of Ranking & Rating

Data available

- k_D DRC , k_O ORC , r_{max}
- n outputs o_i , sensor dependency sets S_i , test data $P_i^* \subseteq \Pi_{inp}(QuEST)$
- m sets of critical values S_i^{crit}
- $m * n$ sets $V_{ij}(o_i, s_j)$

Ranking: associate a rank $\hat{r} = 1, \dots, r_{max}$:

- $\hat{r}(drc)$ $\forall$ $drc_i \in DRC$ ($i = 1, \dots, k_D$)
- $\hat{r}(orc)$ $\forall$ $orc_i \in DRC$ ($i = 1, \dots, k_O$)

Rating: associate a rating $r = 1, \dots, r_{max}$:

- $r(drc)$ $\forall$ $drc_i \in DRC$ ($i = 1, \dots, k_D$)
- $r_{ij}(o_i, orc_j)$ $\forall$ $orc_i \in DRC$ ($i = 1, \dots, k_O$) $\forall$ o_i ($i = 1, \dots, n$)

Intensity Level Generation

- Global Test Necessity Level TNL
  - weight (rank & rating) of a drc_i ($0 < w \leq r_{max}^2$) :

    $w(drc_i) = \hat{r}(drc_i) * r(drc_i)$

  - its maximum, normalized by its possible max. value r_{max}^2:

    $TNL = max(\{w(drc_i): 1 \leq i \leq k_D\}) \; / \; r_{max}^2$

- Local Test Necessity Level $tnl(o_i)$
  - weight (rank & rating) of an o_i (o_i 's *validation necessity*) :

    $$w(o_i) = 1 / k_O \sum_{i=1}^{k_O} \hat{r}(orc_j) * r_{ij}$$

  - normalized by its max. value and put into the perspective of TNL ($0 \leq tnl(o_i) \leq 1$) :

    $$tnl(o_i) = \frac{TNL * w(o_i)}{max(\{w(o_k): 1 \leq k \leq n\})} = TNL * \hat{w}(o_i)$$

Test Case Selection

Informally speaking, the more a test case ...

1. ... has relevant input data that is relevant for other test cases as well,
2. ... has relevant input data that contributes to outputs with high rated local test necessity levels,
3. ... has relevant input data within an important interval of its range, and
4. ... is situated near a border of a region of influence,

the more this test case has a claim to become a member of ***ReST***.

Formally expressed, this can be estimated as follows:

Test Sufficiency Level $tsl(t_k)$ **of a test data** $t_k \in \Pi_{inp}(QuEST)$

1. weight of a sensor dimension s_j
   (how many and how important outputs s_j contributes to)

$$w(s_j) = \Sigma\, c_i * tnl(o_i) \qquad c_i = \begin{cases} 0, \text{ iff } s_j \notin S_i \\ 1, \text{ iff } s_j \in S_i \end{cases}$$

   ... normalized by its maximum:

$$\hat{w}(s_j) = \frac{w(s_j)}{max(\{w(s_k): 1 \leq k \leq m\})}$$

2. weight of a sensor value s_j^{val}

$$w(s_j^{val}) = \sum_{i=1}^{n} c_{ij} * tnl(o_i) \qquad c_{ij} = \begin{cases} 0, \text{ iff } s_j \notin S_i \\ 1, \text{ iff } s_j \in S_i \ \& \ s_j^{val} \notin (V_{ij} \cup S_j^{crit}) \\ 2, \text{ iff } s_j \in S_i \ \& \ s_j^{val} \in (V_{ij} \setminus S_j^{crit}) \\ 3, \text{ iff } s_j \in S_i \ \& \ s_j^{val} \in (S_j^{crit} \setminus V_{ij}) \\ 4, \text{ iff } s_j \in S_i \ \& \ s_j^{val} \in (V_{ij} \cap S_j^{crit}) \end{cases}$$

... normalized by its maximum:

$$\hat{w}(s_j^{val}) = \frac{w(s_j^{val})}{max(\{w(s_j^x): 1 \leq x \leq k\})}$$

3. weight of a test data t_k

$$w(t_k) = 1/m \sum_{j=1}^{m} \hat{w}(s_j) * \hat{w}(s_j^k)$$

... normalized by its maximum & complemented as to 1:

$$tsl(t_k) = 1 - \frac{w(t_k)}{max(\{w(t_x): 1 \leq x \leq |P_i^*|\})}$$

Forming ***ReST*** by reducing $\Pi_{inp}(QuEST) \subseteq \bigcup_{i=1}^{n} P_i^*$:

$$ReST = \{ [t_i , o_i] : (t \in P_i^*) \wedge (tsl(t) \leq tnl(o_i)) \}$$

Remember, ***ReST*** is the one of the sources of test cases for the TURNG Test.

5.3 How is VKB involved in the framework?

1. **The cases in VKB don‘t have to be solved by the expert panel ...**

   ..., since there is already a solution in VKB, which is even the best solution of all past validation sessions.

3. **The cases of VKB have to be rated by the expert panel ...**

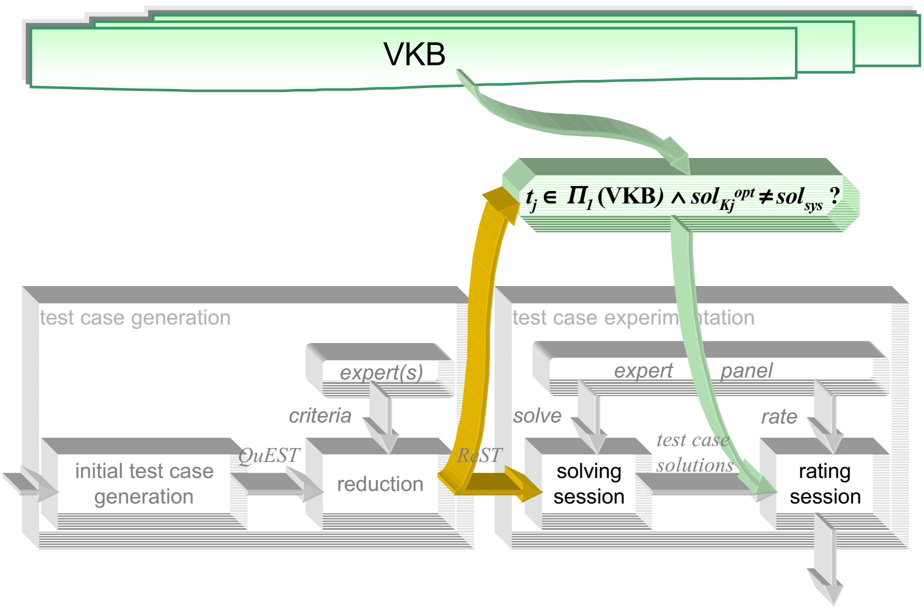
   ..., since this panel is responsible for the results of the present validation process and thus, it needs to have influence on the ratings.

4. **Only cases with solutions different from the system‘s solution are of interest ...**

   ..., since

   a) we are interested in new external knowledge that is outside the expertise of the expert panel and

   b) the system‘s solution is in the process anyway.



5.4 How is VESA involved in the framework?

Objectives

- Forming a model of each validator's individual knowledge and behavior
- Successive refinement of this model by consecutive validation session

Source of VESA's knowledge

- solving and rating results
  - a) of the associated human origin
  - b) of other human validators who often have the same opinion as the associated human origin

For (b), anonymity needs to be ensured.

5.4.1 Dynamic VESAs

- are formed just in the moment of their need and „forgotten" after their usage
- model just the required aspect of their human origin based on historical information of former sessions (i.e. not the current session)
- are requested in case its human origin is not available
- may be requested even if the human origin is present to validate the VESA concept itself by comparing the behavior of VESA with the real one of the human origin. (*subject of upcoming research*)

The knowledge base to dynamically form a VESA in case of need is simple:

For

- each human expert
- each and every solution to
- each test data and
- each and every rating of
- each and every historic session indicated by its time stamp.

5.4.2 The formation and usage of a VESA for test case solving

If a validator e_i is not available to solve a present case t_j the following applies:

Step # 1

In case e_i solved (*with a solution different from „unknown“*) or rated t_j in former sessions, his/her provided or as „correct“ rated solution with the latest time stamp τ_S will be provided by ***VESA***.

Step # 2

(1) All validators e', who ever delivered a solution or a ratring to the present case t_j form a set $Solver_i^0$, which is an initial dynamic agent for e_i :

$$Solver_i^0 := \{e' : [t_j, E_K, E_I ...] \in VKB \wedge (e' \in E_K \vee e' \in E_I)\}$$

(2) Select the most similar expert e_{sim} with the largest set of cases that have been solved by both e_i and e_{sim} (a) with the same solution and (b) in the same session. e_{sim} forms a refined dynamic agent $Solver_i^1$ for e_i :

$$Solver_i^1 := e_{sim} : (e_{sim} \in Solver_i^0) \wedge$$
$$(| \{[[t_j, e_i, sol_{ij}, \tau_s],[t_j, e_{sim}, sol_{sim,j}, \tau_s]] : sol_{ij} = sol_{sim,j}\} | \to \max!)$$

(3) Provide the latest provided or as „correct“ rated solution of the expert e_{sim} to the present case t_j , i.e. the solution with the latest time stamp τ_S by ***VESA***.

Step # 3

If there is no most similar expert, provide the solution $sol := unknown$ by ***VESA***.

5.4.3 The formation and usage of a VESA for test case rating

If a validator e_i is not available to rate a present case t_j the following applies:

Step # 1

If e_i considered (solved or rated) the same test case t_j in former sessions, look at the rating or the provided solution with the latest time stamp τ_S :

1. In case the latest consideration is a rating, both the same rating r and the same certainty c are adopted and provided by ***VESA***.
2. In case the latest consideration is a provided solution sol (*different from „unknown“*), provide for this solution a rating $r := 1$ (correct) and a certainty $c := 1$ (for sure) and for all other solutions a rating $r = 0$ (incorrect) and a certainty $c := 1$ by ***VESA***.

Step # 2

If e_i never considered (solved or rated) the test case t_j in former sessions, look for a „most similar“ expert e_{sim} :

(1) All validators e' , who ever delivered a rating or a solution (*different from „unknown“*) to the present case t_j form a set $Solver_i^0$, which is an initial dynamic agent for e_i :

$$Solver_i^0 = \{e' : ([t_j, E_K, E_I ...] \in VKB) \wedge ((e' \in E_K) \vee (e' \in E_I))\}$$

(2) Select the most similar expert e_{sim} with the largest set of cases that have been considered (solved or rated) by both e_i and e_{sim}

- with the same solution sol (*different from „unknown“*) respectively the same rating r (*different from „norating“*)
- in the same session.

e_{sim} forms a refined dynamic agent $Solver_i^1$ for e_i :

$$Solver_i^1 := e_{sim} : (e_{sim} \in Solver_i^0) \wedge$$
$$(| \{[[t_j, e_i, sol_{ij}, \tau_s], [t_j, e_{sim}, sol_{sim,j}, \tau_s]] : sol_{ij} = sol_{sim,j}\} \cup$$
$$\{[[t_j, e_k, e_i, sol_{kj}, r_{ijk}, \tau_s], [t_j, e_k, e_{sim}, sol_{kj}, r_{sim,j,k}, \tau_s]] : r_{ijk} = r_{sim,j,k}\} | \to \max!)$$

(3) If the latest consideration of t_j by e_{sim} is a rating r along with a certainty c, adopt it and provide both by ***VESA***.

(4) If the latest consideration of t_j by e_{sim} is a solution sol , provide for this solution a rating $r := 1$ (correct) and a certainty $c := 1$ (for sure) and for all other solutions a rating $r := 0$ (incorrect) and a certainty $c := 1$ by ***VESA***.

Step # 3

If there is no „most similar“ expert e_{sim}, provide the rating $r := norating$ and a certainty $c := 0$.

5.5 How to perform and evaluate the TURING Test ?

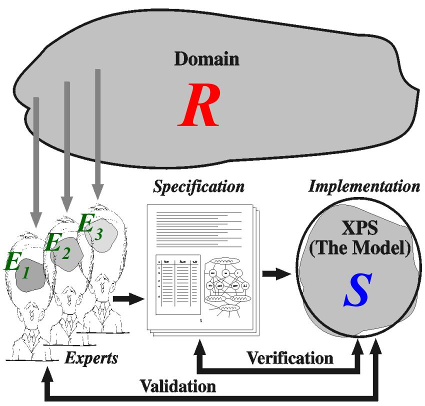
Validation Scenario

(non-accessible) desired target behavior $R \subseteq I \rho O$

(estimated by $\bigcup_{i=1}^{n} E_i$)

n experts holding the domain knowledge $E_1, ..., E_n$

system to be validated with an input-output relation $S \subseteq I \rho O$



In- and Output of the TURING Test and its evaluation

input

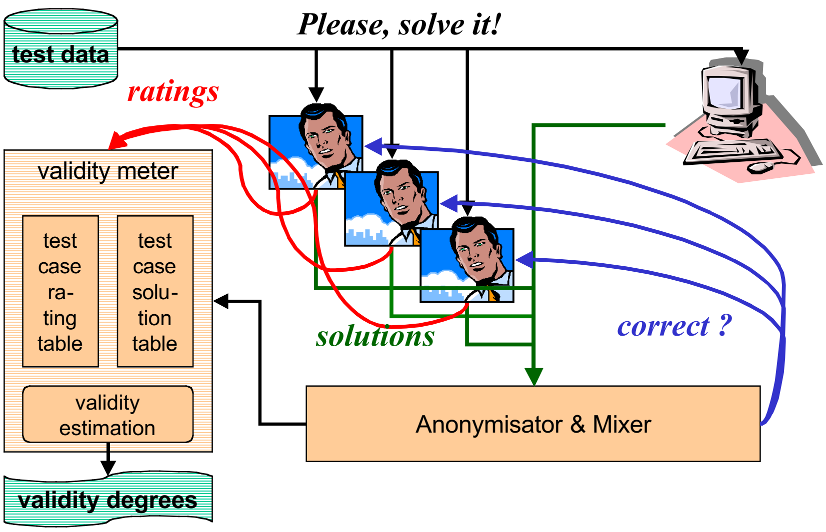
- the system which is to be validated
- a panel of n experts
- the reasonable test data set ***ReST*** consisting of m test data
- two ratings $\{ 0, 1 \}$, in which 1 means to be correct and 0 means to be incorrect
- two degrees of certainty $\{ 0, 1 \}$, in which 1 means to be sure and 0 means to be unsure

output

- m validity degrees associated with test cases $0 \leq v_{sys}(t_j) \leq 1$

Steps of the TURING Test

1. solving test cases by both the experts and the system
2. randomly mixing the test case solutions and removing their authorship
3. rating all upcoming test case solutions (anonymously) by the experts



5.5.1 Solving test cases

Solving m test data t_j by n (human) experts $e_1, \ldots, e_n$ and the system e_{n+1} leads to $m * (n+1)$ solved test cases $[\ t_j, e_i, sol_{ij}\ ]$ with the solution sol_{ij}.

sol_{ij} is either a real output or *unknown* by its provider: $sol_{ij} \in O \cup \{\ unknown\ \}$

The output set O is formed by **all** upcoming solutions:

$$O = \Pi_2(E_{n+1}) \cup \Pi_2(\bigcup_{i=1}^{n} E_i) = \Pi_2(\bigcup_{i=1}^{n+1} E_i)$$

5.5.2 Mixing solutions and removing their authorship

Each of the n human validators receives all the $m * (n+1)$ upcoming solved test cases anonymously, i.e. without any information about the authorship.

5.5.3 Rating test case solutions

- With a rating $r \in \{\ 0, 1\ \}$ and a certainty $c(r) \in \{\ 0, 1\ \}$ the experts express
  - their opinion about the solution ($r := 1$: *correct*, $r := 0$: *incorrect*)
  - their confidence to be valid ($c := 1$: *sure*, $c := 0$: *unsure*)
- Additionally, they have the chance to express a lack of competence by $r = norating$ ($c(norating) = 0$)

Each **rating** r_{ijk} is assigned to a solution sol_{kj} of the expert e_k ($1 \leq k \leq n+1$) of a test data t_j ($1 \leq j \leq m$) and an evaluating (human) expert e_i ($1 \leq i \leq n$) and has the certainty c_{ijk} .

5.5.4 Evaluating the ratings

input	output
➢ $m * (n+1)$ solved test cases ➢ $n * m * (n+1)$ rated test cases	➢ a validity degree $v_{sys}(t_j)$ $(0 \le v_{sys} \le 1)$ for each test data $t_j \in \Pi_1(ReST)$

approach
average rating of the system's solution by the experts, each one weighted by the considered expert's competence for t_j as well as by his/her certainty

Competence estimation

of an expert e_i for a test case t_j is based on

1. his/her own evaluation to be competent
2. his/her certainty while rating other experts' solutions
3. his/her consistency in the solving and the rating process
   Does he/she give his/her own solution good marks?
4. his/her stability
   Is he/she certain while rating his/her own solution?
5. the other experts' ratings of his/her solution

(1) his/her own evaluation to be competent
is indicated by giving the solution ***unknown*** and/or the rating ***norating***:

$$slf\_est(e_i,t_j) = \frac{1}{2}\, ord(sol_{ij} \neq unknown) + \frac{1}{2}\,\frac{1}{n} \sum_{k=1,k\neq i}^{n+1} ord(r_{ijk} \neq norating)$$

(2) his/her certainty while rating other experts' solutions
is indicated by the ratio between the number of certain ratings and the number of rating at all:

$$crt\_est(e_i,t_j) = \frac{1}{n} \sum_{k=1,k\neq i}^{n+1} c_{ijk}$$

(3) his/her consistency in the solving and the rating process
is indicated by the rating of the own solution:

$$cns\_est(e_i,t_j) = r_{iji}$$

(4) his/her stability
is indicated by the certainty of the own solution's rating:

$$stb\_est(e_i,t_j) = c_{iji}$$

(5) the other experts' ratings of his/her solution
is indicated by their average (certain) ratings:

$$frgn\_est(e_i,t_j) = \frac{1}{\sum_{k=1,k\neq i}^{n} c_{kji}} \sum_{k=1,k\neq i}^{n} c_{kji} * r_{kji}$$

Classification of sources that indicate competence/non-competence

(1) intentional reflection (self-estimation & certainty): slf_est , crt_est

(2) non-intended reflection (consistency & stability): cns_est , stb_est

(3) external (foreign) **estimation**: $frgn\_est$

All sources should be taken in account equally:

$$cpt\,(e_i, t_j) = 1/3\ (1/2\ slf\_est + 1/2\ slf\_est) + 1/3\ (1/2\ cns\_est + 1/2\ stb\_est) + 1/3\ frgn\_est$$

Finally, the average rating of the system's solution by the experts, each one weighted by the considered expert's competence for t_j as well as by his/her certainty is

$$v_{sys}(t_j) = 1/(\sum_{i=1}^{n} (cpt(e_i, t_j) * c_{ij(n+1)})) * \sum_{i=1}^{n} (cpt(e_i, t_j) * c_{ij(n+1)} * r_{ij(n+1)})$$

This serves as an estimation of the system's validity for a test data t_j.

How is VKB involved in the TURING Test ?

1. **utilization of VKB** - *see slide #26*
   a) It is not used for the test case solving session
   b) Cases which have a „new“ solution (different from the system‘s solution) in VKB, are subjects of the rating session as well.
2. **maintenance of VKB**
   ♋ The validation of the validation knowledge is ensured automatically:
   - ➢ It is re-validated in future sessions by newly rating it.

   ♌ Updating, in this context, means adding new cases to VKB.
   - ➢ The „experience“ of a session is its (very best) solution sol_{Kj}^{opt} to each test data $t_j \in \Pi_I(ReST)$ that was considered in the session.
   - ➢ Additionally kept in VKB:
     - ➢ a list of solvers E_K ,
     - ➢ a list of raters E_I along with their ratings r_{IjK} and certainties c_{IjK}
     - ➢ a time stamp τ_S (to compute competence trends, e.g.) and
     - ➢ an informal context description D_C

5.6 Validity assessment

Desired validity statements: Validity degrees associated with

(1) test cases = *result of the TURNG Test*
(2) outputs - *to be computed*
(3) rules - *to be computed*
(4) the entire system - *to be computed*

Validity statements like

- ➢ **(2)** and **(4)** might be useful for (potential) **system users** and/or managers,
- ➢ **(3)** for **system developers**, namely knowledge engineers, and
- ➢ **(1)** is the basis of formal **system refinement**.

(1) Validities associated with test data - see slide # 35

$$v_{sys}(t_j) = 1 / \left(\sum_{i=1}^{n} (cpt(e_i , t_j) * c_{ij(n+1)}) \right) * \sum_{i=1}^{n} (cpt(e_i , t_j) * c_{ij(n+1)} * r_{ij(n+1)})$$

(2) Validities associated with outputs

$$v_{sys}(sol_k) = 1 / | T_k | \sum_{[t_j, sol_k] \in T_k} v_{sys}(t_j)$$

with $T_k = \{ [t_j , sol_k] \in ReST : t_j \in \Pi_{inp}(ReST) , [t_j , sol_k] \in E_{n+1} \}$

(3) Validities associated with rules

$$v_{sys}(r_l) = 1 / | T_l | \sum_{[t_j, sol_k] \in T_l} v_{sys}(t_j)$$

with $T_l = \{ [t_j , sol_k] \in ReST : t_j \in \Pi_{inp}(ReST) , [t_j , sol_k] \in E_{n+1} , \ t_j \ uses \ r_l \}$

(4) Global (average) validity

$$v_{sys} = 1 / | ReST | \sum_{j=1}^{|ReST|} v_{sys}(t_j)$$

5.7 Formal System Refinement

Objective: *Building a „refined“ system based on*

1. *a former historic (non - refined) system and*
2. *examples that turned out to be correct in practice.*

Steps of the Technique

1. Finding guilty rules

   A rule is <u>guilty</u>, if it‘s conclusion part is a system‘s output for which some human solution to this test data received better marks then the system‘s one. Also an <u>optimal solution</u> for each test data (the one that got the best marks) is determined.

2. Modification of guilty rules with identical optimal solutions for the test data that used this rule
3. Computing substitutes for rules with several optimal solutions for the test data that used this rule
4. Recompiling the developed new rules into the remaining knowledge base

5.7.1 Finding Guilty Rules

1. The validity assessment procedure determined a rule-associated validity
$$v_{sys}(r_l) = 1/|T_l| \sum_{[t_j, sol_k] \in T_l} v_{sys}(t_j)$$

2. There is a set T_l^* of test cases with test data $t_j \in \Pi_{inp}(T_l)$ and all solution parts which came up in the experimentation by any expert e_i $(i = 1, ..., n, n+1)$: $T_l^* = T_l \cup \{ [t_j, sol(e_i, t_j)] : \exists [t_j, sol_k] \in T_l \}$

3. The set T_l^* can be split into subsets $T_{l1}^*, ..., T_{lm}^*$ according to their different solution parts $sol_1, ..., sol_m$.

4 Analogously to $v_{sys}(sol_k)$, a validity $v(r_l, sol_p)$ $(1 \leq p \leq m)$ of each solution sol_p, - but only based on the test cases of T_{lp}^* - can be computed:
$$v(r_l, sol_p) = \frac{1}{|T_{lp}^*|} \sum \frac{1}{\sum (cpt(e_i, t_j) * c_{ijq})} \sum (cpt(e_i, t_j) * c_{ijq} * r_{ijq})$$

5 The **optimal validity** of r_l is the maximum of all $v(r_l, sol_p)$ among the solutions sol_p occurring in T_l^*.
The associated solution is the ***optimal solution*** sol_{opt} of r_l : $v_{opt}(r_l) = v_{opt}(r_l, sol_{opt}) = max(\{ v(r_l, sol_1), ..., v(r_l, sol_m) \})$.
$v_{opt}(r_l)$ is an upper limit of the reachable rule-associated validity of r_l.

If $v_{opt}(r_l, sol_{opt}) > v(r_l)$, there is a solution in T_l^* that got better marks than the system's one: $v_{opt}(r_l, sol_{opt}) > v(r_l) \Rightarrow r_l$ ***is guilty***

5.7.2 Simple Refinement by Replacing the Conclusion

If all test cases within T_l of a guilty rule r_l have the same optimal solution sol_k , which was different from the system's solution, the conclusion-part of this rule has to be substituted by sol_k:

$\forall [t_j, sol_k] \in T_l : sol_s$ is optimal solution for $[t_j, sol_k]$ $\Rightarrow$

r_l : (if-part $\rightarrow sol_k$) $\leftrightarrow$ (if-part $\rightarrow sol_s$)

5.7.3 Computing Substitutes for the Remaining Guilty Rules

Step # 1

T_l^* of a guilty rule r_l is split into subsets $T_l^{*1}, \ldots, T_l^{*n}$ according to the solution $sol_s \in \Pi_{outp}(T_l^*)$ ($1 \leq s \leq n$) for each $t_j \in \Pi_{inp}(T_l)$ that obtained the highest validity $v(r_l, sol_s)$.

The if-part(s) of the new substitute rule(s) for a guilty rule r_l are expressions $exp_i \in E$ of p new alternative rules $\{ r_l^1, \ldots, r_l^p \}$ for each T_l^s and will be noted as a set of sets

$$P_l^s = \{ \{exp_1^1, \ldots, exp_{p_1}^1\} , \{exp_1^2, \ldots, exp_{p_2}^2\} , \ldots , \{exp_1^p, \ldots, exp_{p_p}^p\} \}$$

The corresponding rules of P_l^s are

$$r_l^1: \bigwedge_{i=1}^{p_1} e_i^1 \rightarrow sol_s \qquad r_l^2: \bigwedge_{i=1}^{p_2} e_i^2 \rightarrow sol_s \qquad \ldots \qquad r_l^1: \bigwedge_{i=1}^{p_p} e_i^1 \rightarrow sol_s$$

Step # 2

Pos is the set of Positions (dimensions of the input space), at which the $t_j \in \Pi_{inp}(T_l^{*s})$ are **not** identical.

The generation of the if-parts P_l^s is managed by a **Reduction System**, which is applied to triples $[T_l^{*s} , Pos , P_l^s]$ until Pos becomes the empty set $\emptyset$.

Step # 3

The **starting point** of the reduction procedure is $[T_l^{*s}, Pos, P_l^s]$ with

$$P_l^s = \{\{ (s_1 = s_1^{ident}), \ldots , (s_q = s_q^{ident})\}\}$$

$s_1, \ldots, s_q$ are those positions, where all test data $t_j \in \Pi_{inp}(T_l^{*s})$ have the same (identical) value $s_1^{ident} , \ldots, s_q^{ident}$ and Pos is the set of the remaining positions:

$$Pos = \{s_i : \forall \; if\text{-}part \in P_l^s : \; (s_i = s_i^{ident}) \notin if\text{-}part\}$$

Step # 4

Applying the Reduction System to $[T_l^s , Pos , P_l^s]$ until Pos becomes the empty set $\emptyset$.

- *The Reduction system consists of two rules.*
- *The system is deterministic, i.e. exactly one of these rules is applicable at a time.*
- *The system terminates, since **Pos** looses one element in each cycle, i.e. at some point it must become empty.*

Reduction Rule # 1

- position $pos \in Pos$, s_{pos} has a value set without any application-driven $\leq$ relation
- $\{ s_{pos}^{1}, \ldots , s_{pos}^{m} \}$ are the values of s_{pos} occurring in T_l^{*s}

$\Rightarrow$

$[T_l^{*s} , Pos , \{ p_1 , \ldots , p_n \}] \quad \leftrightarrow$

1. $[T_l^{*s,1} \setminus \{ [t_j,sol_s]: s_{pos} \neq s_{pos}^{1} \} , Pos \setminus \{ pos \} , \bigcup_{i=1}^{n} (p_i \cup \{ (s_{pos} = s_{pos}^{1}) \})]$

...

m. $[T_l^{*s,m} \setminus \{ [t_j,sol_s]: s_{pos} \neq s_{pos}^{m} \} , Pos \setminus \{ pos \} , \bigcup_{i=1}^{n} (p_i \cup \{ (s_{pos} = s_{pos}^{m}) \})]$

Continue with each $T_l^{*s,i}$ ($1 \leq i \leq m$) separately.

Reduction Rule # 2

- position $pos \in Pos$, s_{pos} has a value set with an application-driven $\leq$ relation
- s_{pos}^{min} / s_{pos}^{max} are the smallest / largest value of s_{pos} in T_l^{*s}

$\Rightarrow$

$[T_l^{*s} , Pos , \{p_1, .., p_n\}] \quad \leftrightarrow$

$[T_l^{*s}, Pos \setminus \{ pos \} , \bigcup_{i=1}^{n} p_i \cup \{ (s_{pos} \geq s_{pos}^{min}) , (s_{pos} \leq s_{pos}^{max}) \} \cup S_{excl}]$

S_{excl} is the set of excluded values for s_{pos} that have to be mapped to a solution different from sol_s because of belonging to some other T_u^{v} with $v \neq s$:

$$S_{excl} = \{ (s_{pos} \neq s_{pos}^{j}) : \exists [t_j , sol_s] \in T_l^{*s} \ \exists [t_m , sol_v] \in T_u^{*v} \ (v \neq s) \ \text{ with } \forall p \neq pos \ ((s_p^{j} = s_p^{m}) \wedge (s_{pos}^{min} \leq s_{pos}^{m} \leq s_{pos}^{max})) \}$$

5.7.4 Recompiling the new Rules into the Knowledge Base

First, in case the ***if*** - part of a new rule contains a subset of expressions that is the ***if*** - part of another rule having an intermediate solution as its ***then*** - part, this subset has to be replaced by the corresponding intermediate solution:

$$\exists r_i : (\, \textit{if}\text{-part}_1 \rightarrow \textit{int}_1 \,) \wedge \exists r_j : (\, \textit{if}\text{-part}_2 \wedge \textit{if}\text{-part}_1 \rightarrow \textit{concl} \,)$$
$$\Rightarrow$$
$$r_j \leftrightarrow (\, \textit{if}\text{-part}_2 \wedge \textit{int}_1 \rightarrow \textit{concl} \,)$$

Second, remove the useless rules that map to an intermediate result which is not used for further inference steps:

$$\exists r_i : (\, \textit{if}\text{-part} \rightarrow \textit{int}_1 \,) \wedge \neg\exists r_j : (\, \text{int}_1 \wedge \text{rest-}\textit{if}\text{-part} \rightarrow \textit{then-part} \,)$$
$$\Rightarrow$$
$$r_i \leftrightarrow \varnothing$$

6 Towards a Prototype Application

6.1 Knowledge Base

6.1.1 Initial Informal Knowledge

What application topic the fine for such an experiment that requests

- much human cooperation of „domain experts“
- without having the money to hire them?

The answer to this issue is: It must be a domain with a certain entertainment factor, so that they like this cooperation, for example ...

The selection of an appropriate wine to a given dinner.

General Rules

- ✓ Meals that are rich in content call for a wine that is rich in body.
- ✓ Light meals call for a light wine.
- ✓ Premium meals call for a fine and premium wine.

Particular Rules

Meat

- ✓ ***Light colored meat***, *such as fowl and veal call for a fruity, grapy red wine with less tannin.*
- ✓ ***Fried and grilled meats*** *call for a young red wine rich in tannin.*
- ✓ *With **smoked meat** there is a correlation between the length of time in the meal's preparation and the time to mature the wine. Furthermore, tannins help to make the food digestible. Thus, a mature Barolo or a mature Brunello fits well.*
- ✓ *The autumnal and slightly sweet taste of **venison** calls for a strong partner. A dark, fruit-accentuated red wine from the "new world" is appropriate. Alternatively, a mature red wine from Burgundy, Bordeaux or the Rhone-Valley is acceptable.*

Fish

- ✓ ***Steamed fish*** *calls for a light, fresh and low acid-accentuated white wine. An alternative is a dry, fruity, low tannin Rosé.*
- ✓ ***Fried or grilled Fish*** *has an intensive taste and gets along well with a (possibly in a wooden barrel matured) white wine or a red wine that has not too much tannin. To summarize, a strong white wine or a low tannin red wine is acceptable.*

Asian Meals

- ✓ *The intensive flavoring and spice fits with the freshness and intensity of an aromatic white wine. Muscatel, Gewuerztraminer, Sauvignon Blanc, or a semi-dry Riesling are appropriate wines.*

Cheese

- ✓ ***Hard cheese*** *calls for a white wine that is rich in content. Or a velvet, low tannin red wine, especially Pinot Noir or Amarone.*
- ✓ ***Soft cheese*** *needs a similar wine that hard cheese, but a little lighter. Beaujolais is also acceptable.*
- ✓ ***Goat cheese*** *calls for a dry and fruity white wine.*
- ✓ ***Blue mold cheese*** *fits well with any wine other than sweet ones.*

Desserts

The switch in taste that comes with the dessert needs a switch in the wine taste as well.

- ✓ *Fruit dessert fits well with Riesling that is rich in acid.*
- ✓ *Aromatic desserts (flavored with cloves, anise, or cinnamon, e.g.) call for a Gewuerztraminer.*
- ✓ *Ice cream fits best with Prot Wine.*

6.1.2 Formalizing Knowledge

Inputs

- the main ingredient
- the kind of preparation
- the style of preparation

input space

I = { [s_1, s_2, s_3] :

s_1	$\in$	{pork, beef, veal, venison, fowl, meat, fish, hard cheese, soft cheese, goat cheese, blue mold cheese, fruit dessert, aromatic dessert, ice cream},
s_2	$\in$	{non (raw), steamed, boiled, grilled, fried, stewed, casserole, deep fried},
s_3	$\in$	{Asian, Western } **}**

theoretical number input combinations 14 x 8 x 2 = 224

(*Some of them don't make sense: grilled ice cream, e.g.*)

Outputs

- a kind of wine

output space **$O = \{ o_1, o_2, \ldots, o_{24} \}$**

o_1	*Red wine, fruity, low tannin, less compound*
o_2	*Red wine, young, rich of tannin*
o_3	*Red wine, dark, fruity, from the „new world"*
o_4	*Red wine, maturely, from the Rhone valley (France)*
o_5	*Red wine, velvet, low tannin*
o_6	*Pinot Noir*
o_7	*Amarone*
o_8	*Burgundy, mature*
o_9	*Bordeaux, mature*
o_{10}	*Barolo, mature*
o_{11}	*Brunello, mature*
o_{12}	*Beaujolais*
o_{13}	*Rosé, dry, fruity, low tannin*
o_{14}	*White wine, light, fresh, low acid*
o_{15}	*White wine, strong, low tannin*
o_{16}	*White wine, rich in content*
o_{17}	*White wine, dry, fruity*
o_{18}	*Muscatel*
o_{19}	*Gewuerztraminer*
o_{20}	*Sauvignon Blanc*
o_{21}	*Riesling, semi dry*
o_{22}	*Riesling, rich of acid*
o_{23}	*Port wine*
o_{24}	*Any wine besides smooth one*

6.1.3 Formal Knowledge Base: $R = \{ r_1, r_2, \ldots, r_{38} \}$

r_1	*Red wine, fruity, low tannin, less compound*	$\leftarrow$		*(main ingredient = fowl)*
r_2	*Red wine, fruity, low tannin, less compound*	$\leftarrow$		*(main ingredient = veal)*
r_3	*Red wine, young, rich of tannin*	$\leftarrow$		*(main ingredient = pork)*
			$\wedge$	*(preparation = grilled)*
r_4	*Red wine, young, rich of tannin*	$\leftarrow$		*(main ingredient = pork)*
			$\wedge$	*(preparation = fried)*
r_5	*Red wine, young, rich of tannin*	$\leftarrow$		*(main ingredient = beef)*
			$\wedge$	*(preparation = grilled)*
r_6	*Red wine, young, rich of tannin*	$\leftarrow$		*(main ingredient = beef)*
			$\wedge$	*(preparation = fried)*
r_7	*Red wine, fruity, low tannin, less compound*	$\leftarrow$		*(main ingredient = fowl)*
r_8	*Red wine, fruity, low tannin, less compound*	$\leftarrow$		*(main ingredient = veal)*
$r_{9.1}$	*Barolo, mature*	$\leftarrow$		*(main ingredient = pork)*
			$\wedge$	*(preparation = stewed)*
$r_{9.2}$	*Barolo, mature*	$\leftarrow$		*(main ingredient = beef)*
			$\wedge$	*(preparation = stewed)*
$r_{9.3}$	*Barolo, mature*	$\leftarrow$		*(main ingredient = veal)*
			$\wedge$	*(preparation = stewed)*
$r_{9.4}$	*Barolo, mature*	$\leftarrow$		*(main ingredient = venison)*
			$\wedge$	*(preparation = stewed)*
$r_{9.5}$	*Barolo, mature*	$\leftarrow$		*(main ingredient = fowl)*

● ● ●

6.2 Initial Test Cases

6.2.1 Generating *QuEST*

6.2.1.1 Dependency Sets

First, we computed for each $o_i \in O$

- a rule dependency set $R_i \subseteq R$ that contains the rules $r_k \in R$ on which o_i depends and
- a sensor dependency set $S_i \subseteq S$ that contains sensor variables $s_k \in S$ on which o_i depends

$R_1 = \{ r_1, r_2, r_7, r_8, r_{19}, r_{20} \}$	$R_{13} = \{ r_{16} \}$	$S_1 = \{ s_1, s_2 \}$	$S_{13} = \{ s_1, s_2 \}$
$R_2 = \{ r_3, r_4, r_5, r_6 \}$	$R_{14} = \{ r_{15} \}$	$S_2 = \{ s_1, s_2 \}$	$S_{14} = \{ s_1, s_2 \}$
$R_3 = \{ r_{11} \}$	$R_{15} = \{ r_{17}, r_{18} \}$	$S_3 = \{ s_1 \}$	$S_{15} = \{ s_1, s_2 \}$
$R_4 = \{ r_{14} \}$	$R_{16} = \{ r_{25}, r_{29} \}$	$S_4 = \{ s_1 \}$	$S_{16} = \{ s_1 \}$
$R_5 = \{ r_{26}, r_{30} \}$	$R_{17} = \{ r_{34} \}$	$S_5 = \{ s_1 \}$	$S_{17} = \{ s_1 \}$
$R_6 = \{ r_{27}, r_{31} \}$	$R_{18} = \{ r_{21} \}$	$S_6 = \{ s_1 \}$	$S_{18} = \{ s_3 \}$
$R_7 = \{ r_{28}, r_{32} \}$	$R_{19} = \{ r_{22}, r_{37} \}$	$S_7 = \{ s_1 \}$	$S_{19} = \{ s_1, s_3 \}$
$R_8 = \{ r_{12} \}$	$R_{20} = \{ r_{23} \}$	$S_8 = \{ s_1 \}$	$S_{20} = \{ s_3 \}$
$R_9 = \{ r_{13} \}$	$R_{21} = \{ r_{24} \}$	$S_9 = \{ s_1 \}$	$S_{21} = \{ s_3 \}$
$R_{10} = \{ r_9 \}$	$R_{22} = \{ r_{36} \}$	$S_{10} = \{ s_1, s_2 \}$	$S_{22} = \{ s_1 \}$
$R_{11} = \{ r_{10} \}$	$R_{23} = \{ r_{38} \}$	$S_{11} = \{ s_1, s_2 \}$	$S_{23} = \{ s_1 \}$
$R_{12} = \{ r_{33} \}$	$R_{24} = \{ r_{35} \}$	$S_{12} = \{ s_1 \}$	$S_{24} = \{ s_1 \}$

6.2.1.2 Critical values and scanning distances

- ✓ no numerical input sensor
- ✓ no input has a reasonable ordering relation in-between its possible values
- ⇒ all values of $\boldsymbol{s_1}$, $\boldsymbol{s_2}$, and $\boldsymbol{s_3}$ are considered critical

$\boldsymbol{S_1^{krit}}$ = {pork, beef, veal, venison, fowl, fish, hard cheese, soft cheese, goat cheese, blue mold cheese, fruit dessert, aromatic dessert, ice cream}

$\boldsymbol{S_2^{krit}}$ = {non (raw), steamed, boiled, grilled, fried, stewed, casserole, deep fried}

$\boldsymbol{S_3^{krit}}$ = {Asian, Western}

Scanning distances don't apply in this case.

6.2.1.3 Potential test case values

$\boldsymbol{V_{i,j}}$: set of potential values of the sensor (input) variable $\boldsymbol{s_j}$ to validate output $\boldsymbol{o_i}$.
(introduced as „normal value“: *any*)

$V_{1,1} = S_1^{krit}$	$V_{1,2} = S_2^{krit}$	$V_{1,1} = \{ any \}$
$V_{2,1} = S_1^{krit}$	$V_{2,2} = S_2^{krit}$	$V_{2,3} = \{ any \}$
$V_{3,1} = S_1^{krit}$	$V_{3,2} = \{ any \}$	$V_{3,3} = \{ any \}$
$V_{4,1} = S_1^{krit}$	$V_{4,2} = \{ any \}$	$V_{4,3} = \{ any \}$
$V_{5,1} = S_1^{krit}$	$V_{5,2} = \{ any \}$	$V_{5,3} = \{ any \}$
$V_{6,1} = S_1^{krit}$	$V_{6,2} = \{ any \}$	$V_{6,3} = \{ any \}$
$V_{7,1} = S_1^{krit}$	$V_{7,2} = \{ any \}$	$V_{7,3} = \{ any \}$
$V_{8,1} = S_1^{krit}$	$V_{8,2} = \{ any \}$	$V_{8,3} = \{ any \}$
$V_{9,1} = S_1^{krit}$	$V_{9,2} = \{ any \}$	$V_{9,3} = \{ any \}$
$V_{10,1} = S_1^{krit}$	$V_{10,2} = S_2^{krit}$	$V_{10,3} = \{ any \}$
$V_{11,1} = S_1^{krit}$	$V_{11,2} = S_2^{krit}$	$V_{11,3} = \{ any \}$
$V_{12,1} = S_1^{krit}$	$V_{12,2} = \{ any \}$	$V_{12,3} = \{ any \}$
$V_{13,1} = S_1^{krit}$	$V_{13,2} = S_2^{krit}$	$V_{13,3} = \{ any \}$
$V_{14,1} = S_1^{krit}$	$V_{14,2} = S_2^{krit}$	$V_{14,3} = \{ any \}$
$V_{15,1} = S_1^{krit}$	$V_{15,2} = S_2^{krit}$	$V_{15,3} = \{ any \}$
$V_{16,1} = S_1^{krit}$	$V_{16,2} = \{ any \}$	$V_{16,3} = \{ any \}$

• • •

6.2.1.3 Potential test data

$$P=\bigcup_{i=1}^{24}\prod_{j=1}^{3}V_{i,j}=(S_1^{krit}\times S_2^{krit}\times\{any\})\cup(S_1^{krit}\times\{any\}\times\{any\})\cup(\{any\}\times\{any\}\times S_3^{krit})\cup(S_1^{krit}\times\{any\}\times S_3^{krit})$$

$|S_1^{krit}|$ = 13 $|S_2^{krit}|$ = 8 $|S_3^{krit}|$ = 2 $|\{ any \}|$ = 1

⇒ 13∗8∗1 + 13∗1∗1 + 1∗1∗2 + 13∗1∗2 = 145 potential test data $t_1, \ldots, t_{145}$:

t_1	*pork*	*non (raw)*	*any*
t_2	*pork*	*steamed*	*any*
t_3	*pork*	*boiled*	*any*
t_4	*pork*	*grilled*	*any*
t_5	*pork*	*fried*	*any*
t_6	*pork*	*stewed*	*any*
t_7	*pork*	*casserole*	*any*
t_8	*pork*	*deep fried*	*any*
t_9	*beef*	*non (raw)*	*any*
t_{10}	*beef*	*steamed*	*any*
• • •			

6.2.2 Reducing *QuEST* to *ReST*

(simplified) **criteria**

1. *If a potential test case is semantically more general and subsumed by another one (for instance [pork, any, any] is more general than [pork, any, Asian]), it is removed, i.e. only the more specific one "survives" the reduction procedure.*
2. *If a test case is a meal that doesn't exist at all (for instance "grilled ice cream"), it is removed.*
3. *Meals which exists in only one of the styles Asian or Western (raw fish, e.g.), are only considered in this style, not in the other one.*
4. *Meals, which don't have a system's solution, don't become an element of **ReST**.*
5. *Desserts and Cheese are not distinguished in Asian and Western style.*

t_1	*pork*	*boiled*	*Asian*
t_2	*pork*	*grilled*	*any*
t_3	*pork*	*fried*	*any*
t_4	*pork*	*stewed*	*any*
t_5	*beef*	*boiled*	*Asian*
t_6	*beef*	*grilled*	*any*
t_7	*beef*	*fried*	*any*
t_8	*beef*	*stewed*	*any*
t_9	*veal*	*boiled*	*any*
t_{10}	*veal*	*grilled*	*any*
t_{11}	*veal*	*fried*	*any*
t_{12}	*veal*	*stewed*	*any*
t_{13}	*venison*	*boiled*	*any*
t_{14}	*venison*	*grilled*	*any*

t_{15}	*venison*	*boiled*	*Asian*
t_{16}	*venison*	*grilled*	*any*
t_{17}	*fowl*	*fried*	*any*
t_{18}	*fowl*	*stewed*	*any*
t_{19}	*fowl*	*boiled*	*Asian*
t_{20}	*fowl*	*grilled*	*any*
t_{21}	*fish*	*fried*	*any*
t_{22}	*fish*	*stewed*	*any*
t_{23}	*fish*	*boiled*	*any*
t_{24}	*fish*	*grilled*	*any*
t_{25}	*fish*	*fried*	*any*
t_{26}	*fish*	*stewed*	*any*
t_{27}	*fish*	*boiled*	*any*
t_{28}	*hard cheese*	*grilled*	*any*

t_{29}	*hard cheese*	*boiled*	*Asian*
t_{30}	*hard cheese*	*grilled*	*any*
t_{31}	*soft cheese*	*fried*	*any*
t_{32}	*soft cheese*	*stewed*	*any*
t_{33}	*soft cheese*	*boiled*	*Asian*
t_{34}	*goat cheese*	*grilled*	*any*
t_{35}	*goat cheese*	*fried*	*any*
t_{36}	*goat cheese*	*stewed*	*any*
t_{37}	*blue mold cheese*	*boiled*	*any*
t_{38}	*blue mold cheese*	*grilled*	*any*
t_{39}	*blue mold cheese*	*fried*	*any*
t_{40}	*fruit desert*	*stewed*	*any*
t_{41}	*aromatic dessert*	*boiled*	*any*
t_{42}	*ice cream*	*grilled*	*any*

⇒ ***ReST*** ***| ReST | = 42***

6.3 Application Conditions and Experimentation Plan

Available Resources

- 3 human experts ***e***e_1, e_2, e_3
- ***ReST*** with 42 test cases { t_1, ..., t_{42} }
- server application ***TestMeToo*** to perform the TURING Test (cf. next session)

Desired outcome: Answers to the following questions

1. *Does a **VKB** contribute to the validation sessions in an increasing degree with an increasing number of validation sessions?*
2. *Does the **VKB** contribute valid knowledge (best rated solutions) in an increasing degree with an increasing number of validation sessions?*
3. *Does the **VKB** skim the human expertise in an increasing degree with an increasing number of validation sessions?*
4. *Do the **VESA**s really model their human origin in an increasing degree with an increasing number of validation sessions?*

Session Plan

session #	e_1	e_2	e_3	$VESA_1$	$VESA_2$	$VESA_3$	$ReST$
1	+	+	+	-	-	-	$ReST_1 = \{ t_1, ..., t_{28} \}$
2	⊕	+	+	+	-	-	$ReST_2 = \{ t_{15}, ..., t_{42} \}$
3	+	⊕	+	-	+	-	$ReST_3 = \{ t_1, .., t_{14}, t_{29}, .., t_{42} \}$
4	+	+	⊕	-	-	+	$ReST_4 = \{ t_i : t_i \bmod 3 \neq 0 \}$

+ *takes part in the sessions*
– *takes **not** part in the sessions*
⊕ *takes part in solving and rating session only for being compared with its **VESA***

The result of the ***i** -th* session are VKB^i, $VESA_1^{\,i}$, $VESA_2^{\,i}$, and $VESA_3^{\,i}$.

For a fair evaluation of the usefulness of ***VKB***, the intersection of test data in ***VKB*** and ***ReST*** (***EK*** = external knowledge) needs to be considered in each session:

- $EK_1 = \emptyset \cap ReST_1 \quad = \emptyset \quad |EK_1| = 0$
- $EK_2 = \Pi_1(VKB^1) \cap ReST_2 \quad = \{t_{15}, ..., t_{28}\} \quad |EK_2| = 14$
- $EK_3 = \Pi_1(VKB^2) \cap ReST_3 \quad = ReST_3 \quad |EK_3| = 28$
- $EK_4 = \Pi_1(VKB^3) \cap ReST_4 \quad = ReST_4 \quad |EK_4| = 28$

Evaluation of the Sessions

After the session # ***i*** we determine

- the number a_i of cases from $VKB^{\,i-1}$, which were subject of the rating session and relate it to $|EK_i|$: $A_i := a_i / |Ek_i|$
- the number b_i of cases from $VKB^{\,i-1}$, which provided the optimal (best rated) solution and relate it to $|EK_i|$: $B_i := b_i / |Ek_i|$
- the number c_i of cases from $VKB^{\,i-1}$, for which a new solution has been introduced into VKB and relate it to $|EK_i|$: $C_i := c_i / |Ek_i|$
- the number d_i of solutions and ratings, which are identical reflections of e_{i-1} and $VESA_{i-1}$ and relate it to the number of required solutions and ratings: $D_i := d_i / 2 * |ReST_i|$

Answers to the vacant questions can expressed as

1. *Does a VKB contribute to the validation sessions in an increasing degree with an increasing number of validation sessions:* $A_4 > A_3 > A_2$?
2. *Does the VKB contribute valid knowledge (best rated solutions) in an increasing degree with an increasing number of validation sessions:* $B_4 > B_3 > B_2$?
3. *Does the VKB skim the human expertise in an increasing degree with an increasing number of validation sessions:* $C_4 < C_3 < C_2$?
4. *Do the **VESA**s really model their human origin in an increasing degree with an increasing number of validation sessions:* $D_4 > D_3 > D_2$?

7 Implementation Aspects

Software for the ***Turing Test*** along with the management of ***VKB*** and ***VESA***

- ***TestMeToo*** (= ***Test*** *Case Experi**Me**ntation **Tool***)
  - server application
  - available at http://www.virtual-land.de/zope/testmetoo
  - implemented with an Open Source tool ***Zope***
    - high-performance object-oriented platform for building dynamic Web applications
    - integrates
      - Web server
      - FTP server
      - Object Oriented data bank
    - enjoys a GNU Public License
    - implemented with ***Python***
      - Open Source Pearl- or PHP – like script language for Web applications.

Zope

- highly maintained product by ***Zope Corporation***
- combines benefits of both commercial and Open Source software:
  - special license of the ***Zope*** Corporation: ***ZPL*** (Zope Public License)
    - products developed within the ***Zope*** framework are owned
      - half by the developer
      - half by ***Zope*** Corporation
    - Both the developer and ***Zope*** Corporation have the right to sell it or to provide it for free.
  - developers sell commercial applications based on the Open Source framework
    - ⇒ they have a high interest in providing a stable base
    - ⇒ ***Zope*** developments are usually moderated
    - ⇒ developers follow the given (or discussed) standards
  - ⇒ developers and applicants form a huge validation panel and validate the Open Source Tool for free

Pros and Cons of the Open Source decision for ***TestMeToo***

- ☺ free of charge
- ☺ source goes through "many hands" and is evaluated frequently
- ☺ ∃ user groups, mailing lists etc. for discussions, suggestions of updates and bug fixes
- ☺ Other products may be interfaced (even integrated) seamlessly, since the interfaces of the Open Source solutions can be adopted to new needs.
- ☺ comments are in the code itself ⇒ no external documentation needed
- ☺ existing standards are followed (unwritten law)
  Python, e.g., exports and imports standardized XML without problems

- ☹ (external) documentation often
  - stays behind the code development
  - is of bad quality

  (as like with ***Zope***, e.g.)
- ☹ maintenance and test might be costly and inefficient
- ☹ no guarantee, that a function of the software will still be available (or working) in the next version
- ☹ many developers represent many different views on the *What* and *How* of program development
  - ⇒ discussions about minor issues paralyze the development
  - ⇒ especially, if ∃ conflicting standards for the same problem

8 Summary, conclusion, upcoming research and development

1. Dependability of intelligent systems is corresponds with the validity of their knowledge base
2. The only source of domain knowledge are often human experts
3. AI system validation technologies so far wasted this time consuming, expensive, and not always undependable resource
4. The concept of ***VKB*** is the key to use this resource more efficiently towards dependable systems
5. While ***VKB*** aims at modeling the human experts‘ collective and most accepted (best rated) knowledge, the ***VESA*** concept aims at modeling the human experts itself
6. At some point (after learning an appropriate model) ***VESA*** allows the replacement of this undependable resource human experts
7. Also validation concepts like ***VKB*** and ***VESA*** must be a subject of validation.
8. Experiments to empirically validate the ***VKB*** and ***VESA*** concept are subject of our actual research at Tokyo Denki University
9. Open source software is the bases to perform these experiments efficiently and with dependable tools