A Fuzzy Logic Based Set of Measures for Software Project Similarity:

Validation and Possible Improvements

Ali IDRI, ENSIAS, Rabat, Morocco

Alain ABRAN, UQAM, Montreal, Canada

7th IEEE International Software Metrics Symposium, London, 4-6 April, 2001

UQÀM

© 2001 Software Engineering Management Research Laboratory

Plan

- Introduction
- Fuzzy Logic
- Software Project Similarity Measures
- Validation
- Possible Improvements
- Conclusions and future work

© 2001 Software Engineering Management Research Laboratory

Introduction

- **Software Project similarity is one of the most important process attribute**
- **It is often used when estimating software development effort by analogy**
- **Intuitively, two software projects are not similar if the differences between their sets of attributes are obvious**
- **Analogy**

Human
Size
Color of skin
Color of eyes
Color of hair
…

≡

Software Project
Size
Complexity
Reliability
Analyst competence
…

© 2001 Software Engineering Management Research Laboratory

Sheppered et al. (1997)

$$d(P_1, P_2, V) = \frac{1}{\sum_{v_j} d_{v_j}(P_1, P_2)}$$

$$d_{v_j}(P_1, P_2) = \begin{cases} (v_j(P_1) - v_j(P_2))^2 \\ 0 \quad \text{if } v_j(P_1) = v_j(P_2) \\ 1 \quad \text{if } v_j(P_1) \neq v_j(P_2) \end{cases}$$

Critics

- Euclidean distance is used when the attributes are measured in at least an **interval** scale
  - Most of the software attribute are measured in an **ordinal** or **nominal** scales (COCOMO'81, COCOMOII, Function Points,…)
- The equality distance is used when the values are classical intervals rather than fuzzy sets
- The equality distance is not precise and can give great difference when estimating effort for two similar projects (**Idri, Abran. 7th FT&T, Atlantic City, 2000)**

© 2001 Software Engineering Management Research Laboratory

Objective

- To measure the similarity between software projects when they are described by linguistic values such as **‘very low’, ‘high’, ‘complex’, ‘excellent’**…

- Software projects are described by linguistic values => software projects are described by **vagueness** informations

- This is always the case in software measurement for an **ordinal** or **nominal** scale! But it can be presented in **ratio** or **interval** scale (fuzzy numerical number).

Our comprehension of the software is still limited

© 2001 Software Engineering Management Research Laboratory

Fuzzy Logic

- **Values between '<u>TRUE</u>' and '<u>FALSE</u>' ?**

  'The main motivation of fuzzy logic is the desire to build up a formal, quantitative framework that captures the vagueness of human knowledge via natural languages' Dubois and Prade 1991

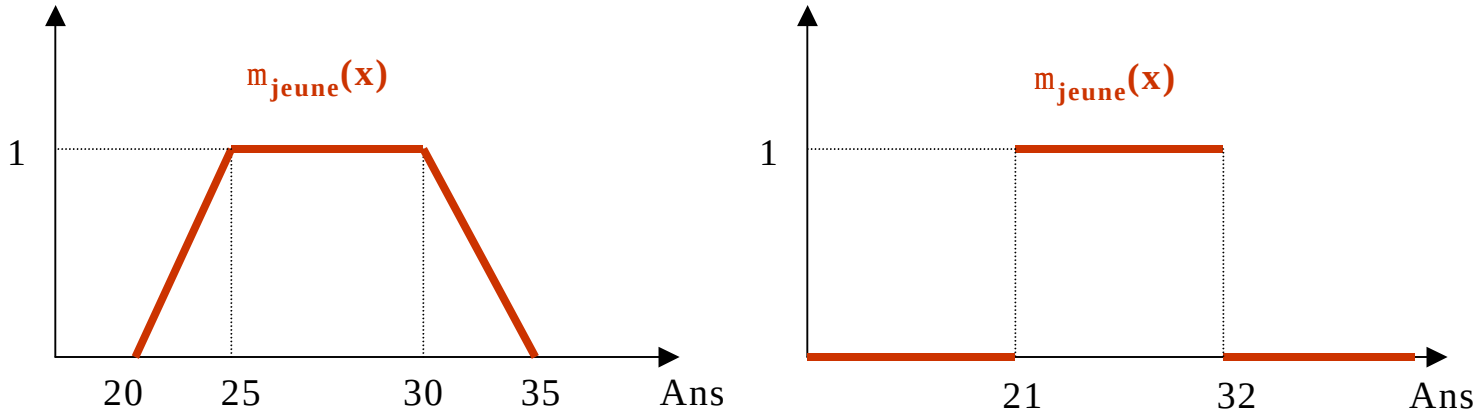
- **1965, Zadeh : Fuzzy Set**

- **1994, Zadeh : Fuzzy Logic = Fuzzy Set Theory**

- **Fuzzy Set**
  - A person X is '**young**' ?
  - Among 100 answers:
    - 50 ==> [21, 30]
    - 30 ==> [25, 30]
    - 20 ==> [22, 35]

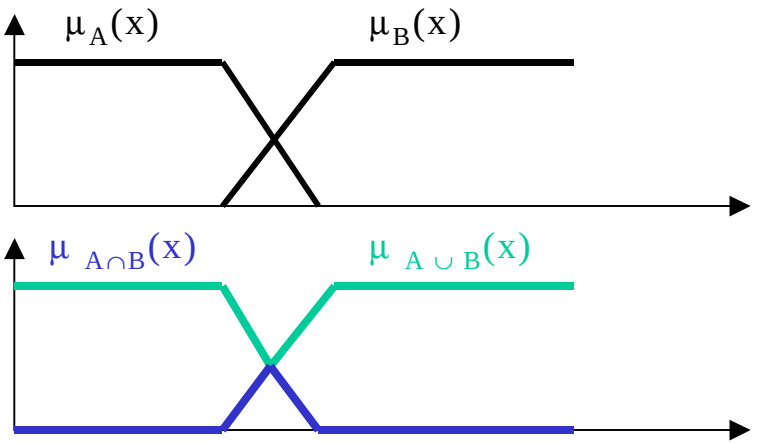
UQÀM

© 2001 Software Engineering Management Research Laboratory



Operations on fuzzy sets

- **Intersection**
  - $m_{A \cap B}(x) = \min(m_A(x), m_B(x))$
- **Union**
  - $m_{A \cup B}(x) = \max(m_A(x), m_B(x))$
- **Complement**
  - $m_{\neg A}(x) = 1 - m_A(x)$



Fuzzy Rules, Fuzzy Relations, Fuzzy Reasoning, Fuzzy Control, ...

UQÀM

© 2001 Software Engineering Management Research Laboratory

Fuzzy Logic Based Measures for Software Project Similarity

Objective:

- Evaluating similarity between two software projects P_1 and P_2 described by M linguistic variables :

$$d(P_1, P_2) = ?$$

Hypothesis :

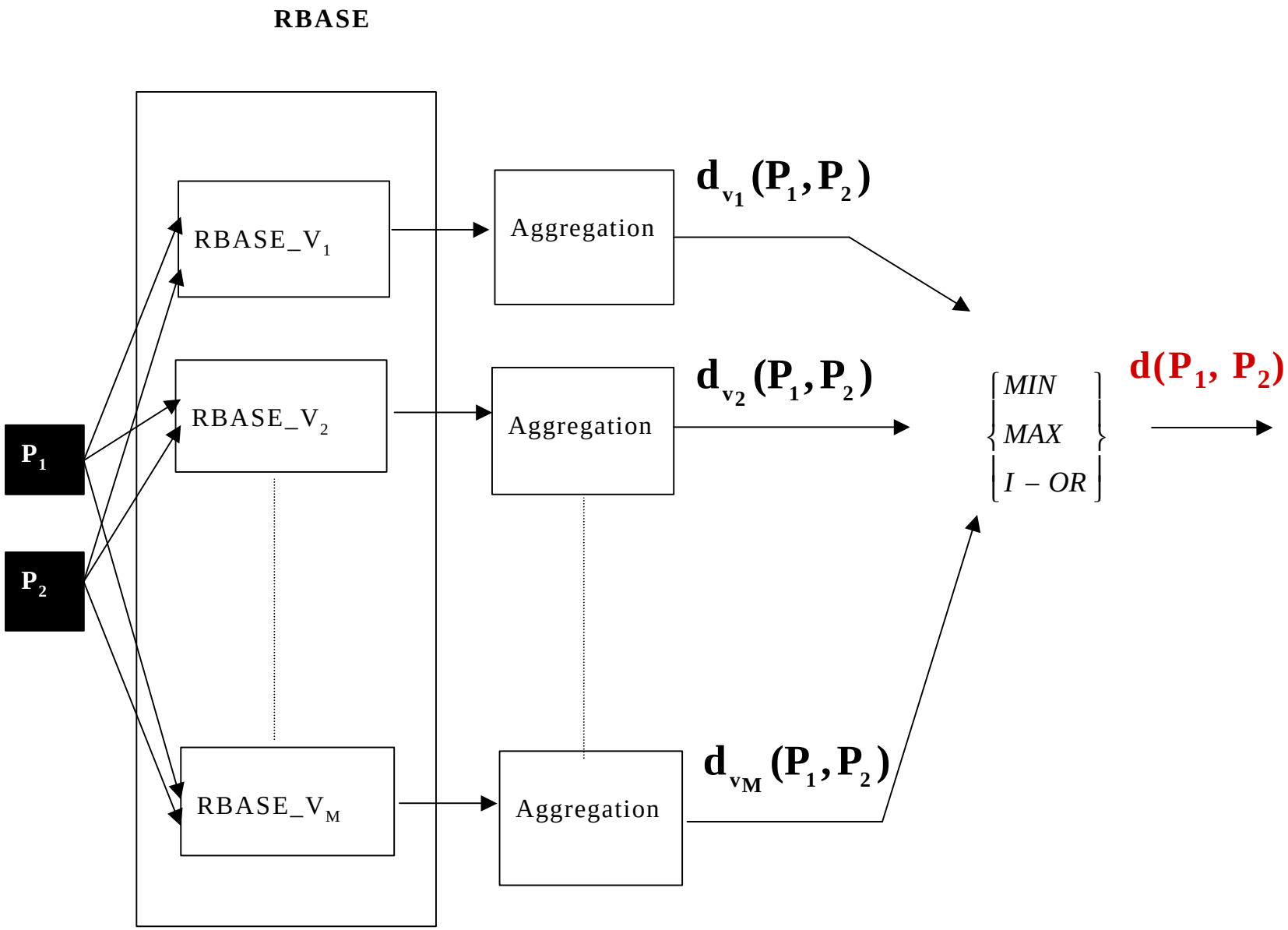
- P_1 and P_2 are described by M linguistic variables
- Each linguistic variable, V_j, is measured by linguistic values, A_k^j
- Each linguistic value is represented by a fuzzy set with a membership function $\mu_{A_k}^{v_j}$

How?

- The computing process is organized in two steps:

UQÀM

© 2001 Software Engineering Management Research Laboratory



© 2001 Software Engineering Management Research Laboratory

First Step:

- $d_{vj}(P_1, P_2)$ must express the fuzzy equality according to V_j of P_1 and P_2

- The fuzzy equality can be represented by the following fuzzy relation :

$\underset{\approx}{R}^{v_j}$ = **P_1 and P_2 are <u>approximately</u> equal according to V_j**

- Problem :

$$\grave{\imath}_{\underset{\approx}{R}^{v_j}}(P_1, P_2) = ?$$

© 2001 Software Engineering Management Research Laboratory

- $\mathbf{R}_{\approx}^{\mathbf{v_j}}$ = Combination $\mathbf{R}_{\approx,k}^{\mathbf{v_j}}$
- $\mathbf{R}_{\approx,k}^{\mathbf{v_j}}$ is the fuzzy if-then rule :

if $v_j(P_1)$ is A_k then $v_j(P_2)$ is A_k

- Combination of $\mathbf{R}_{\approx,k}^{\mathbf{v_j}}$ is called **aggregation**
- The way this is done is different for the various types of **fuzzy implication functions** adopted for the fuzzy rules $\mathbf{R}_{\approx,k}^{\mathbf{v_j}}$

$$\mathbf{R}_{\text{»}}^{\mathbf{v_j}} = \text{È}\,\mathbf{R}_{\text{»},\mathbf{k}}^{\mathbf{v_j}} = \text{È}\,(\mathbf{A}_{\mathbf{k}}^{\mathbf{j}} \text{ Ç } \mathbf{A}_{\mathbf{k}}^{\mathbf{j}}) \quad (\mathbf{A} \text{ ® } \mathbf{B} \textbf{ is } \mathbf{A} \text{ Ç } \mathbf{B})$$

$$\mathbf{R}_{\text{»}}^{\mathbf{v_j}} = \text{Ç}\,\mathbf{R}_{\text{»},\mathbf{k}}^{\mathbf{v_j}} = \text{Ç}\,(\text{Ø}\,\mathbf{A}_{\mathbf{k}}^{\mathbf{j}} \text{ È } \mathbf{A}_{\mathbf{k}}^{\mathbf{j}}) \quad (\mathbf{A} \text{ ® } \mathbf{B} \textbf{ is } \text{Ø}\,\mathbf{A} \text{ È } \mathbf{B})$$

© 2001 Software Engineering Management Research Laboratory

$$m_{R^{\vec{v}_j}}(P_1, P_2) = \begin{cases} \max_k \min(m_{A_k}^{v_j}(P_1), m_{A_k}^{v_j}(P_2)) \\ \max - \min \text{ aggregation} \\ \text{ou} \\ \sum_k m_{A_k}^{v_j}(P_1) \cdot m_{A_k}^{v_j}(P_2) \\ \text{sum - product aggregation} \end{cases}$$

$$m_{R^{\vec{v}_j}}(P_1, P_2) = \begin{cases} \min_k \max(1 - m_{A_k}^{v_j}(P_1), m_{A_k}^{v_j}(P_2)) \\ \\ \min - \text{Kleene Dienes aggregation} \end{cases}$$

© 2001 Software Engineering Management Research Laboratory

Second Step:

- $d(P_1, P_2)$ is calculated from the various individual distances $d_{vj}(P_1, P_2)$

$$d(P_1, P_2) = F(d_{v_1}(P_1, P_2), \ldots, d_{v_M}(P_1, P_2))$$

- The function F is one of the three operators: **min, max, i-or**

$$d(P_1, P_2) = \begin{cases} \min(d_{v_1}(P_1, P_2), \ldots, d_{v_M}(P_1, P_2)) \\ \max(d_{v_1}(P_1, P_2), \ldots, d_{v_M}(P_1, P_2)) \\ i-or(d_{v_1}(P_1, P_2), \ldots, d_{v_M}(P_1, P_2)) = \begin{cases} 0 & \exists k, h / d_{v_k}(P_1, P_2) = 1 \text{ and } d_{v_h}(P_1, P_2) \\ \dfrac{\prod_{j=1}^{M} d_{v_j}(P_1, P_2)}{\prod_{j=1}^{M}(1 - d_{v_j}(P_1, P_2)) + \prod_{j=1}^{M} d_{v_j}(P_1, P_2)} & \text{otherwise} \end{cases} \end{cases}$$

© 2001 Software Engineering Management Research Laboratory

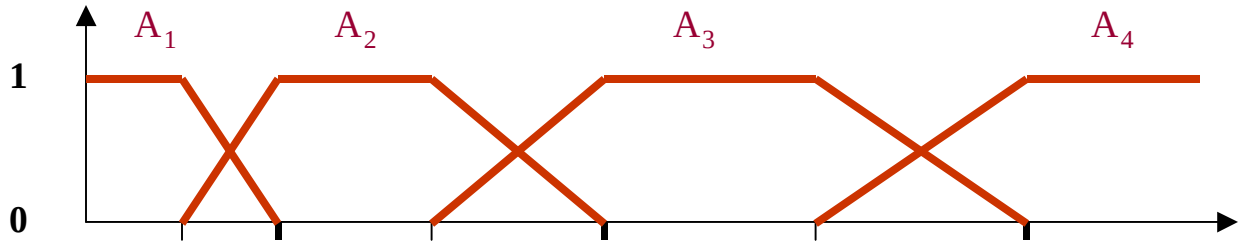
Software Measurement Validation

- **Software measurement validation is an important step in software metrics building process**
- **It allows us to choose the best measures from a large number of software measures for a given attribute**
- **No common definition : *What is a valid measure?***
- **Fenton's approach (1997):**
  - **Validation in the narrow sense**
    - *« The measure is a proper numerical characterization of the claimed attribute by showing that the representation condition is satisfied »*
  - **Validation in the wide sense**
    - *« The measure is valid in the narrow sense and it is a component of a valid prediction system »*

© 2001 Software Engineering Management Research Laboratory

Axiomatic Validation

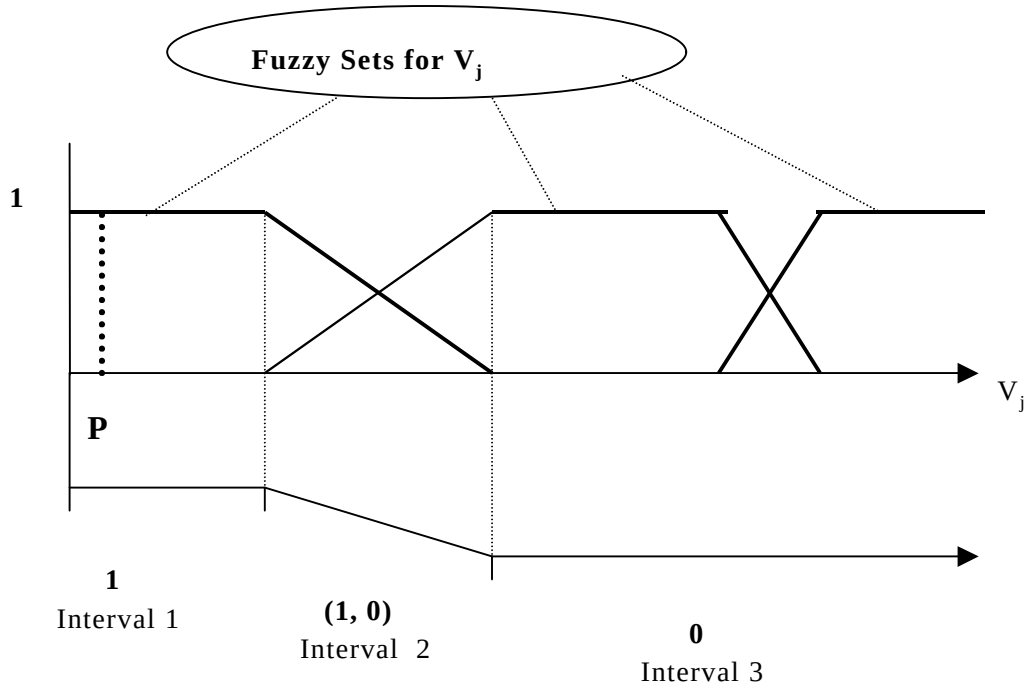
- Similarity measures satisfy the representation condition if they do not contradict any intuitive notions about the similarity of P_1 and P_2
- Our initial understanding will be codified by four axioms.
- We check whether or not the two measures satisfy these axioms
- A tuple of fuzzy sets (A_1, A_2,.., A_n) satisfy the ***normal condition*** if (A_1, A_2,.., A_n) is a fuzzy partition and each A_i is normal and convex



© 2001 Software Engineering Management Research Laboratory

Axiom 0

$$d_{v_j}(P_1, P_2) \neq 0 \;\; iff \;\; \exists \; A_k \;/\; \mu_{A_k}^{V_j}(P_1) \neq 0 \; and \; \mu_{A_k}^{V_j}(P_2) \neq 0$$



Axiom 1

$$d(P_1, P_2) \geq 0; \; d(P, P) = 0$$

© 2001 Software Engineering Management Research Laboratory

- Axiom 2

$$d(P, P_i) \leq d(P, P)$$

- Axiom 3

$$d(P_1, P_2) = d(P_2, P_1)$$

- Results of the axiomatic validation

	$d_{v_j}(P_1, P_2)$ / $d(P_1, P_2)$		
	Max-min	**Sum-product**	**Kleene-Dienes**
Axiom 0	Yes/	Yes/	No/
Axiom 1	Yes/Yes	Yes/Yes	Yes/Yes
Axiom 2	Yes/Yes	No/No	Yes/Yes if NC
Axiom 3	Yes/Yes	Yes/Yes	No/No

© 2001 Software Engineering Management Research Laboratory

Towards an empirical validation

- The measures will be valid in the wide sense if they are both valid in the narrow sense and a component of a valid prediction system
- The prediction system adopted is the estimation of software development effort by analogy
- Estimation by analogy is composed by :
  - Characterization of the projects by a set of attributes such as Reliability, Complexity, Analysts competence …
  - Evaluation of the similarity between the candidate project and each project in the database
  - Adaptation

UQÀM

© 2001 Software Engineering Management Research Laboratory

- **Intermediate COCOMO'81 database is used as an historical data**
  - **Each project is described by 17 attributes:**
    - Software size measured in KDSI
    - Project Mode is defined as Organic, semi-detached or embedded
    - 15 cost drivers related to the software environment
  - **Each cost driver is measured using rating scale of six linguistic values:**

Vey low, Low, Nominal, High, Very high, Extra-high

  - **The assignment of linguistic values uses the conventional quantization where the values are classical intervals**

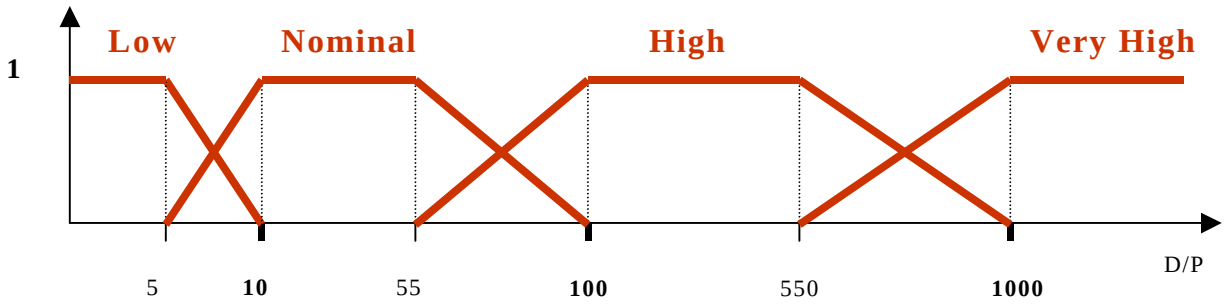
© 2001 Software Engineering Management Research Laboratory

Example: DATA cost driver

$$\frac{D}{P} = \frac{\text{Database size in bytes or characters}}{\text{Program size in DSI}}$$

Low	Nominal	High	Very high
D/P<10	10<=D/P<100	100<=D/P<1000	D/P>=1000

- It is more general
- It mimics the way in which humans interpret linguistic values
- The transition from one linguistic value to a contiguous linguistic value is **gradual** rather than **abrupt**



UQÀM

© 2001 Software Engineering Management Research Laboratory

Results for max-min and sum-product aggregations

	$d_{v_j}(P_m,P_n)$ Max-min aggregation														
	$d(P_m, P_n)$														
	Min					Max					i-or				
	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5
P_1	.521	0	0	0	0	1	1	1	1	.849	1	0	0	0	0
P_2	0	.742	0	0	0	1	1	1	1	1	0	1	0	0	0
P_3	0	0	.659	0	0	1	1	1	1	1	0	0	1	0	0
P_4	0	0	0	.849	0	1	1	1	1	1	0	0	0	1	0
P_5	0	0	0	0	.897	.849	1	1	1	1	0	0	0	0	1

	$d_{v_j}(P_m,P_n)$ Sum-product aggregation														
	$d(P_m, P_n)$														
	Min					Max					i-or				
	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5
P_1	.501	0	0	0	0	1	1	1	1	.807	1	0	0	0	0
P_2	0	.660	0	0	0	1	1	1	1	1	0	1	0	0	0
P_3	0	0	.550	0	0	1	1	1	1	1	0	0	1	0	0
P_4	0	0	0	.744	0	1	1	1	1	1	0	0	0	1	0
P_5	0	0	0	0	.815	.807	1	1	1	1	0	0	0	0	1

UQÀM

© 2001 Software Engineering Management Research Laboratory

Min operator

- It expresses the '**and**' logical operation

$$d(P_m, P_n) = 0 \Leftrightarrow \exists v_j / \; d_{v_j}(P_1, P_2) = 0$$

- $d(P_m,P_n)=0$ if $n \neq m$

  Explication: often in the COCOMO'81 database, two projects have at least one variable for which the associated linguistic values are different

- $d(P_m,P_m) \neq 1$!!

  If Normal Condition $d(P_m,P_m) \geq 0.5$

- $d(P_m,P_n)$ using max-min aggregation is different of $d(P_m,P_n)$ using sum-product aggregation

$$\left| d(P_m, P_n)_{max-min} - d(P_m, P_n)_{sum-product} \right| \leq \frac{1}{8}$$

© 2001 Software Engineering Management Research Laboratory

Max operator

- It expresses the '**or**' logical operation

$$d(P_m, P_n) \neq 0 \Leftrightarrow \exists v_j / \; d_{v_j}(P_1, P_2) \neq 0$$

- $d(P_m, P_n) \neq 0$ if $n \neq m$

  Explication: often in the COCOMO'81 database, two projects have at least one variable for which the associated linguistic values are the same

- $d(P_m, P_m) = 1$

- $d(P_m, P_n)$ using max-min aggregation is different of $d(P_m, P_n)$ using sum-product aggregation

$$\left| d(P_m, P_n)_{\max - \min} - d(P_m, P_n)_{\text{sum} - \text{product}} \right| \leq \frac{1}{8}$$

© 2001 Software Engineering Management Research Laboratory

I-or operator

- Between the 'and' and the 'or' logical operations

$$d(P_m, P_n) = 0 \Leftrightarrow \exists v_j / \; d_{v_j}(P_1, P_2) = 0$$

$$d(P_m, P_n) = 1 \Leftrightarrow \begin{cases} \exists v_j / \; d_{v_j}(P_1, P_2) = 1 \\ \forall v_j / \; d_{v_j}(P_1, P_2) \neq 0 \end{cases}$$

- $d(P_m, P_n) = 0$ if $n \# m$

  Explication: Like the case of min operator

- $d(P_m, P_m) = 1$

- $d(P_m, P_n)$ using max-min aggregation is different of $d(P_m, P_n)$ using sum-product aggregation

- This difference is not obvious because
  - The $\left| d(P_m, P_n)_{max-min} - d(P_m, P_n)_{sum-product} \right| \leq \frac{1}{8}$
  - The i-or function is continuous

© 2001 Software Engineering Management Research Laboratory

Possible Improvements by using Linguistic quantifiers

- **Human discourse uses a large number of linguistic quantifiers**
- **Zadeh distinguishes between two classes:**
  - Absolute linguistic quantifiers
  - Proportional linguistic quantifiers (most, few, at least, at most,…)
- **Yager has distinguished three categories of proportional quantifiers:**
  - RIM quantifiers (**most**, **at least** a,…)
  - RDM quantifiers (**few**, **at most** a,…)
  - RUN quantifiers (**about** a)

© 2001 Software Engineering Management Research Laboratory

- **In this work, we have used only two RIM quantifiers '<u>all</u>' and '<u>there exist</u>' to combine the individual distances** $d_{v_j}(P_m,P_n)$

- **Critics:**
  - The **'all'** and the **'there exist'** quantifiers are not always a good combination
  - In many situations, other linguistic quantifiers can be useful such that **'most'**, **'many'**, and **'at least** a**'**
  - We must take into account the importance of the variables describing the projects
  - The **i-or** operator has no clear natural interpretation

© 2001 Software Engineering Management Research Laboratory

Solution

Evaluation of the $d(P_m,P_n)$ by aggregating the individual distances using RIM linguistic quantifiers

(Idri and Abran, IFSA/NAFIPS Conference, Vancouver, 25-28 July, 2001)

$$d(P_m,P_n) = \begin{cases} \text{most of } d_{v_j}(P_m,P_n) \\ \text{many of } d_{v_j}(P_m,P_n) \\ \text{at least four of } d_{v_j}(P_m,P_n) \\ \cdots \end{cases}$$

© 2001 Software Engineering Management Research Laboratory

Conclusions and Future work

- **We have developed and validated a set of similarity measures. These measures are also applicable when the variables are numeric**
- **From an axiomatic validation, we have retained two measures for the individual similarities:**
  - Are the four axioms represent an exhaustive list of all required properties?
  - What about the transitivity of $d(P_1, P_2)$?
- **The empirical validation of estimation effort by analogy must be achieved:**
  - For the individual distance, we use the two retained measures
  - For the overall distance, we use RIM linguistic quantifiers

© 2001 Software Engineering Management Research Laboratory

- **Can I use our measures for prediction of Size, Reliability, Maintenability,…?**
- **Fuzzification of the software measurement theory**
- **Building prediction systems that satisfy Soft Computing:**
  - Tolerance of imprecision (Fuzzy Logic)
  - Learning (Neural Networks)
  - Uncertainty (Belief networks, genetic algorithms,…)

© 2001 Software Engineering Management Research Laboratory

UQÀM

© 2001 Software Engineering Management Research Laboratory