Ontology-driven Improvement of Business Process Quality

Alexandra Galatescu Taisia Greceanu
National Institute for R&D in Informatics
8-10 Averescu Avenue, 011455, Bucharest, ROMANIA
{agal, gresta}@ici.ro

Content

Application: Process Quality Improvement (PQI)

- A <u>continuous and incremental analysis and improvement</u> of the *processes and work flows* within and between organizations (*vertical* (hierarchical) processes or *horizontal*)
- A <u>less radical reengineering</u> of the enterprise processes;
- A <u>team-based process</u> (members have different specializations: management, technical, economic, marketing, human resources, etc).

Motivated by:

- Discrepancy between the customers' requirements and the offered products or services,
- Instability of the enterprise processes,
- Requirements for new types of products,
- High costs of the activity/ production,
- Too many losses,
- Low productivity, etc.

Methodologies: TQM (Total Quality Management) and Taguchi method

- **Addressed processes**: organizational or technological processes, processes for product design, for marketing, for delivery, etc
- **Main objectives:** to help the team for
  - analysis of the existing processes, according to the improvement objectives,
  - decision on the process change.
- **Additional targets:** to help for
  - organization of the team for PQI;
  - collection and organization (during brainstorming sessions) of ideas on the *process to analyze*, on the *improvement objectives*, on the *representation of the existing process*, on the *quality characteristics*, on the *changes* to apply on the existing process, etc.
- **Conceptual tools:** activities and structures for analysis of existing processes, by :
  - Building process flowcharts; data collection sheets; graphical charts for statistical analyses *(TQM);*
  - Planning, designing, executing experiment matrices *(Taguchi method);*
  - Analysis of the experiment results: effect on factors and interactions on the quality characteristics and costs; calculation of the loss function *(Taguchi method);*
  - Organization of the team *(TQM, Taguchi method);*
  - Brainstorming sessions: *verbal structures* (mainly cause-effect diagrams, structures with ideas, affinity diagrams); *decision-making tools* (voting on ideas) *(TQM, Taguchi method)*.
- **TQM and Taguchi method are complementary**:
  - TQM: (1) analyzes only the quality characteristics; (2) the causes of the process instability and the necessary changes in the process are informally stated (during the brainstorming); (3) these causes must be removed from the process, although there are situations when their removal is not possible.
  - Taguchi method: (1) analyzes the factors which impact on the values of the characteristics; (2) the proposed changes in the process are technically and economically motivated; (3) the main target is to diminish the impact of the causes (which must not be necessarily removed).

Limits and unsolved problems in existing software for PQI

<u>Software products for PQI</u>

Pathmaker (SKYMARK), Memory Jogger (GOALQPC), Solutions-PROSPER and PRO-QMS (DSS Infotech), Qualitek (NUTEK), Microsoft Visio, DataLyzer Spectrum (Stephen Computer Services, USA), SQCpack (Quality Management Products, Canada), ECHIP (ECHIP Inc.), JMP (SAS Institute Inc.), knowledge bases with "how-to", guidebooks, tool libraries, etc.

<u>Limits and unsolved problems</u>

- they do not provide guidance and the users must be specialists in TQM, Taguchi method and mathematical statistics;
- the users have to manage symbols with an informal semantics that cannot be compared or transferred between different types of diagrams and structures;
- they do not allow the description and automatic use of *explicit correlations between*
  - *the activities* and the objects which describe them or participate in their execution.
  - *objects and their quality characteristics* (which are statistically analyzed).

  These correlations are supposed to be in the user's mind and cannot be stored and automatically analyzed.
- they do not encourage the use of a common vocabulary between the members of the team (usually, with different specializations). The ideas are collected in natural language and cannot be automatically compared. To reach a final decision, many (virtual) discussions are needed.
- they implement either TQM methodology or Taguchi method. There is no tool which integrates them both.

Types of ontologies in PQI (Process Quality Improvement) system

- ***PQI*** (*application*) ***ontology*** (predefined) - describes, categorizes and constrains concepts and relationships in the *PQI process* (which integrates the TQM and Taguchi methodologies);

  <u>Main use</u>: for the *interface of the system,* dynamically built relying on the concepts in PQI ontology, during the user's navigation throughout the two methodologies.
- a ***domain*** (*business*) ***ontology*** (created by the user) - describes the *enterprise processes*, the analyzed objects, the objects' characteristics, the technical factors that impact on object/ product quality, etc.

  <u>Main use:</u> *common vocabulary* between the members in the PQI team.

<u>Main benefits from the two ontologies</u> for the PQI system:

- the <u>semantics of the concepts in the PQI ontology</u> is explicit (*outside the code), formalized* and represented *at the analysis and design time* of the PQI system, before its execution:
  - ➔ system interface is dynamically built during the system execution
  - ➔ the user's actions are dynamically checked and interpreted, depending on the user's choices;
  - ➔ PQI process is easily changed/ extended (e.g by adding Taguchi method to TQM methodology);
- the explicit <u>semantics of the concepts in the domain ontology</u> helps for:
  - ➔ *common vocabulary* between the members of the team
  - ➔ *semantics-based analysis* of the domain process (e.g. by comparison and merge of process flowcharts, expression, comparison and grouping of ideas about the process);
  - ➔ *semantics-based access of the data in the repository* (by means of concepts in the domain ontology),
  - ➔ *semantics-based integration of the PQI steps and operations* (which share the concepts in the domain ontology);
- the <u>encoded reasoning</u> of the system is substantially <u>minimized</u> (due to PQI ontology).

Need for ontology integration

Need for:

- ❑ <u>execution of certain PQI steps/ operations</u> (described in PQI ontology) <u>depending on the results of previous steps / operations</u> (results described in the domain ontology); E.g.
  - ➢ definition of a process, operations, objects and characteristics in the domain ontology (Step 3) only for a domain already defined (in Step 2)
  - ➢ modification of the existing process (Step 14) only if the new process is stable and able of further improvement (results of Steps 12, 13);

- ❑ <u>easy and quick learning of the system</u> by similar semantics for the representation of both application (PQI system) and business (analyzed process)
  - ➢ *same types of concepts/ relationships*,
  - ➢ a *uniform representation* for the user interface (in PQI ontology) and the analyzed process (in domain ontology).

  In most existing applications, the ontologies are used only for business description.

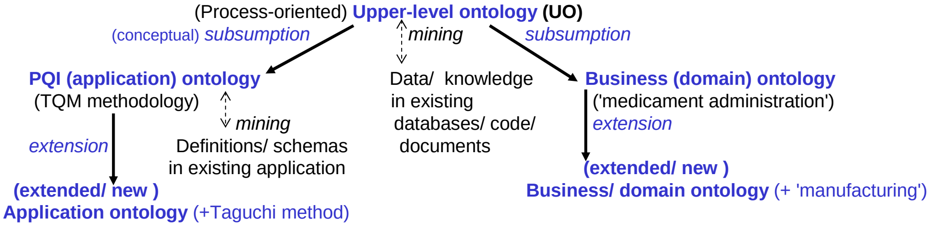
- ❑ <u>integration of the interface</u> (described in PQI ontology) <u>with the analyzed process</u> (described in the domain ontology); E.g.
  - ➢ execution of PQI operations is possible only after the selection of the mandatory concepts in the analyzed process;
  - ➢ structures with meta-schema described in PQI ontology and content composed of concepts in the domain ontology;

Ontology integration in the PQI system

<u>Alternative solutions for ontology integration</u>

- ❑ an ***upper-level ontology;*** or
- ❑ a ***translation algorithm*** between the concepts and rules in the two ontologies. ➔ <u>Disadvantages;</u> it is mostly encoded and the conceptual integration is accomplished only at the run time of the PQI system.

<u>Business and application integration in PQI system</u> (thick arrows: already implemented functions)



<u>Benefits from ontology integration</u>

- ❑ <u>same types of concepts, relationships and axioms</u> in UO used for representing both the PQI (application) ontology and a business/ domain ontology
- ❑ <u>application ontology can be extended</u> or new application ontologies can be integrated with the existing ones (e.g. in PQI system, first was built the ontology for TQM methodology, extended with Taguchi method)
- ❑ an initial <u>domain ontology can be extended with other domains</u>, analyzed using the same PQI software and repository (e.g. TQM methodology has been tested for 'medicament administration' domain and Taguchi method for the 'manufacturing' domain);
- ❑ besides its direct subsumption from UO, an application ontology can also be built by <u>mining the existing applications for definitions and schemas</u> (according to the concepts and axioms of UO).
- ❑ business ontology could be created by <u>mining the existing databases, code, documents</u> and by the conversion of the extracted information/ knowledge according to the concepts and axioms in UO.

Ontology languages and editors for process representation

Existing applications of process-oriented ontologies:

- ❑ for enterprise process management;
- ❑ lately, for managing Web service-oriented processes.

Not satisfied requirement for enterprise process representation: representation of:

- ❑ the enterprise dynamics (changes/ improvements of enterprise processes)
- ❑ enterprise integration and interoperability (organizational, technological, informational, etc)

Limits of the existing ontology languages and editors:

❑ same orientation as most conceptual models: most of them are **object-oriented:**

- ➔ only object (entity)-like concepts;
- ➔ explicit relationships only between objects and attributes,
- ➔ only explicit specialization/ generalization relationships between objects (➔ inheritance);
- ➔relationships between objects and events upon them (encoded methods);

❑ ontology editors and management tools (e.g. Protégé, OilEd, OntoBroker/ OntoEdit, KAON, Ontolingua, OntoWeb, OntoSaurus, etc) are **not process-oriented**, i.e.

- ➔ without semantic separation of the *process and activity-like concepts* from the *object (entity)-like* concepts involved in the process execution (it is usually encoded)
- ➔ no capability for the explicit representation, decomposition and interpretation of the processes and activities ➔ designers should devise their own types of concepts and relationships for process description (possibly, complying with a process-oriented language (e.g. PSL, BPEL, etc))
- ➔ no reasoner and management functionality for processes (should be built by the developer)

Recommended business process ontology: PSL (Process Specification Language) (initiated by NIST-National Institute of Standards and Technology), because it can be logically integrated with KIF (Knowledge Interchange Format) (for knowledge description and exchange).

Types of concepts and relationships in the upper-level ontology

Process

(composed-of) **Complex Operations** → **new concept and relationship**

(composed-of) **Atomic operations**

(composed-of) **Atomic Operations**

(described-by) **Objects** → **relationship outside the code**

(described-by) **(Object) Characteristics** (attributes)

(described-by) **Characteristics (attributes) of Operation** (e.g state)

Conceptual models

Conceptual models and ontology languages **Proposed representation**

Additional information in conceptual models and in proposed upper-level ontology

❑ sequential order and preconditions of the atomic operations in a process;

New elements in the process representation using the upper-level ontology

❑ explicit correlation of atomic operations and objects in ***ontology-based sentences*** (operation description structures), inspired from the syntax of the simple sentence in natural language;

❑ complex operations which:

- ➢ compose the process and are composed of atomic operations;
- ➢ difference from subprocess: are represented by ontology-based sentences, inspired from the syntax of the compound/ complex sentence in natural language;

❑ atomic/ complex operations are preceded by their 'modality' ('must', 'may') and by 'pre-conditions';

❑ in a complex operation, the atomic operations are correlated by interoperation 'connectors' (e.g. 'case', 'then', 'must repeat', etc),), which can be used for the procedural description of the process.

NOTE PQI and domain ontologies differ by the instances of these basic concept types (by particular processes, operations, objects, characteristics).

Basic concepts in the upper-level ontology

- ***process***: controlled sequence of operations (e.g. operations that compose the process 'medicament administration' in the domain ontology);
- atomic or complex (decomposable) ***operation***: action or sequence of actions, e.g atomic actions in the domain ontology are 'order', 'check', 'supervise', etc. Each atomic operation (e.g. 'med order') is described by objects that participate in its execution (similarly to the object-like parameters that compose the operation signature in programming languages);
- ***object***: entity, e.g. 'patient', 'medicament';
- ***characteristic of an object*** (e.g. an attribute like 'patient age') ***or of an operation*** (e.g. an object standing for the operation's determiner like in '*med* order', '*pharmacy* check' or an operation attribute like in '*wrong* order', '*failed* check');
- ***factor***, element which impacts on the values of one/ more characteristics of an object. Used only in the implementation of Taguchi method and represented only in the domain ontology.

Ontology-based simple sentences

Basic elements (inspired from the syntax and semantics of natural language)

- 'object', 'operation', 'characteristic'- corresponding to 'nouns', 'verbs', 'adjectives' or 'adverbs' in NL
- (universal/ existential) quantifiers and the plural of the objects;

➔ the sentence unifies object-like concepts with different syntactic roles in the description (and execution) of the main operation (verb) in the sentence.

Formal representation: a **star** (conceptual) **graph** with the linear form

(OPERATION)	**-** *an atomic operation, standing for the predicate in NL sentence*
AGNT ∀[AGENT]	- *subject(s) in the active voice*
PTNT ∃/∃?[Object_Type$_1$:C/D{}]	- *direct object(s), i.e. the object(s) upon which OPERATION acts*
RCPT ∃/∃?[Object_Type$_2$:C/D{}]	- *indirect object(s), i.e. the recipients of the results of OPERATION*
<*preposition role*> ∃/∃?[Object_Type$_3$]	- *prepositional object(s);*
<*adverb role*> ∃/∃?[Object_Type$_4$]	- *adverbial modifier(s) (i.e. operation modifier);*

where

- ***nodes*** are objects or operations; and
- ***links*** are roles, standing for 'object-operation' links or for links between active objects and the attributive objects that describe them.
- ***quantifiers*:** universal quantifier ∀ replaces the indefinite pronouns 'any', 'all', 'every', 'each' in NL; the two existential quantifiers: ∃, meaning compulsory existence ('must exist') and ∃?, meaning optional existence ('may exist'), replace the definite or indefinite articles in NL;
- ***plural*:** collective/ distributive denoted by C/ D{};
- ***preposition and adverb roles*** with acronyms: RSLT (result), INST (instrument), LOC (location), SRC (source), etc; with a preposition, conjunction, adverb as linguistic synonym; with disjunctive semantics (which *eliminates the ambiguities* in NL).

➔ **Benefits**: *domain independent processing of the operations* (code uses roles instead of domain-specific types of objects/ attributes).

Special types of ontology-based simple sentences

Sentences around generic operators for:

- semantic relations between objects/ operations (e.g. holonymy, hypernymy, synonymy, antonymy etc);
- object definition using attributes or other objects;
- dynamic qualification of objects or operations.

Sentences representing semantic relations in object and process representation

- Noun meronymy/ holonymy ➔ Object fragmentation / aggregation
- Noun hyponymy/ hypernymy ➔Object specialization/ generalization
- Noun synonymy ➔ Class synonymy (identical structures for classes with different names)
- Noun antonymy ➔ Class antonymy (classes with opposite meanings)
- Noun homonymy (identical form and different meanings) ➔ Class homonymy (same name, diff. structures)
- Verb meronymy/ holonymy ➔ Functional decomposition / composition
- Verb hyponymy/ hypernymy ➔ Functional specialization/ generalization
- Verb synonymy ➔ Operation synonymy (identical functions for operations with different types / names)
- Verb antonymy ➔ Functional antonymy (operations with opposite functions, e.g. that undo each other)
- Verb homonymy ➔ Functional homonymy (operations with same name, but diff. functions ➔ polimorphism)
- Verb troponymy (manner relation) ➔ Operation specialization (by the manner of action)
- Verb entailment ➔ Functional, semantic, temporal entailment
- Cause-effect relations between verbs ➔ Relation between an event and the operation it stimulates

Formal examples of sentences in PQI ontology

(Object_HOLONYMY) DEST ∀[**FlowChart**] *- whole* PART1 ∃[StartPoint] *- part* PART2 ∃[Activity:C{}] PART3 ∃[DecisionPoint:C{}]..	**(Operation_ENTAILMENT)** RCPT ∀(Diagram INTERPRETATION) *-entailed* PTNT ∃(Diagram CREATION) *-entailing oper.* **(Object QUALIFICATION)** RCPT ∀[MemberID] GOAL ∃ [Responsibility]	**(Object_DEFINITION)** *- definition of Member* *RCPT* ∀ [**MemberID**] *NAME* ∃[Member Name] *LOC* ∃[Department: {Production /Marketing/...}]

Logic of the ontology-based simple sentences

Generic simple sentence for operation description (**OPERATION**) $role_1$ ∀[$Object_1$] $role_2$ ∃[$Object_2$] …… $role_p$ ∃?[$Object_p$]	$OPERATION = \lambda(x_{i1},\dots,x_{ik})\ (x_{j1},\dots,x_{j(p-k)})\ (\forall x_1)\ (Object_1(x_1) \wedge role_1(x_1)) \supset (\exists x_2)\dots(\exists x_p)\ (Object_2(x_2) \wedge role_2(x_2) \wedge \dots \wedge (Object_p(x_p) \vee NULL) \wedge (role_p(x_p) \vee NULL)) \wedge OPERATION(x_1,\dots,x_p)$ ➢*λ-expression with* $(x_{i1},\dots,x_{ik})$ *(input/ output parameters) as bound variables and a subset of objects* $(x_{i1},\dots,x_{ik}) \subset (x_1,\dots,x_p)$ ➢*'role1' is usually AGNT (subject) and NULL helps for the representation of the quantifier ∃? ('may exist').*
Generic operator for object (dynamic) qualification (**Object QUALIFICATION**) PTNT ∃[active_object_type] ROLE ∃[attributive_object_type]	(∃a)(attributive_object_type(a)) ⊃ (∃o)(active_object_type(o))(PTNT(o)∧ ROLE(a))∧ Object QUALIFICATION(o,a) ➢ *where any attributive object 'a' necessarily qualifies an active object 'o' (that directly participates in an operation).* ➢ *the attributive object can become active object in other circumstances (e.g. the attribute 'medicene cost' for the object 'patient', used as quality characteristic during the process analysis, can become the subject in the sentence (idea) 'Medicine cost is too high').*
Generic operator for object definition, (**Object_DEFINITION**) RCPT ∀ [Defined_Object_TYPE] *role1* ∃[$Object\_Type_1$] …. *rolen* ∃?[$Object\_Type_n$]	$Defined\_Object\_TYPE = (\lambda x)\ (x_1,\dots,x_n)\ (\forall x)(Defined\_Object\_TYPE(x) \wedge RCPT(x)) \supset (\exists x_1) \dots (\exists x_n)\ (Object\_Type_1(x_1) \wedge role_1(x_1)) \wedge \dots \wedge (Object\_Type_n(x_n) \vee NULL) \wedge (role_n(x_n) \vee NULL)) \wedge Object\_DEFINITION(x, x_1,\dots,x_n)$ ➢ *where 'x' is a bound variable*

Ontology-based compound/ complex sentences

Composition rules in NL

- *compound sentence* joins independent simple sentences by coordinating conjunctions (copulative, disjunctive, adversative, resultative, explanatory) or adverbs or asyndetically (without conjunctions);
- *complex sentence* correlates dependent (subordinated) sentences (clauses) to a main sentence (clause) by subordinators like conjunctions, pronouns, adverbs, etc;
- correlation between two verbs in two simple sentences by an anaphoric or generic reference, introduced by the definite article or by a pronoun in the second sentence.

Composition rules in an ontology-based *compound and complex sentence*

Atomic/ complex operations in a process (and, implicitly, the simple sentences that describe them):

- are correlated by inter-operation connectors (as *intersentential relations*):
  - E.g. for compound sentences, e.g. 'must', 'may', 'and', 'or', 'not', 'case', etc.
  - E.g., in a complex sentence, subordinating relations can be abstracted by 'if-then-else', 'dscr' (description), 'goal', 'event', 'do', 'while', 'subordinating cause or result', 'then', 'case', 'spec' (specialization), 'before', 'after', 'but' etc.
- are preceded by operation modality (implicitly represented by the connectors 'must', 'may') and operation pre-conditions;
- the correlation between operations in two sentences is represented by a coreference that correlates the coreferent concepts in two sentences (e.g. object IDs).

Use in PQI of rules and elements in compound sentences

- *Connectors, modality and pre-conditions* are used for visual guidance and automatic verification of the user's actions: checking operation precondition, operation obligativity, obligativity of the object selection before operation execution, existance of the selected objects in the repository, etc;
- *Coreferences* are used for representing the information flow in the PQI process, i.e. the transfer of information (particular concepts in domain ontology, e.g.'Current Domain', 'Current Process', 'Current Object', etc) between atomic PQI operations. These coreferences compose the working context.

Logic of interoperation connectors

Modality	PO **MUST** $Oi_1, \ldots, Oi_n$	$(\forall x)\,((PO(x) \supset Oi_1(x)) \wedge \ldots \wedge (PO(x) \supset Oi_n(x)))$
	PO **MAY** $Oi_1, \ldots, Oi_n$	$(\forall x)((PO(x) \supset (Oi_1(x) \vee NULL)) \wedge \ldots \wedge (PO(x) \supset (Oi_n(x) \vee NULL)))$
	PO (**MAY** ∧ **condition**) $Oi_1, \ldots, Oi_n$	$(\forall x)\,(((PO(x) \wedge condition) \supset (Oi_1(x) \vee NULL)) \wedge \ldots \wedge ((PO(x) \wedge condition) \supset (Oi_n(x) \vee NULL)))$
Sequence	PO **THEN** / **BFOR** O	$(\forall x)\,((PO(x) \supset O(x)) \wedge \neg (O(x) \supset PO(x)))$
	PO **AFTR** O	$(\forall x)\,(\neg(PO(x) \supset O(x)) \wedge (O(x) \supset PO(x)))$
Alternatives	**I F** condition **THEN** O_1/ **ELSE** O_2	$(\exists x)((condition(x) \supset O_1(x)) \wedge (\neg condition(x) \supset O_2(x)))$
	PO **CASE** $Oi_1, \ldots, Oi_n$	$(\forall x)\,(PO(x) \supset Oi_1(x) \vee Oi_2(x) \vee \ldots \vee Oi_n(x))$
	PO (**CASE** ∧ **condition_value**) $Oi_1 \ldots Oi_n$	$(\forall x)((PO(x) \wedge condition\ value_1) \supset Oi_1(x) \vee (PO(x) \wedge condition\ value_2) \supset Oi_2(x) \vee \ldots \vee (PO(x) \wedge condition\ value_n) \supset Oi_n(x))$
Iteration	**WHILE** condition DO O	$(\exists x)((condition(x) \supset O(x)) \wedge (\neg condition(x) \supset NULL))$
	PO **MUST REPEAT**	$(\forall x)\,(\exists y)\,(PO(x) \supset PO(y))$. **- To avoid the infinite loops, NULL will be instead of 'y' when the repetition ends.**
	PO **MAY REPEAT**	$(\forall x)\,(\exists y)\,(PO(x) \supset (PO(y) \vee NULL))$ **- A procedural end of the repetition that stops before PO expansion/ start.**
Logical Relations	PO **AND** O	$(\forall x)\,((PO(x) \supset O(x)) \wedge (O(x) \supset PO(x)))$
	PO **OR** O	$(\forall x)((PO(x) \supset (O(x) \vee NULL)) \wedge (O(x) \supset (PO(x) \vee NULL)))$
	PO **XOR** O	$(\forall x)\,((PO(x) \supset \neg O(x)) \wedge (O(x) \supset \neg PO(x)))$
	NOT O	$(\forall x)\,(\neg O(x))$ **- Execution negation. 'x' an input concept of O**
Motivation	PO **GOAL** O	$(\forall x)\,((PO(x) \supset O(x)) \wedge (\neg O(x) \supset \neg PO(x)))$
Stimulation/ Cause	EO **EVNT** / **CAUS** O	$(\forall x)((\exists e) EVNT(e) \supset O(x))$, **where** $EVNT(e) = (\lambda e)\lambda\text{-definition}(EO)[e]$ **. 'e' is a particular event/ cause and 'λ-definition (EO)' is the λ-expansion of the event/ cause operation EO.**

NOTE *x* - an individual of an output object type in the premise operation, transferred to its child operations;

$\supset$ compulsory activation of the implied operations; $\wedge$ conjunction of the operations or rules; $\vee$ (exclusive or not) disjunction of operations or rules; **NULL** is a null (ineffective) operation.

Types of concepts and relationships in PQI ontology

PQI Process (General Scenario of the PQI methodology)

(composed-of) { **Steps in scenario** (as Complex Operations)
(composed-of) **Atomic operations**
Atomic Operations
(*described-by)* { **Objects**
(*described-by)* **Characteristics (attribute) of the objects**
Characteristic (attribute) of Operation

❑ *PQI process* is described by a general scenario, composed of PQI steps (complex operations).

❑ the steps are further composed of complex or atomic operations.

❑ each atomic operation is described by objects which participate in its execution. Similarly to:

- ➢ parameters in operation signature in programming languages;
- ➢ nouns which describe the action of a verb in *natural language*.

❑ the objects are described by their characteristics (attributes);

❑ the steps and operations are controlled by pre-conditions and inter-operation connectors that state their sequential or parallel execution, obligatory or optional execution, alternation, repetition, etc (e.g connectors like 'must', 'may', 'case', 'must repeat', etc).

❑ steps and atomic operations are preceded by their modality;

❑ attributes of operations are 'name', 'description', 'type', 'form name', 'help', etc.

❑ attributes of objects are 'name', 'description', 'form name', 'retrieval condition (where condition)', 'predefined table/ query in the repository', 'help', etc.

Types of concepts and relationships in domain ontology

Process in domain

(composed-of) { **Complex Operations in process**
Atomic Operations in process
(*described-by)* { **Objects in operation (and in domain)**
(*described-by)* **(Quality) Characteristics (attribute) of the objects**
(impact-on) **Factors**
Characteristic (attribute) of Operation

❑ description of the domain-specific process implicitly refers to descriptions of the component (complex or atomic) operations;

❑ description of the atomic operation unifies the objects involved in the operation execution

❑ objects are described by their characteristics (mainly, the characteristics which are subject to analysis).

❑ the values of one or more characteristics of an object depend on certain (controllable/ uncontrollable) factors in the process (e.g. temperature, composition consistency, humidity, etc)

❑ attributes of the operations are specified in the domain ontology and, also, in the *process flowchart* (e.g operation goal, precondition, responsible person, if the operation is selected for data collection, etc).

❑ in the domain ontology, the user is also allowed to specify synonymy relationships, between operations or between objects, used for the comparison of the ideas collected from the members of the team.

Benefits from a DBMS for PQI automation

DBMS helped:

- For building the infrastructure for both ontologies in the same database, with **benefits** for:
  - ➔ integration of PQI ontology with the domain ontology (needed, for instance, when the execution of certain PQI steps depends on results of previous steps, results stored in the domain ontology);
  - ➔ integration of the PQI-specific tools (operations for the creation of diagrams, data collection sheets, statistical analyses, etc) which share concepts in the domain ontology;
  - ➔ retrieval of the objects and operations in the user interface by means of *preconditions* explicitly defined in the PQI ontology, *dynamically customized* with domain-specific concepts in the working context (current project, domain, process, operation, object, etc), selected by the user;
- With mechanisms for concept management (insert, delete, update, select), for concept correlation, for assuring the ontology consistency and physical integrity;

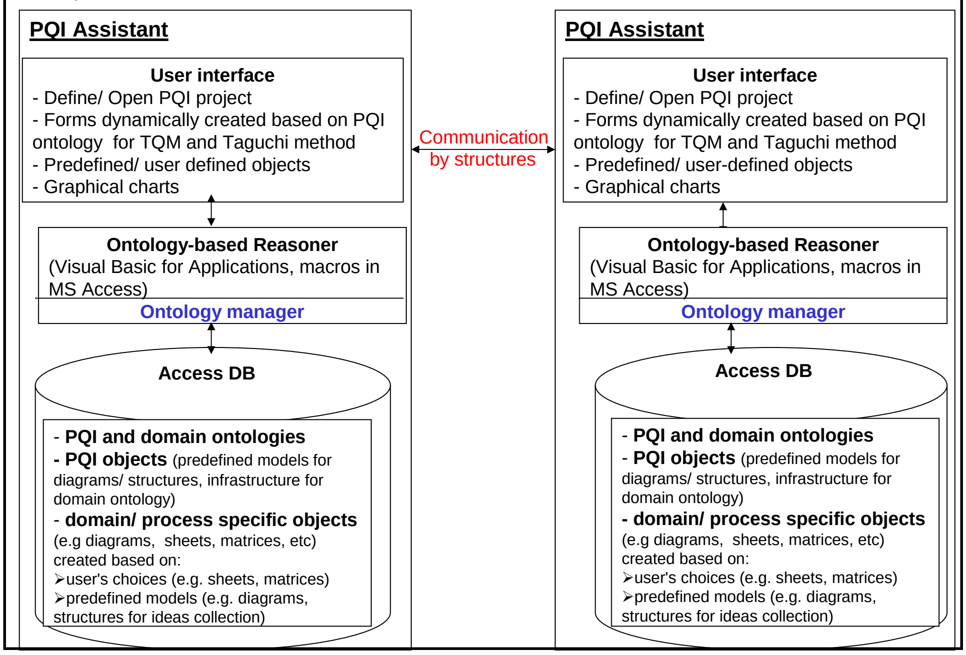
Relational DBMS for PQI system implementation

Microsoft Access in cooperation with other tools in MS Office (Word, Excel, OutLook Express) and with NetMeeting. **Benefit**: widely spread Windows platforms ➔ without additional costs for using the PQI system.

- **Repository** for storing ontologies, predefined objects for PQI (infrastructures for diagrams, ideas collection, structures for experiments), domain-specific objects (diagrams, matrices, ideas);
- **Reasoning** (in macros and Visual Basic for applications) based on PQI&domain ontologies for:
  - dynamic creation of the system interface;
  - guidance and verification of the user's actions;
  - dynamic creation of the schema for data collection sheets and for experiment matrices;
  - comparison and grouping of the ideas;
  - statistical analysis of the collected data;
  - comparison and concatenation of the process flowcharts;
  - customization of the PQI assistant, depending on the members' roles in the team, etc.

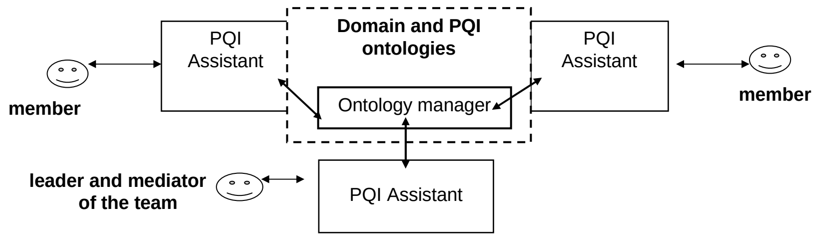
Implementation platform. Team of PQI assistants

The system can be used stand-alone or in a virtual team of PQI assistants



Ontology manager

- an intrinsic component of the PQI assistant
- it facilitates:
  - dynamic creation of the system interface, based on PQI ontology;
  - definition, navigation, extension of the domain ontology;
  - automatic classification and retrieval of the domain-specific concepts, according to the working context;
  - communication between the members of the team, relying on PQI and domain ontologies;
  - correlation, comparison and inference on members' ideas expressed using concepts in the domain ontology



Using a relational DBMS for managing sentences based on PQI and domain ontologies

Part of compound sentence describing the PQI process (general scenario)

ck	Condition	Mode	Operation Name
		MAY	Customize Scenario for Process Quality Management
		MAY	Project Scheduling
		MAY	Brainstorming Session
		MAY	BUSINESS PROCESS IMPROVEMENT BY TQM
		MUST	Step 1 - Organize BPR Team
		MUST	Step 2 - Create/ Modify/ Delete Domain and Process Definitions
		MUST	Step 3 - Create Objects and Characteristics in process
		MUST	Step 4 - Create AS-IS Process Diagram
		MUST	Step 5 - Plan Data Collection for the current AS-IS Process
		MUST	Step 6 - Check Stability of current AS-IS Process
	Process stable	MUST	Step 7 - Check Improvement Ability of current AS-IS Process
	Process unstable or unable of improveme	MAY	Step 8 - Identify Root Cause for current AS-IS Process
		MUST	Step 9 - Create and Implement current TO-BE Flowchart Diagram
		MUST	Step 10 - Plan Data Collection for the current TO-BE Process
		MAY	Step 11 - Test Changes, Create Implementation Plans
		MUST	Step 12 - Check Stability of the current TO-BE Process
	Process stable	MUST	Step 13 - Check Improvement Ability of the current TO-BE Process
		MAY	Step 14 - Standardize the current TO-BE Process
		MAY	TAGUCHI EXPERIMENTS
		MUST	Organize the Team for Experiments
		MUST	Define/ Select Domain and Process
		MUST	Define/ Select Objects and Characteristics
		MAY	Define problems to solve
		MUST	PLAN Experiments (factors, interactions, conditions)
		MUST	DESIGN Experiments
		MAY	CONDUCT (Execute) Experiments

<u>Compound sentence</u> describing the <u>PQI operation</u> ' Step 4 – Create AS-IS Process Diagram'

ent Project: lth care | Current Domain: Medicament administration | Current Process: Med Administer | Current Activity: | Cu

ck Condition	Mode	Operation Name
	MUST	SELECT current Domain and Process
	MAY	MANAGE Structure
Selected AS-IS process	MAY	Add/ Modify/ Delete the Flowchart for current AS-IS Proc
Selected Flowchart	MAY	Interpret Flowchart
Selected AS-IS and TO-BE process	MAY	Flowchart comparison
Selected domain and process	MAY	Create final flowchart and merge with it a collected flowc
Selected process	MAY	Replace flowchart for current AS-IS process with the flow
	MAY	Replace/Delete structures with ideas/ diagrams
		VIEW Structure
Selected AS-IS process	MAY	VIEW the Flowchart for the current AS-IS Process
	MAY	VIEW a selected flowchart (previously created)
Selected structure	MAY	VIEW a structure with flowchart (identified by the name o
	MAY	VIEW results from flowchart comparison
	MAY	EXPORT / IMPORT Structures
Existing export structure	MAY	Prepare Send Ideas/ Diagram (create Excel file)
Existing Excel file	MAY	Send/ Receive definitions/ structures (call OutLook Exp
Existing Excel file	MAY	Collect and view definitions/ structures (from Excel files)

<u>Simple sentence</u> for the description of the <u>PQI operation</u> 'add/modify/delete AS-IS flowchart'

File Edit View Insert Tools Window Help

Add/ Modify/ Delete AS-IS Flowchart

Current Project: Health care | Current Domain: Medicament administra | Current Process: Med Administer | Current Activity: | Current Object: | Current Inspection Sheet/ Structure/ Matrix:

Operation Name	Object Name	Object Quantifier	Execute
Add/ Modify/ Delete AS-IS Flowchart	Current Domain	MUST EXIST	Select
Add/ Modify/ Delete AS-IS Flowchart	Current Process	MUST EXIST	Select
Add/ Modify/ Delete AS-IS Flowchart	Flowchart General Description	MUST EXIST	Add/Modify/Delete
Add/ Modify/ Delete AS-IS Flowchart	Flowchart for current AS-IS process	MAY EXIST	Add/Modify/Delete
Add/ Modify/ Delete AS-IS Flowchart	Graphical view of the current flowchart	MAY EXIST	View
Add/ Modify/ Delete AS-IS Flowchart	Extracted Flowchart for the current process	MAY EXIST	Save

<u>Simple sentence</u> describing the <u>domain operation</u> 'Med order'('in 'Medicament administration' domain)(schema of next table is in PQI ontology describing the PQI object 'Domain Operation")

Current Project: Health care | Current Domain: Medicament administration | Current Process: Med Administer | Current Activity: Med Order | Current Object:

Domain	Operation Type	Object	Object Quantifier	Object Plural
Medicament admini	Med Order	Patient	MUST EXIST	just one
Medicament admini	Med Order	Medicament	MUST EXIST	several together
Medicament admini	Med Order	Physician	MUST EXIST	several distinct
Medicament admini	Med Order	Dose	MUST EXIST	just one

<u>Simple sentence</u> describing the <u>domain object</u> 'Flowchart' as instance of the PQI object 'Flowchart for current AS-IS process' whose attributes are columns in the next table.

PreCondition	Operation	ParentOperation	Operation Insp	Responsible P	Result_Type
	Med Order	Med Administer	No		Value added
	Order Transcript	Med Order	Yes		Cost added
	Pharmacy Check	Order Transcript	Yes		Value added
Medicament available	Med Pick	Pharmacy Check	No		Value added
Medicament missing	Med Reorder	Pharmacy Check	Yes		Cost added
	Administer to patient	ToNurse Send	No		Value added
	Tray Fill	Med Pick	No		Value added
	Cart Load	Tray Fill	No		Cost added
	ToNurse Send	Cart Load	No		Value added
	Examine	Med Administer	Yes		Value added
	Specialized Examine	Med Administer	No		Value added
	Patient Supervise	Administer to patient	No		Value added
	Blood Examine	Patient Supervise	No		Value added
	Diet Prescript	Patient Supervise	No		Value added

- **Med Administer**
  - **Med Order**
    - **Order Transcript**
      - **Pharmacy Check**
        - **Med Pick (Precondition: Medicament available)**
          - **Tray Fill**
            - **Cart Load**
              - **ToNurse Send**
                - **Administer to patient**
                  - **Patient Supervise**
                    - **Blood Examine**
                    - **Diet Prescript**
        - **Med Reorder (Precondition: Medicament missing)**

<u>Compound sentence</u> representing ideas in a cause-effect diagram with objects and operations (with capital letters) in the <u>domain ontology</u>. It is an instance of the PQI object 'Cause-Effect diagram'

- **Current Domain: Medicament administration in hospital**
- **Current Object: Patient**
- **Target characteristic: Medicine_cost**
  - **Negative/ positive effect: High medication cost**
    - **Main Cause(1) (1) many Physician Med Order many expensive Medicament**
    - **Main Cause(1) (2) some Medicament is incompatible for some Patient**
    - **SubCause(1) (3) AND Physician must Med Reorder**
    - **Second Cause(1) (4) some Medicament is not found in Pharmacy**
    - **SubCause(1) (5) AND Physician must Med Reorder expensive Medicament**
    - **Main Cause(2) (1) some Treatment is ineffective to many Patient**
    - **Second Cause(2) (2) AND many Medicament must Administer to patient without payment**
  - **Negative/ positive effect: Too many days in hospital**
    - **Main Cause(3) (1) some Treatment is ineffective to some Patient**
    - **Second Cause(3) (2) OR some poor, old Patient cannot pay Medicament**
    - **Main Cause(5) (1) many very sick Patient**
    - **SubCause(5) (2) AND Physician must Patient Supervise every day**
    - **Second Cause(5) (3) OR ineffective Treatment to some Patient**
    - **Main Cause(6) (1) many Physician wish not Patient Supervise at home**
    - **Second Cause(6) (2) AND all poor Patient cannot pay Nurse**
- **Target characteristic: No_Complaints**
  - **Negative/ positive effect: Too many complaints**
    - **Main Cause(4) (1) some inefficient Physician Med Order inappropriate Medicament**
    - **Main Cause(4) (2) OR some lazy Personnel**
    - **Main Cause(4) (3) OR few money for food**
    - **Main Cause(4) (4) OR some thief Personnel**

<u>Benefits from ideas expression using ontology-based sentences:</u>

•users have a <u>common vocabulary</u> and are forced to focus on the <u>most relevant aspects and concepts</u> in the domain and process.

•ideas can be automatically compared and grouped (at least by matching concepts with same syntactic role <u>(subject, predicate, complement)</u>).

Dynamic creation of a data collection sheet

Selection of characteristics describing the object 'patient' in the domain ontology (quality characteristic is 'medicine cost')

Current Project:	Current Domain:	Current Process:	Current Activity:	Current Object:	Current Inspection Sheet:
Health care	Medicament administration	Med Administer		Patient	

Selected Attribute	Domain	Object Type	Data Type	Attribute Name	Attribute Description	Improvement Criterium	Target Va
☒	Medicament administration in hospital	Patient	Date	Month	Period the patients are		
☐	Medicament administration in hospital	Patient	Decimal	Temperature			
☐	Medicament administration in hospital	Patient	Integer	Days_in_hospital		Minimize	
☑	Medicament administration in hospital	Patient	Decimal	Medicine_cost			
☐	Medicament administration in hospital	Patient	Integer	No_pills			
☑	Medicament administration in hospital	Patient	Text	Hospital_department			
☐	Medicament administration in hospital	Patient	Text	Name			
☐	Medicament administration in hospital	Patient	Text	Complaint_Type			
☐	Medicament administration in hospital	Patient	Integer	No_Complaints			

Schema of above table is defined in PQI ontology for PQI object "Domain Object"

Dynamic creation of the sheet

Month	Medicine_cost	Hospital_department
1/30/2002	2378	Dep1
1/30/2002	2094	Dep2
1/30/2002	1137	Dep3
1/30/2002	1389	Dep4
2/28/2002	2078	Dep1
2/28/2002	1954	Dep2
2/28/2002	1789	Dep3
2/28/2002	986	Dep4

Checking the Process Stability using Run Chart

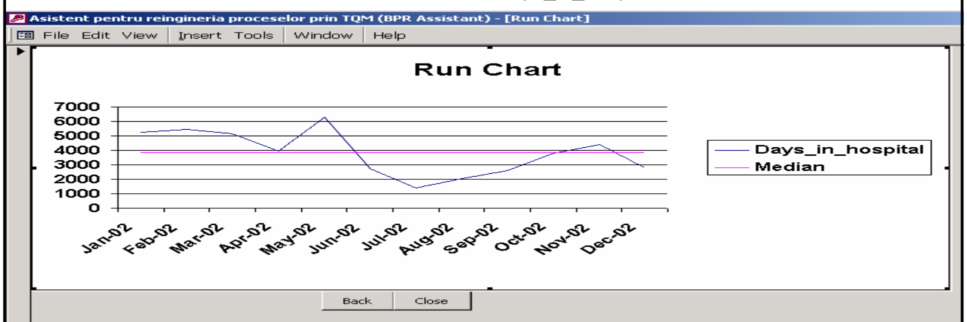
Creation of **run chart** or **control chart** using the data on the analysed characteristic, collected in a previously created and filled sheet. The charts represent the evolution in time of the analyzed characteristic.

<u>Runchart</u> reveals the existance of special causes for process instability when *trends*, *runs* (consecutive values on one side of the centerline (median)) or *cycles* (repeating patterns) appear.

<u>Example</u>

Object 'Patient'

Characteristics: 'Month' as time interval and 'Days_in_hospital' as controlled characteristic



Checking the Process Stability using Control Chart

With **control chart**, the causes of the process instability can be further controlled by

- *control limits* (three standard deviations from centerline); and
- rules for the interpretation of the values falling outside the control limits.

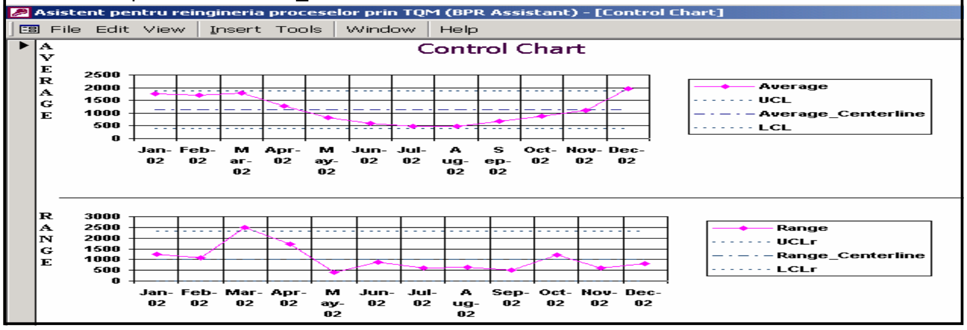
The software allows the representation of:

- *X-Bar* and *R* charts for variable data and sample size between 2 and 15;
- *Individual X* and *Moving Range* charts for attribute (count) data or variable data with sample size 1.

Example of XBar and R charts for variable data and sample size between 2 and 15

Object 'Patient'

Characteristics: 'Month' as time interval, 'Hospital_department' as generic name of the sample and 'Medicine_cost' as controlled characteristic



Checking the Process Improvement Ability using Histograms

Histograms (charts with vertical bars) for the analysis of a quality characteristic.

A histogram shows:

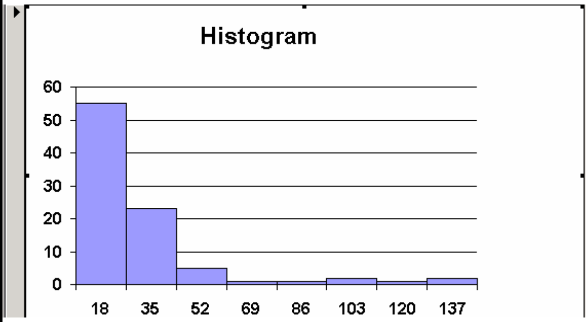
- where the values for a characteristic (e.g 'Days_in_hospital') fall in a measurement scale;
- how much is the variation.

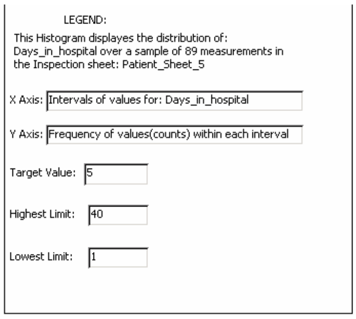
Example of histogram

Object 'Patient'

Characteristic 'Days_in_hospital'

The histogram allows the comparison of the values with the *specification limits* (e.g. minimum 1 day and maximum 40 days) and with the *target value* (e.g. 5 days) for the analysed quality characteristic





Identification of the Causes for Process Instability using Pareto chart

Pareto chart represents the *categories of problems* by bars, whose heights reflect the frequency or impact of the respective type of problems (number of times each category of problems occurs).

Pareto principle postulates that 80% of troubles comes from 20% of problems.

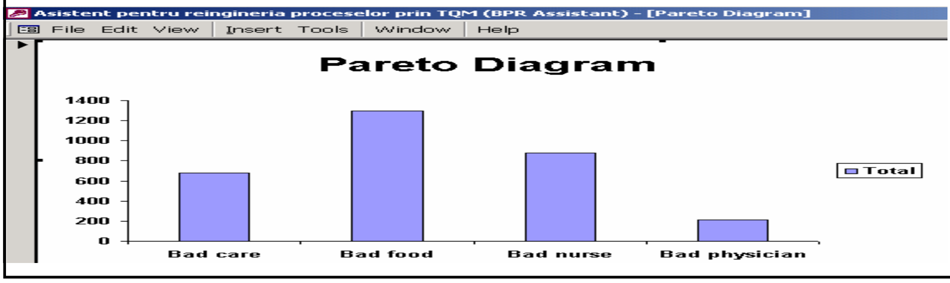
The tallest bars indicate the most important problems that must be analysed using cause-effect diagram.

Example

Object 'Patient' in the domain "Medicament administration in hospital".

Characteristics: 'Complaint_Type' as category name, 'Month' as generic name of the sample and 'No_Complaints' as controlled characteristic

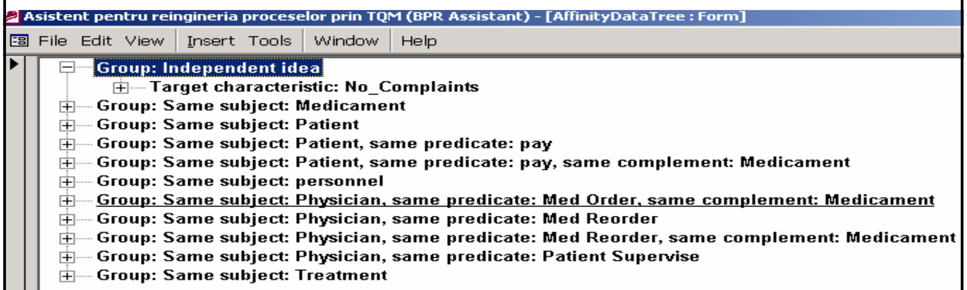
Computation type is 'Total' (among 'Total', 'Percentage', 'Sum_of_Percent', 'One sample')



Brainstorming Sessions

- users express ideas (in NL or ontological sentences) on any subject and PQI step.
- the ideas expressed by ontological sentences can be automatically grouped by matching concepts with same syntactic role (subject, predicate, complement), resulting into the ***affinity diagram***.

Example of affinity diagram for the ideas in the cause-effect diagram



- the members can express their vote on the final list of ideas and the mediator calls the *multivote* function, that automatically calculates the ***vote per idea*** (complex sentence) ***or sequence of idea*** (simple sentence).

Part of affinity diagram grouping ideas in the cause-effect diagram (according to a syntactic comparison algorithm)

- Group: Same subject: Medicament
  - Target characteristic: Medicine_cost
    - Negative/ positive effect: High medication cost
      - Main Cause(1) (2) some Medicament is incompatible for some Patient
      - Second Cause(1) (4) some Medicament is not found in Pharmacy
      - Second Cause(2) (2) AND many Medicament must Administration to patient without payment
- Group: Same subject: Patient
  - Target characteristic: Medicine_cost
    - Negative/ positive effect: Too many days in hospital
      - Second Cause(3) (2) OR some poor, old Patient cannot pay Medicament
      - Main Cause(5) (1) many very sick Patient
      - Second Cause(6) (2) AND all poor Patient cannot pay Nurse
- Group: Same subject: personnel
  - Target characteristic: No_Complaints
    - Negative/ positive effect: Too many complaints
      - Main Cause(4) (2) OR some lazy personnel
      - Main Cause(4) (4) OR some thief personnel
- Group: Same subject: Physician, same predicate: Med Order, same complement: Medicament
  - Target characteristic: Medicine_cost
    - Negative/ positive effect: High medication cost
      - Main Cause(1) (1) many Physician Med Order many expensive Medicament
  - Target characteristic: No_Complaints
    - Negative/ positive effect: Too many complaints
      - Main Cause(4) (1) some inefficient Physician Med Order inappropriate Medicament
- Group: Same subject: Physician, same predicate: Med Reorder
  - Target characteristic: Medicine_cost
    - Negative/ positive effect: High medication cost
      - SubCause(1) (3) AND Physician must Med Reorder
      - SubCause(1) (5) AND Physician must Med Reorder expensive Medicament
- Group: Same subject: Physician, same predicate: Patient Supervise
  - Target characteristic: Medicine_cost
    - Negative/ positive effect: Too many days in hospital
      - SubCause(5) (2) AND Physician must Patient Supervise every day
      - Main Cause(6) (1) many Physician wish not Patient Supervise at home
- Group: Same subject: Treatment
  - Target characteristic: Medicine_cost
    - Negative/ positive effect: High medication cost

Conclusions

- the developers of applications can <u>benefit from ontologies for</u> both:
  - description of the *business* (e.g. 'medicament administration'), but also for
  - description of the *applications* (e.g. a PQI system) and for their *interfaces*. Most part of the *code for the application interface can be reused* for other applications, by changing/ extending the application ontology.
- the <u>use of a relational DB for the representation of the process-oriented ontologies</u> (i.e. for the composition of controlled sequence of operations, for the description of objects and operations in the process) was possible because of the *natural representation in relational DBs of the conceptual graphs*, the inspiration source for the ontology-based sentences;
- <u>benefits for PQI system from the proposed process-oriented representation</u>:
  - *conceptual integration of ontologies* describing both PQI system and analyzed process, using the *same types of concepts* and a *uniform representation for both ontologies*.

    It *saves* the user's *time for system learning*, because the same representation was for both user *interface* and the infrastructure for the analyzed *process* (defined by the user)
  - *integration of operation/ process and object descriptions and of the semantic relationships* between processes/ operations/ objects.
    - most ontology languages do not have explicit representation capabilities for processes;
    - symbolic models do not allow the explicit integration of processes and objects (they propose separate diagrams for objects and processe, integrated in the programming code);

Conclusions (cont)

- benefits for PQI system from the proposed process-oriented representation (cont):
  - natural *extensibility* of the application and business ontologies, using the same conceptual background
    - application ontology can be extended or new application ontologies can be integrated with the existing ones (e.g. first the ontology for TQM and, then, for the integration of the Taguchi method);
    - an initial domain ontology can be extended with other domains, analyzed using the same PQI software and repository (e.g. TQM tested for 'medicament administration' and Taguchi method for the 'manufacturing' domain).
  - *code reusability*.
    - *interface* for both methodologies (TQM and Taguchi) is dynamically built using the *same reasoning algorithm, but different concepts in the PQI ontology*.
    - same algorithm for the automatic guidance and verification of the user's actions throughout both methodologies.
    - the two methodologies share:
      - ontology manager and the infrastructure (predefined in PQI ontology) for the domain ontology.
      - the utility steps (for scenario customization, project scheduling and brainstorming).

    Only the specific functions of the Taguchi operations (e.g. specific computations) have been be added to the existing code for TQM.
  - *domain-independent processing of the ontology-based sentences*
    - activities and processes described, integrated and stored outside the code can be reused and customized for other applications and can be processed using a general algorithm.
  - *technology-independent representation* (see example in XML)

Part of the representation in XML of the ‘PQI process’

```
<OPERATION Type="process" Name="PQI process“>
   ……
   <OPERATION Type= "step" Name=" Step 4 – Create AS-IS Process Diagram"
                          Quantif="must exist" Precondition ="“>
      ……
      <OPERATION Type="atomic" Name="Add/ Modify/ Delete AS-IS flowchart"
                          Quantif="may exist" Precondition =""“>
          <OBJECT Name="Current Domain" Quantif = "must  exist"  Role="LOC"  Value="“>
               <ATTRIBUTE Type="Description" Quantif="may exist" Role="CHRC" Value=""/>
               <ATTRIBUTE Type="PQI Project" Quantif="must exist" Role="LOC" Value=""/>
          </OBJECT>
             …..
          <OBJECT Name="Flowchart for  current AS-IS process"
                          Quantif = "must  exist"   Role="PTNT" Value="“>
              <ATTRIBUTE Type="Domain Process" Quantif="must exist" Role="PTNT" Value=""/>
              <ATTRIBUTE Type="Operation in process" Quantif="must exist"
                          Role="PART" Value=""/>
                 …….
          </OBJECT>
             …..
          <ATTRIBUTE Type="Operation State" Quantif="must exist" Role="STAT" Value=""/>
      </OPERATION >
      ……
   </OPERATION >
   ……
</OPERATION >
```