2010 Seventh International Conference on the Quality of Information and Communications Technology

Definition and Validation of Metrics for ITSM Process Models

Fernando Brito e Abreu
QUASAR/CITI/FCT
Universidade Nova de Lisboa
Caparica, Portugal
fba@di.fct.unl.pt

Raquel de Bragança V. da Porciúncula
QUASAR/CITI/FCT
Universidade Nova de Lisboa
Caparica, Portugal
raquel.braganca.porciuncula@gmail.com

Jorge Manuel Freitas
QUASAR/CITI/FCT
Universidade Nova de Lisboa
Caparica, Portugal
jorge.manuel.freitas@gmail.com

José Carlos Costa
QUASAR/CITI/FCT
Universidade Nova de Lisboa
Caparica, Portugal
jcgcosta@gmail.com

Abstract – **Process metrics can be used to establish baselines, to predict the effort required to go from an "as-is" to a "to-be" scenario or to pinpoint problematic ITSM process models. Several metrics proposed in the literature for business process models can be used for ITSM process models as well.**

This paper formalizes some of those metrics and proposes some new ones, using the Metamodel-Driven Measurement (M2DM) approach that provides precision, objectiveness and automatic collection. According to that approach, metrics were specified with the Object Constraint Language (OCL), upon a lightweight BPMN metamodel that is briefly described. That metamodel was instantiated with a case study consisting of two ITSM processes with two scenarios ("as-is" and "to-be") each. Values collected automatically by executing the OCL metrics definitions, upon the instantiated metamodel, are presented.

Using a larger sample with several thousand meta-instances, we analyzed the collinearity of the formalized metrics and were able to identify a smaller set, which will be used to perform further research work on the complexity of ITSM processes.

Keywords – ***IT Service Management; Process Modeling; BPMN; Metamodel; Process Metrics***

I. INTRODUCTION

A continuous improvement of IT Service Management (ITSM) processes is required to keep competitiveness. To improve existing processes, Business Process Reengineering (BPR) actions have long been carried out [1]. In BPR, organizations look at their processes from a "clean slate" perspective and determine how they can best construct these processes to improve the way they can be conducted. Reengineering is a fundamental rethinking and radical redesign of those processes to achieve effective improvements in cost, quality, speed, and service. A technique adopted for process reengineering is gap analysis, which aims at determining the steps to take in moving from the current state (aka "as-is") to a desired future state (aka "to-be"). Before the reengineering team can proceed to redesign an ITSM process, it should understand the existing one (the "as-is" or baseline), namely to identify what prevents the process from achieving the desired results. Modeling the current state can be performed with the Business Process Model and Notation (BPMN) [2] which has a well defined syntax and semantics, due to a precise metamodel, therefore allowing unambiguous modeling. The increasing importance and adoption[1] of BPMN stems from the fact of being promoted by the Object Management Group (OMG), a major stakeholder in the IT field, that has pushed other important initiatives like the UML standardization.

The objective of the "to-be" phase is to produce alternatives to the current situation which satisfy the strategic goals of ITSM [3]. Besides performing self-assessments on the status of IT performance, it is equally important to test and compare it with the view the market has on which are the best practices that characterize best-of-breed organizations. Fortunately, several best practices frameworks for ITSM are available to guide us in setting "to-be" scenarios, like the IT Infrastructure Library (ITIL) [4], the Control Objectives for Information and related Technology (CobiT) [5], the Microsoft Operations Framework (MOF) [6] or the IBM Tivoli Unified Process (ITUP). Those ITSM frameworks have a lot in common. Several mapping initiatives have been conducted, such as between COBIT and ITIL [7] or MOF and ITIL [8].

The combined results of best-practices benchmarking and self-assessments lead to the identification of gaps in terms of people, process and technology [9]. Having identified the potential improvements to the existing ITSM processes, the "to-be" process models can then be designed. Summing up, the deliverables of either the "as is" or the "to be" phases are a set of process models expressed in the chosen process modeling language. Herein we will consider that language to be BPMN. For a comparison with other process modeling languages see [10].

To estimate the costs of process reengineering we need to quantify the gap between "as-is" and "to-be" scenarios [11]. The larger the gap, the greater effort we will require to bridge it. One way of quantifying that gap is measuring the difference in complexity between the "as-is" and "to-be" process models. This requires quantifying the complexity of each process model.

Instead of the somehow radical process reengineering approach, we can adopt a more conservative continual process improvement approach to the effectiveness and efficiency of service delivery and management [9], by following a Plan-Do-

[1] - An indicator of this claim is the observation that a growing number of IT modeling tools are offering BPMN.

978-0-7695-4241-6/10 $26.00 © 2010 IEEE
DOI 10.1109/QUATIC.2010.13

Check-Act cycle. Checking involves monitoring, measuring and analyzing, so process complexity metrics are also a must. Last, but not the least, process complexity has also been found to be a good predictor of the propensity to fail [12, 13], so measuring process complexity is required if we intend to develop process reliability estimation models.

Although several metrics for assessing the complexity of process models have been recently proposed, we were not able to find in the literature a systematic and replicable approach to specify and collect them. In this paper we formalize several process complexity metrics using the Metamodel Driven Measurement (M2DM) technique [14], using OCL [15] upon a BPMN metamodel. The advantages of this technique are (i) the clarification of the target domain (due to the use of a metamodel), (ii) the accuracy achieved by the use of OCL, (iii) the availability of tools for automatic metrics collection that use as input an OCL specification, as the USE (UML-based Specification Environment) used herein [16], and finally (iv) it facilitates the replication of experiments due to its objectivity.

This paper is organized as follows: first, we briefly describe a BPMN metamodel; next, an overview is given of a case study; then, we outline the metamodel instantiation with the USE tool; then we show how a set of metrics can be formally defined upon that metamodel using the M2DM approach; we then describe the metrics collection process and present the resulting values for the case study; finally, we reduce the number of metrics to obtain a non-collinear set and envision some research questions where those process metrics will play an important role, as an outline of our future research in the area of ITSM process complexity.

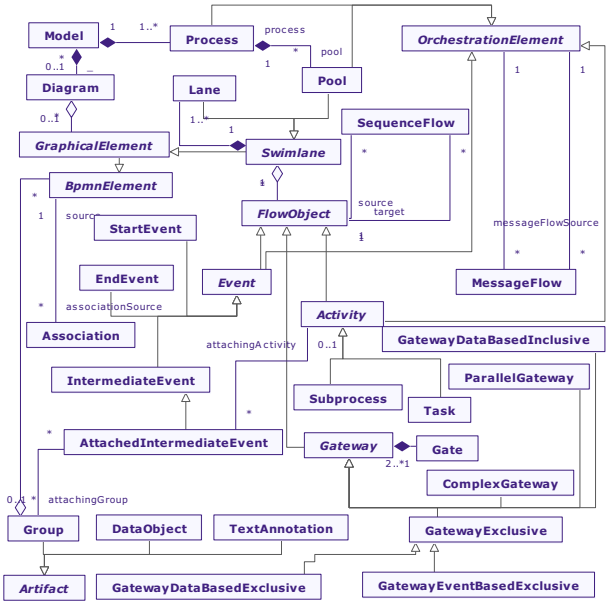


Figure 1. BPMN metamodel (abridged version)

II. BPMN METAMODEL

OMG specifications are usually represented by UML class diagrams where metamodel well-formedness rules are expressed as OCL clauses. A public domain metamodel expressing OMG's BPMN specification [2] was not available by the time we started this research thread on the complexity of ITSM process models. Therefore, we decided to develop one in our research group [17]. Although not compliant with OMG's MOF [18], our metamodel is based in the BPMN v1 specification and has a full coverage of its modeling concepts, as outlined in Figure 1. This metamodel was modularized into several packages, by grouping related concepts, to facilitate understanding:

- **Model** - Is the top view where we represent the project we are modeling and its versions;
- **StructuralElements** - Represents how BPMN elements that constitute a process interact;
- **SupportingElements** – Represents concepts that are required for process implementation, but do not have graphical representation in BPMN, such as gateway ports.
- **FlowObjects** – Represents Activities, Gateways and Events; these are the more recurrent BPMN elements;
- **Connectors** - Represents elements that interconnect the other BPMN elements such as Associations, Message Flows and Sequence Flows;
- **InternalConnections** - Represents how the different BPMN elements are connected to each other within the same pool (using Sequence Flows);
- **Orchestration** - Represents how BPMN elements on different pools communicate (using Message Flows);
- **Artifacts** – Represents additional elements such as Groups (of elements), Data Objects and Text Annotations;
- **PackageUseCases** – Represents the traceability of process modeling to use case modeling;
- **Simulation** - Represents the resources definition and their allocation to tasks, to support model enactment.

The package diagram in annex A briefly outlines the organization of our BPMN metamodel. As with OMG's MOF-based metamodels, this one was enriched with well-formedness rules (e.g. referential integrity constraints) expressed as OCL clauses.

To allow replication by other researchers, the full metamodel and its instantiation (both expressed in the USE tool input format) are made available at our web site in: http://ctp.di.fct.unl.pt/QUASAR/Resources/DataFiles/

III. CASE STUDY AND METAMODEL INSTANTIATION

We now introduce a case study that will be used to instantiate the BPMN metamodel and to illustrate the metrics collection process.

A. Case Study

This case study concerns a public sector organization responsible for supporting the governance of Portuguese Government electronic domains, with a strategic focus on electronic security, communication and decision-support systems. It has a CEO, 2 department coordinators, 34 internal IT professionals and 12 outside collaborators and is organized in 3 units:

1. **Quality and Services Department:** its mission is to assure quality management and services of this organization, including the whole range from services of local support and personalized services to users, management of technology support services and the provision of services. It is also in charge of auditing. This organizational unit integrates all the organizational skills of Users and Services, Control and Operations, Service Desk, Project Management, Service Management and Media Lab.

2. **Technology and Applications Department:** its mission is to manage the entire technology infrastructure, including the whole range of services from the infrastructure and communications services and networking, operating systems, applications and internet content. This organizational unit has all the organizational skills of Communications and Systems, Security and Digital Identity, Applications and Systems and Decision Support, Internet Content and different dimensions of support and assistance, operational control of infrastructures and engineering research and development.
3. **Board Secretariat**: its mission is supporting the Director on Administrative and Financial matters, Marketing and Communication activities.

This organization aims to be certified on the adoption of good practices in IT Service Management and therefore chose the ISO/IEC 20000:2005 standard [19, 20], which is based on ITIL version 2 [21]. Besides being a good basis for independent (third-party) assessment, this standard may be used by businesses that are going out to tender for their services, to provide a consistent approach by all service providers in a supply chain, to benchmark IT service management, to demonstrate the ability to meet customer requirements and to improve services.

To be certified according to ISO/IEC 20000, thirteen 13 distinct processes must be implemented. To do so, this organization implemented new processes and improved existing ones. One of the co-authors of this paper was actively involved in this process reengineering action. In annexes B through E we present the "as-is" and "to-be" models for two processes that were already in place: *Incident Management* and *Problem Management* and were improved according to good practices.

In ISO/IEC 20000 and ITIL terminology, an *incident* is defined as an unplanned interruption to an IT service or reduction in the quality of an IT service. Failure of a configuration item that has not yet impacted service is also an incident, for example failure of one disk from a mirror set. The primary goal of the *Incident Management* process is to restore normal service operation as quickly as possible and minimize the adverse impact on business operations, thus ensuring that the best possible levels of service quality and availability are maintained.

ISO/IEC 20000 and ITIL define a *problem* as the unknown cause of one or more incidents. Problem Management is the process responsible for managing the lifecycle of all problems. The primary objectives of the *Problem Management* process are to prevent problems and resulting incidents from happening, to eliminate recurring incidents and to minimize the impact of incidents that cannot be prevented.

The main difference between *Incident Management* and *Problem Management* is that the former deals with fighting symptoms to incidents, while the latter seeks to remove the causes of incidents permanently from the IT infrastructure. In *Incident Management*, interaction with customers is usually reactive, with the main objective being to find a workaround solution to restore normal services for the customer, as soon as possible. In *Problem Management*, IT support staff is more proactive as they dedicate resources to establishing the underlying causes of incidents. In this process there is usually little or no interaction with the customers, as this is left to the responsibility of the *Service Desk*.

B. Metamodel Instantiation

Using a model-driven transformation, our BPMN metamodel, originally produced with a UML visual modeling tool, was transformed into its textual representation for input in the USE validation environment [17]. Also using a model-driven transformation, as described in [22], we generated BPMN metamodel instances out of the four ITSM processes in annexes B through E, which were modeled as BPMN diagrams, also using a visual modeling tool. This transformation required several steps, which are summarized:

Step 1 – Identification of all process objects (represented as meta-objects in the BPMN metamodel) and naming them with a convention that guaranteed uniqueness. This was required to avoid name clashing, because the USE tool only supports a single namespace.

Step 2 – Identification of the source and target objects for each sequence flow and message flow. Flows were named by composing the source and target identifiers.

Step 3 – Generation of meta-class and meta-association instances (meta-objects and meta-links, respectively) in the USE input format.

Step 4 – Loading the BPMN metamodel and then the ITSM instances upon it.

Step 5 – Checking of all situations described by the well-formedness rules, and correct those were they were being violated. In several cases we were also able to detect metamodel inconsistencies that were promptly eliminated.

IV. Metrics Formalization and Collection

A. Metrics definition

A significant amount of research has been conducted in the past decades on the complexity of software programs [23], and software complexity metrics have been used for many purposes such as predicting error rates, detecting design flaws, assessing modularity, supporting refactoring decisions, estimating maintenance costs or identifying pieces of software that should be re-engineered or migrated to another paradigm [24-29]. Several authors have proposed to adapt software complexity metrics for analyzing the complexity of process models. Some of these authors also proposed to adapt metrics originating from research on network complexity, due to the similarities among software designs, network designs and process designs (all are domain-specific directed graphs).

We now present the formalization of some of those metrics, along with others proposed in [30], using the M2DM approach, as mentioned previously. To fully comprehend metrics expressions, an adequate understanding of OCL syntax and BPMN metamodel semantics is required. The former can be obtained in several textbooks such as [31]. As for the latter, the full metamodel can be found in [17], available at the QUASAR group page. Figure 1 is only an excerpt of that metamodel, therefore not sufficiently detailed to fully understand the metamodel transversals operated by several OCL expressions presented hereafter.

Size Metrics

The IEEE Standard Computer Dictionary defines complexity as *"the degree to which a system or component has a design or implementation that is difficult to understand and verify"* [32]. Size (aka length) is one facet of complexity and has long been measured in software by using absolute scale type metrics such as lines of code (LOC) or number of classes. For process models, the **number of activities** in the model can be regarded as an equivalent to the number of executable statements in a piece of software and can be used as simple, easy to understand, metric model size [33]. However, this metric does not take into account the structure of the model: a model with 50 activities may be designed using a well-structured control flow which is easy to follow or in an unstructured way which makes understanding very hard [33]. We also present other size metrics defined in the absolute scale, which will be required for the definition of subsequent metrics. Metric names, which are self-explanatory, are prefixed by the context (the metaclass **Process** in most cases). The "self" identifier refers to an instance (meta-object) of the context metaclass:

```
Process::countBpmnElements(): Integer = self.pool ->
        iterate(elem: Pool; acc: Set(BpmnElement) =
                oclEmpty(Set(BpmnElement)) | acc ->
                        union(elem.bpmnElements())) -> size()

Process::totalNumberProcessActivities(): Integer =
        self.bpmnElements()->select(oclIsKindOf(Activity))->size()

Process::totalNumberProcessStartEvents(): Integer =
        self.bpmnElements()->select(oclIsKindOf(StartEvent))->size()

Process::totalNumberProcessEndEvents(): Integer =
        self.bpmnElements()->select(oclIsKindOf(EndEvent))->size()

Process::totalNumberProcessGateways(): Integer =
        self.bpmnElements()->select(oclIsKindOf(Gateway))->size()

Process::totalNumberProcessTextAnnotations(): Integer =
        self.bpmnElements()->select(oclIsKindOf(TextAnnotation))->size()

Process::totalNumberProcessSequenceFlows(): Integer =
        self.bpmnElements()->select(oclIsKindOf(SequenceFlow))->size()

Process::totalNumberProcessFlowObjects(): Integer =
        self.bpmnElements()->select(oclIsKindOf(FlowObject))->size()
```

CNC - Coefficient of Network Complexity

The Coefficient of Network Complexity (CNC) is a widely used metric in network analysis and was proposed to measure the degree of complexity of a critical pass network [34]. It can be calculated as the number of arcs divided by the number of nodes. In the context of a business process model, Cardoso defined it as the number of arcs divided by the number of activities, joins and splits:

CNC = number of arcs / (number of activities, joins and splits)

Using M2DM we can express it as:

```
Process::CNC() : Real =
totalNumberProcessSequenceFlows() / totalNumberProcessFlowObjects()
```

Henry and Kafura Metric (HKM)

The benefits of divide-and-conquer approaches are well known and sought in all fields to mitigate complexity. Dividing a process model in modular sub-models cannot only help to make it easier to understand, it can also lead to smaller, reusable models [33].

For analyzing the modularization of a process model, we can adapt the ideas of Henry and Kafura [35]. They discussed the structure of modularized software systems and proposed a metric based on the information flow in a program's structure. They picked two terms used for electronic gates and dubbed them to software modules: *fan-in* for a module is the number of other modules which use it and *fan-out* for a module is the number of other modules it uses. Henry and Kafura then proposed the following metric for modules complexity, where the *length* metric can be LOC or McCabe's CFC:

$$\text{Module complexity} = length * (fan\text{-}in * fan\text{-}out)^2$$

Henry and Kafura validated their metric using historical data on bugs found during the UNIX system development. They found that components with higher complexity where more problem-prone and therefore are candidates for redesign.

Ghani [36] suggested using this metric in the same way for analyzing process models, but does not provide the corresponding concept mapping. If we consider that a process model may have several cooperating processes, each on its own pool, we can consider processes to be the modules and the input and output events to be their *fan-in* and *fan-out*, respectively. The mapped Henry and Kafura metric is then:

$$HKM = \text{total of activities} \times (\text{n. of } StartEvents \times \text{n. of } EndEvents)^2$$

Using M2DM we can express this as follows:

```
Process::HKM(): Real =
    totalNumberProcessActivities() *
    power((totalNumberProcessStartEvents() * totalNumberProcessEndEvents()), 2)
```

CFC - Control Flow Complexity

The cyclomatic number, introduced by Tom McCabe [37], is one of the most widely used software metrics. It is calculated from the control flow graph and measures the number of linearly-independent paths (possible control flows) through a program [33]. A lower cyclomatic number is claimed to indicate that the program is easier to understand and maintain. The cyclomatic number is also an indicator of testability, because it corresponds to the number of test cases needed to achieve full path coverage. It was found that there is a significant correlation between the cyclomatic number of a piece of software and its defect level [38].

Cardoso [39] suggested a process complexity measure, as a generalization of McCabe's *cyclomatic number*. His Control-Flow Complexity (CFC) metric is based on the analysis of control-flow elements (XOR, OR and AND-splits). The main idea behind this metric is to evaluate the number of mental states that have to be considered when a designer is developing a process. Mathematically, the control-flow complexity metric is additive, thus it is possible to calculate the complexity of a process by simply adding the CFC of all split constructs. The CFC metric is then calculated as follows:

$$CFC\ (P) = \sum_{a \in P, a\ isa\ xor-split} CFC_{XOR}(a) + \sum_{a \in P, a\ isa\ or-split} CFC_{OR}(a) + \sum_{a \in P, a\ isa\ and-split} CFC_{AND}(a)$$

***Note**: "P" is a process and "a" an activity.*

According to [33], every split in the process adds to the number of possible decisions as follows:

– AND-split: All transitions outgoing from an AND-split must be processed; the designer needs only to consider one state as the result of the execution of an AND-split; therefore, every AND-split in a process adds *1* to its CFC metric;

– XOR-split with *n* outgoing transitions: Exactly one out of *n* possible paths must be taken, i.e. we have to consider *n* possible states that may arise from the execution of the XOR-split; hence, every XOR-split with *n* outgoing transitions adds *n* to the CFC metric of this process;

– OR-split with n outgoing transitions: There are $2n-1$ possibilities to process at least one and at most *n* of the outgoing transitions of an OR-split, i.e. every OR-split with *n* outgoing transitions adds $2n - 1$ to the CFC metric.

A preliminary proof of CFC validity is provided in [40], where a significant correlation between the *perceived complexity* (as rated by students) and the CFC metric is reported. Cardoso concluded that the CFC metric is highly correlated with the control-flow complexity of processes. This metric can, therefore, be used by business process analysts and process designers to analyze the complexity of processes and, if possible, develop simpler processes. A shortcoming of this metric is that the number of possible decisions in a process does not tell much about its structure [33].

In our BPMN metamodel, **XOR-splits** are the *Data-Based Exclusive Gateways* and *Event-Based Exclusive Gateways*, **OR-splits** are the *Data-Based Inclusive Gateways* and **AND-splits** are *Parallel Gateways*. Using M2DM we can therefore express the CFC metric as follows:

```
Process::CFC(): Integer = CFC_XOR_DataBased() +
          CFC_XOR_EventBased() + CFC_OR() + CFC_AND()
```

where:

```
Process::CFC_XOR_DataBased(): Integer = self.bpmnElements()->
  select(oclIsKindOf(GatewayDataBasedExclusive)).
    oclAsType(GatewayDataBasedExclusive)->asSet->
      select(isSplit())->collect(numberOutputGates())->sum()

Process::CFC_XOR_EventBased(): Integer = self.bpmnElements()->
  select(oclIsKindOf(GatewayEventBasedExclusive)).
   oclAsType(GatewayEventBasedExclusive)->asSet->
    select(isSplit())->collect(numberOutputGates())->sum()

Process::CFC_OR(): Integer = self.bpmnElements()->
  select(oclIsKindOf(GatewayDataBasedInclusive)).
   oclAsType(GatewayDataBasedInclusive)->asSet->
     select(isSplit())->collect(numberOutputGates())->sum()

Process::CFC_AND(): Integer = self.bpmnElements()->
  select(oclIsKindOf(ParallelGateway)).
    oclAsType(ParallelGateway)->asSet->
      select(isSplit())->collect(numberOutputGates())->sum()

Gateway::isSplit() : Boolean = numberOutputGates() > 1

Gateway::numberOutputGates() : Integer =
          self.gate-> select(type=#GateType_Output)->size()
```

Path metrics

Since process models are stereotyped directed graphs, we can use graph transversal algorithms (e.g. breadth-first or depth first transversal), to determine aspects such as the number of paths or the length of the shortest or longest paths. Hereafter we propose some path metrics using the M2DM approach:

```
// This function computes all possible paths given a source and a destination.
// It basically initializes the recursive node visitor with adequate arguments
Process::compute(origin: FlowObject, destination: FlowObject):
                                              Set(Sequence(FlowObject)) =
        visit( origin, destination,
               oclEmpty(Set(FlowObject))->including(origin),
               oclEmpty(Sequence(FlowObject))->append(origin),
                oclEmpty(Set(Sequence(FlowObject))))

// Depth-first recursive node visitor
Process::visit ( x: FlowObject,                    //current node
                 destination: FlowObject,          //destination node
                 visited: Set(FlowObject),         //set of visited nodes
                 path: Sequence(FlowObject),       //current path
                 paths:Set(Sequence(FlowObject))):  //all paths visited
                                              Set(Sequence(FlowObject)) =
        if x=destination then                  // destination was found
            paths->including(path)
        else
            if x.successors()->isEmpty() then //recursion stops in this path
                    paths
            else                    // iterates recursively on node successors
                    x.successors()->iterate(elem: FlowObject;
                        acc: Set(Sequence(FlowObject)) = paths |
                        if visited->excludes(elem) then
                            visit(elem, destination, visited ->
                              including(elem), path-> append(elem), acc)
                        else
                            acc    // accumulator variable is returned
                        endif)
            endif
        endif

// Returns the successors of the current flowObject
FlowObject::successors(): Set(FlowObject) = self.flowSource.target->asSet()
// Returns the number of possible paths, given an origin and a destination
Process::numberPaths(origin: FlowObject,
                     destination: FlowObject): Integer =
    compute(origin, destination) -> size()

// Returns all flowObjects in all possible paths given an origin and destination
Process::allElements(origin: FlowObject,
                     destination: FlowObject): Set(FlowObject) =
    compute(origin, destination) -> flatten

// Counts all flowObjects in all possible paths given an origin and destination
Process::countAllElements(origin: FlowObject,
                          destination: FlowObject): Integer =
     allElements(origin, destination) -> size()

//Returns the longest path; when in a tie, returns the first of the longest ones
Process::biggestPath(origin: FlowObject,
                     destination: FlowObject): Sequence(FlowObject) =
    compute (origin, destination) ->
        iterate(elem: Sequence(FlowObject);
            acc: Sequence(FlowObject)=oclEmpty(Sequence(FlowObject)) |
                if (elem->size() > acc->size()) then
                    elem
                else
                    acc
                endif)

// Returns the length (number of flowObjects) of the longest path
Process::sizeBiggestPath(origin: FlowObject, destination: FlowObject):
                          Integer = biggestPath(origin, destination) -> size()
```

Nesting depth

Like in source code, unstructured models are less understandable than the equivalent structured ones [41]. How-

ever, too much (deep) nesting in the structure is a bad practice, since it reduces readability [42]. Gruhn and Laue [33] observed that the term "nesting depth" may be misleading. Current graph-oriented process modeling languages do not require proper nesting i.e. splits and joins does not have to occur pair-wise. This is comparable with programming languages that besides structured iteration constructs (e.g., for, forearch, while, repeat...until) also support unconditional jumps (GOTOs). Holl and Valentin [43], picking the classic repudiation of spaghetti code [44], observed that the current unstructured style of process modeling, results in "spaghetti process models". Aalst also discussed this model unstructuredness issue related to the nesting of split/join constructs [45]. He uses Petri Nets for representing process models and defined a process to be well-structured if the corresponding workflow net does not contain handles. We have a handle, if for any pair of nodes x and y such that one of the nodes is a place and the other a transition, there exist two different paths from x to y which have more common elements than just x and y. This means that the number of handles is a measure for the number of unstructured constructs.

Gruhn and Laue [33] claim that a model with greater nesting depth implies greater complexity. The nesting depth of an element equals the number of decisions in the control flow that are required to reach this element. Ghani et al. [36] claim that models with nested XOR-splits and XOR-joins are more complex and harder to understand than an almost linear model, but the CFC for both models may be same. For this reason, they suggested using the nesting depth metric to get the nesting depth value and add the value to the CFC in order to measure the process complexity:

Process complexity = Nesting depth + CFC

Expressing it with M2DM we get:

```
Process::procComplexity(origin: FlowObject, destination: FlowObject): Integer =
    nestingDepth(origin, destination) + CFCAux(origin, destination)

Process::nestingDepth(origin: FlowObject, destination: FlowObject): Integer =
    compute(origin, destination) -> flatten->
        select(oclIsKindOf(Gateway))->
            collect(oclAsType(Gateway))->size

Process::CFCAux(origin: FlowObject, destination: FlowObject): Integer =
    compute(origin, destination) ->flatten->
        select(oclIsKindOf(Gate))->
            collect(oclAsType(Gate))->
                select(type=#Output) -> size
```

B. Metrics collection

For each process defined in our case study we collected the corresponding metrics, as represented in Table I. Due to space constraints, not all metrics definitions are included in this paper, but can be found in [30].

As can be observed in Table I, all "to-be" metrics have values greater or equal to those of the corresponding "as-is" counterparts. For instance, the number of possible paths in the "to-be" phase of the *Incident Management* (IM) and the *Problem Management* (PM) process is much higher than their "as-is" correspondents. This is due to the increased number of gateways and hence to the increase of the number of activities and connectors. This increase is also evident by the total number of objects, where we can note that the IM process has 22 elements in the "as-is" phase and 37 in the "to-be" phase, and the PM process has 12 elements in the "as-is" phase and 25 elements in the "to-be" phase. This comes as no surprise, because more immature ITSM processes are usually trivial and first efforts to adopt best practices increase their complexity. We will recall this issue in the conclusions section.

TABLE I - METRICS COLLECTION FOR THE CASE STUDY

	Incident Management		Problem Management	
	"As-Is"	*"To-Be"*	*"As-Is"*	*"To-Be"*
N. of Processes	1	1	1	1
N. of possible paths	3	10	1	8
N. of different objects in all possible paths	22	37	12	25
N. of Objects in biggest path	18	27	12	21
N. flowObjects in smallest path	12	13	12	19
N. of Pools	2	2	1	1
N. of Lanes	4	4	3	3
N. of Swimlanes	6	6	4	4
N. of FlowObjects	16	22	12	18
N. of Activities	12	15	10	14
N. of SubProcesses	0	0	0	0
N. of Tasks	12	15	10	14
N. of Events	2	2	2	2
N. of Start Events	1	1	1	1
N. of End Events	1	1	1	1
N. of Intermediate Events	0	0	0	0
N. of Gateways	2	5	0	2
N.GatewayDataBasedExclusive	2	5	0	2
N.GatewayEventBasedExclusive	0	0	0	0
N. GatewayDataBasedInclusive	0	0	0	0
N. ComplexGateways	0	0	0	0
N. ParallelGateways	0	0	0	0
N. of Gates	6	15	0	6
N. of InputGates	2	5	0	2
N. of OutputGates	4	10	0	4
N. of Artifacts	0	0	0	1
N. of DataObjects	0	0	0	0
N. of TextAnnotations	0	0	0	1
N. of Groups	0	0	0	0
N. of Connectors	17	26	12	20
N. of SequenceFlows	17	26	12	19
N. of MessageFlows	0	0	0	1
N. of Associations	0	0	0	0
CFC Metric	4	10	2	4
CNC Metric	1,1	1,2	0,9	1,1
NestingDepth Metric	3	6	1	4
Proc. Complexity Metric	8	17	2	10
HKM Metric	12	15	10	13

C. Cross-correlation analysis

Process metrics formalized in this study can be used as explanatory (predictor) variables in regression analysis. However, their large number may not be justifiable in the presence of multicollinearity. The latter is a statistical phenomenon in which two or more predictor variables in a multiple regression model are highly correlated. In this situation the coefficient estimates may change erratically in response to small changes in the model or the data. Multicollinearity does not reduce the predictive power or reliability of the model as a whole; it only affects calculations regarding individual predictors. That is, a multiple regression model with correlated predictors can indicate how well the entire bundle of predictors predicts the outcome variable, but it may not give valid results about any individual predictor, or about which predictors are redundant with others.

By means of a cross-correlation analysis, we now present a preliminary assessment on the redundancy among the process metrics that were introduced in this paper. Since most metrics were not normally distributed, we used a non-parametric

correlation coefficient – *Spearman's rank* – upon a much larger process model that was defined in the context of [22]. The latter includes 13 BPMN diagrams, meta-objects (instances of the BPMN metamodel).

This cross-correlation analysis aimed at identifying a set of metrics that are weakly correlated, therefore allowing to reduce the number of variables and thus the effort of harvesting (and data processing), if they are redundant. This technique is often used for data reduction. If two metrics have a very high correlation, this probably accounts for the fact that they are measuring the same facet of complexity and so we can get rid of one of them. Table II presents the values of the correlation coefficient for all weakly correlated metrics.

TABLE II – CROSS-CORRELATION ANALYSIS

ID	*Variables*	*B*	*C*	*D*	*E*
A	N. Possible Paths	,002	,054	,143	,539
B	N. TextAnnotations		,376	,054	,378
C	CNC Metric			-,022	,364
D	HKM Metric				,324
E	HPC difficulty Metric				

V. CONCLUSIONS AND FUTURE WORK

ITSM process models are directed graphs, such as those underlying computer network diagrams or sequence flow diagrams representing source code. Therefore, complexity metrics proposed on other knowledge areas can be adapted to measure the complexity of ITSM process models, as other researchers cited in this paper, in particular those concerned with business process modeling (BPM), have already proposed. However, we could not find in the literature a systematic and replicable approach to measure the complexity of process models. To mitigate this problem we propose herein the application of the *MetaModel Driven Measurement (M2DM)* approach. M2DM allows the clarification of the target domain, grants accurateness in metrics specification due to the use of OCL and offers automatic metrics collection, provided we are able to instantiate the BPMN metamodel appropriately. As a result, it facilitates metrics usage by practitioners and experiments replication by researchers.

After applying a data reduction technique, we have identified a set of five metrics of process complexity that are mostly non-collinear. They are candidates to act as descriptive variables in further research works. In concrete, we plan to set up experimental designs to answer the following research questions:

- How does ITSM process complexity affect process operation? In particular, we are interested in exploring how these process complexity metrics are related to some operational metrics such as "mean time to restore service", "calls to second-tier resolver teams" or "% of incidents/problems resolved within service targets". Here we will follow a similar validation approach as the one we used in [46]. We also plan to research if the proposed process complexity metrics can be used in the formulation of Critical Success Factors (CSFs) and Key Performance Indicators (KPIs) for ITSM processes.

- Is it possible to estimate the effort (cost) to perform an ITSM process reengineering action, based on the distance between "as-is" and "to-be" scenarios? If so, Continual Service Improvement (CSI) could use these process complexity metrics as input in identifying improvement opportunities for each process [9]. Although sometimes small changes can cost a lot to make, we expect that the distance calculated upon the metrics of the "as-is" and "to-be" models may give some pointers to the likely effort involved in such process improvement work. Its feasibility will require the "as-is" and "to-be" models to be expressed at the same level of abstraction / granularity.

- How is process maturity related with process complexity? While maturity is usually defined in a finite number of levels (typically 5), process complexity can be arbitrarily large. Although we have not collected sufficient supporting evidence, we believe that the corresponding transfer function (process maturity versus process complexity) will be somehow trapezoidal or parabolic (convexity pointing upwards). In the beginning, as organizations move from ad-hoc through defined levels of process maturity, an increase in process maturity is reflected by an increase in process complexity, as we observed in our case study, but that increase tends to stabilize. As organizations move into high maturity settings, their process complexity decreases as they learn and improve their processes, by applying continual innovation techniques.

We cannot conclude this paper without mentioning the most serious threat in addressing all previously described research questions: most IT service processes are highly dependent on people and culture. We agree when one reviewer mentioned that the implementation of ITIL is often a "hearts-and-minds" exercise. As such, the precision, objectiveness and automation that the proposed approach to collect process metrics allows, must be somehow combined with other less objective factors that can only be obtained with the help of the IT staff on the ground. In other words, any estimation model based upon process metrics will need on-site calibration.

ACKNOWLEDGMENT

The work presented herein was partly supported by the *VALSE project* of the CITI research center within the Department of Informatics at FCT/UNL in Portugal.

REFERENCES

[1] R. L. Manganelli and M. M. Klein, *The Reengineering Handbook: A Step-by-Step Guide to Business Transformation*, New York, American Management Association, New York, 1994.

[2] OMG, "Business Process Model and Notation (BPMN)", FTF Beta 1 for Version 2.0, Specification dtc/2009-08-14, Object Management Group (OMG), 2009.

[3] M. Iqbal and M. Nieves, *ITIL v3 Core Practice Book 1: Service Strategy*, London, UK, The Stationery Office (TSO), Office of Government Commerce (OGC), London, UK, 2007.

[4] *The Official Introduction to the ITIL Service Lifecycle Book*, London, UK, The Stationery Office (TSO), Office of Government Commerce (OGC), London, UK, ISBN:13: 9780113310616, 2007.

[5] "COBIT® 4.1 - Framework, Control Objectives, Management Guidelines, and Maturity Models", The IT Governance Institute - ITGI, 2007.

[6] "Microsoft Operations Framework (MOF): MOF Overview", 4th ed, Microsoft Corporation, 2008.

[7] "ITIL-cobit-mapping: Gemeinsamkeiten und Unterschiede der IT-standards", Information Systems Audit and Control Association (ISACA) / IT Service Management Forum (itSMF), May 2008.

[8] "Microsoft Operations Framework; Cross Reference ITIL V3 and MOF 4.0", Microsoft Corporation, May 2009.

[9] G. Case and G. Spalding, *ITIL v3 Core Practice Book 5: Continual Service Improvement*, London, UK, The Stationery Office (TSO), Office of Government Commerce (OGC), London, UK, 2007.

[10] B. List and B. Korherr, "An evaluation of conceptual business process

modelling languages", in proceedings of the *Proceedings of the 2006 ACM symposium on Applied computing*, Dijon, France, 2006.

[11] H. Sneed, "Estimating the costs of a reengineering project", in proceedings of the *12th Working Conference on Reverse Engineering (WCRE'2005)*, pp. 111-119, 2005.

[12] J. Mendling, "Testing Density as a Complexity Metric for EPCs", Technical Report JM-2006-11-15, Vienna University of Economics and Business Administration, Vienna, Austria, 2006.

[13] J. Mendling, M. Moser, G. Neumann, H. M. W. Verbeek, B. F. v. Dongen, and W. M. P. v. d. Aalst, "A Quantitative Analysis of Faulty EPCs in the SAP Reference Model", BPM Center Report BPM-06-08, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.

[14] F. Brito e Abreu, "Using OCL to Formalize Object-Oriented Design Metrics Definitions", Technical Report ES007/2001, INESC, Lisbon, Portugal, May, 2001.

[15] "Object Constraint Language (OCL)", Specification formal/06-05-01, v. 2.0, The Object Management Group (OMG), May, 2006.

[16] M. Richters, "A UML-based Specification Environment", http://www.db.informatik.uni-bremen.de/projects/USE, University of Bremen, 2001.

[17] J. Freitas, "Process and Service Metamodeling (in Portuguese)", *MSc dissertation,* F. Brito e Abreu (advisor), Departamento de Informática, Caparica, Portugal, FCT/UNL, 2010.

[18] "Meta Object Facility (MOF) Versioning and Development Lifecycle", Specification, v. 2.0, The Object Management Group (OMG), 2007.

[19] "ISO/IEC 20000-1: Information technology - Service management - Part 1: Specification", Standard, International Organization for Standardization (ISO/IEC), 2005.

[20] "ISO/IEC 20000-2: Information technology - Service management - Part 2: Code of practice", Standard, International Organization for Standardization (ISO/IEC), 2005.

[21] *ITIL Service Delivery Version 2.0*, London, UK, The Stationery Office (TSO), Office of Government Commerce (OGC), London, UK, ISBN:0 11 330017 4, 2003.

[22] J. C. Costa, "MGPSI – A Methodology for Information Systems Project Management (in Portuguese)", *MSc dissertation,* F. Brito e Abreu (advisor), Departamento de Informática, Caparica, Portugal, FCT/UNL, 2010.

[23] H. Zuse, "Software Complexity Metrics/Analysis", in *Encyclopedia of Software Engineering*, J. J. Marciniak (Ed.), John Wiley & Sons, Inc., pp. 131-166, 1994.

[24] S. Bryton and F. Brito e Abreu, "Modularity-Oriented Refactoring", in proceedings of the *12th European Conference on Software Maintenance and Reengineering (CSMR'2008)*, pp. 294-297, Athens, Greece, 2008.

[25] S. Bryton, "Modularity Improvements with Aspect-Oriented Programming", *MSc dissertation,* F. Brito e Abreu (advisor), Departamento de Informática, Caparica, Portugal, FCT/UNL, 2008.

[26] F. Brito e Abreu and M. Goulão, "Coupling and Cohesion as Modularization Drivers: Are we being over-persuaded?", in proceedings of the *5th European Conference on Software Maintenance and Reengineering (CSMR'2001)*, pp. 47-57, P. Sousa and J. Ebert (Eds.), Lisboa, Portugal, 2001.

[27] F. Brito e Abreu and M. Goulão, "A Merit Factor Driven Approach to the Modularization of Software Systems", *L'Objet*, vol. 7, n. 4, Hermes Penton Science, ISSN:2-84107-748-9, 2001.

[28] F. Brito e Abreu and W. Melo, "Evaluating the Impact of Object-Oriented Design on Software Quality", in proceedings of the *3rd International Software Metrics Symposium (Metrics'96)*, pp. 90-99, Berlin, Germany, 1996.

[29] F. Brito e Abreu and S. Bryton, "An Empirical Study on Refactoring Objects to Aspects", in proceedings of the *13th Workshop on Quantitative Approaches in Object Oriented Software Engineering (QAOOSE'2010)* (co-located with TOOLS'2010 - 48th International Conference on Objects, Models, Components, Patterns), Malaga, Spain, 2010.

[30] R. Porciúncula, "Governance and IT Process Modeling (in Portuguese)", *MSc dissertation,* F. Brito e Abreu (advisor), Departamento de Informática, Caparica, Portugal, FCT/UNL, 2010.

[31] J. Warmer and A. Kleppe, *The Object Constraint Language: Getting Your Models Ready for MDA*, 2nd ed., Addison-Wesley Publishing Company, ISBN:0321179366, 2003.

[32] A. Geraci, *IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries*, IEEE Press, ISBN:1559370793, 1991.

[33] V. Gruhn and R. Laue, "Approaches for Business Process Model Complexity Metrics", in *Technologies for Business Information Systems*, W. Abramowicz and H. C. Mayr (Eds.), Springer, pp. 13-24, ISBN:978-1-4020-5633-8, 2007.

[34] R. A. Kaimann, "Coefficient of Network Complexity", *Management Science*, vol. 21, n. 2, pp. 172-177, October, The Institute of Management Sciences, 1974.

[35] S. M. Henry and D. G. Kafura, "Software Structure Metrics Based on Information Flow", *IEEE Transactions on Software Engineering*, vol. 7, n. 5, pp. 510-518, September, IEEE, ISSN:0098-5589, 1981.

[36] A. Ghani, K. Muketha, and W. Wen, "Complexity Metrics for Measuring the Understandability and Maintainability of Business Process Models using Goal-Question-Metric (GQM)", *International Journal of Computer Science and Network Security*, vol. 8, n. 5, May, http://ijcsns.org/, 2008.

[37] T. McCabe, "A Complexity Measure", in *Software Engineering Metrics - Volume I: Measures and Validations*, M. Shepperd (Ed.), McGraw Hill Book Company, pp. 184-199, 1993.

[38] R. B. Grady, "Successfully Applying Software Metrics", in *Applying Software Metrics*, P. Oman and S. Pfleeger (Eds.), IEEE Computer Society, 1997.

[39] J. Cardoso, J. Mendling, G. Neumann, and H. A. Reijers, "A Discourse on Complexity of Process Models", in proceedings of the *BPI'06 - Second International Workshop on Business Process Intelligence (in BPM'2006)*, pp. 115-126, J. Eder and S. Dustdar (Eds.), Vienna, Austria, 2006.

[40] J. Cardoso, "Process control-flow complexity metric: An empirical validation", in proceedings of the *International Conference on Services Computing (SCC'06)*, pp. 167-173, Chicago, USA, 2006.

[41] B. Kiepuszewski, A. H. M. t. Hofstede, and C. J. Bussler, "On structured workflow modelling", in proceedings of the *Conference on Advanced Information Systems Engineering (CAiSE'2000)*, pp. 431-445, B. Wangler and L. Bergman (Eds.), 2000.

[42] M. Held and W. Blochinger, "Structured collaborative workflow design", *Future Generation Computer Systems*, vol. 25, n. 6, pp. 638-653, ISSN:0167-739X, 2009.

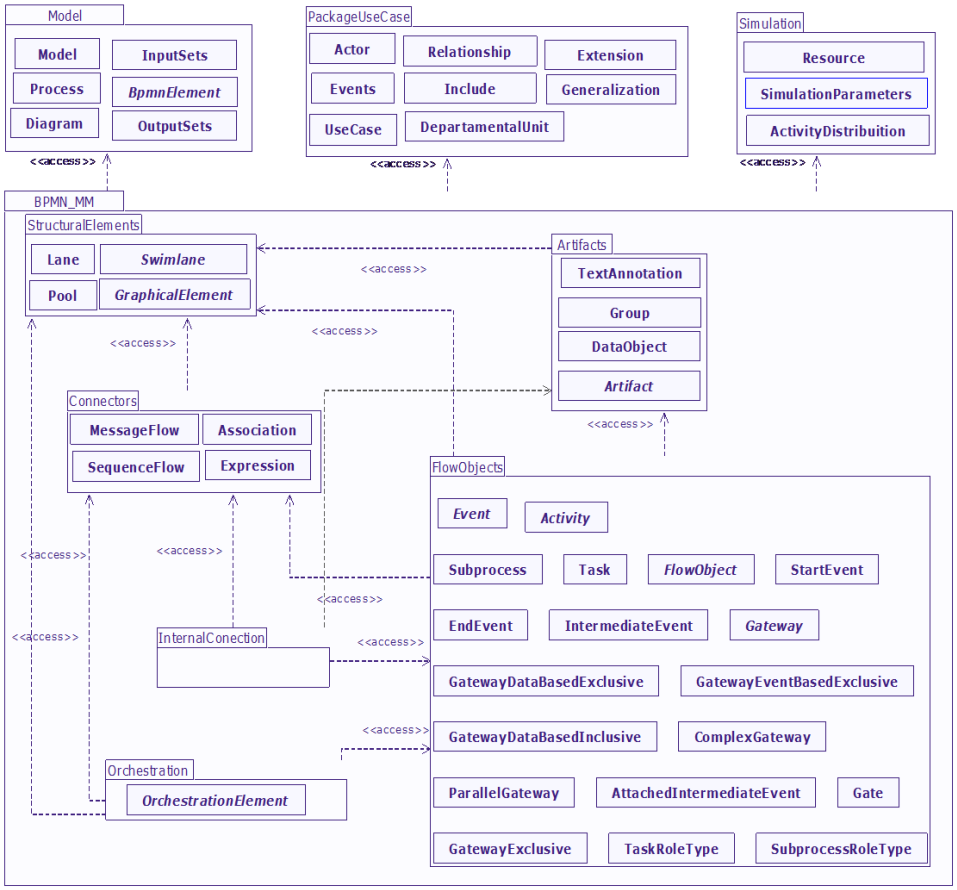
[43] A. Holl and G. Valentin, "Structured business process modeling (SBPM)", in proceedings of the *Information Systems Research in Scandinavia (IRIS 27)* t. I. S. R. S. i. S. (IRIS'27) (Ed.), Falkenberg, Sweden, 2004.

[44] E. W. Dijkstra, "Letters to the editor: go to statement considered harmful", *Communications of the ACM*, vol. 11, n. 3, pp. 147-148, March, ACM Press, ISSN:0001-0782, 1968.

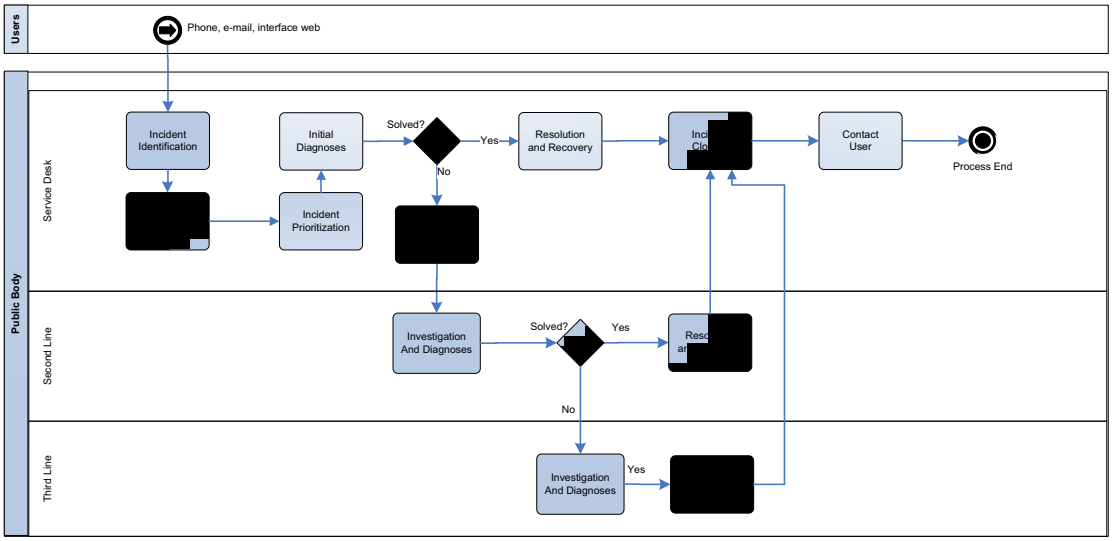
[45] W. Aalst, "The Application of Petri Nets to Workflow Management", *The Journal of Circuits, Systems and Computers*, vol. 8, n. 1, pp. 21-66, 1998.

[46] J. Caldeira and F. Brito e Abreu, "Influential Factors on Incident Management: Lessons Learned from a Large Sample of Products in Operation", in proceedings of the *9th International Conference on Product Focused Software Development and Process Improvement (PROFES'2008)*, A. Jedlitschka and O. Salo (Eds.), Rome, Italy, 2008.

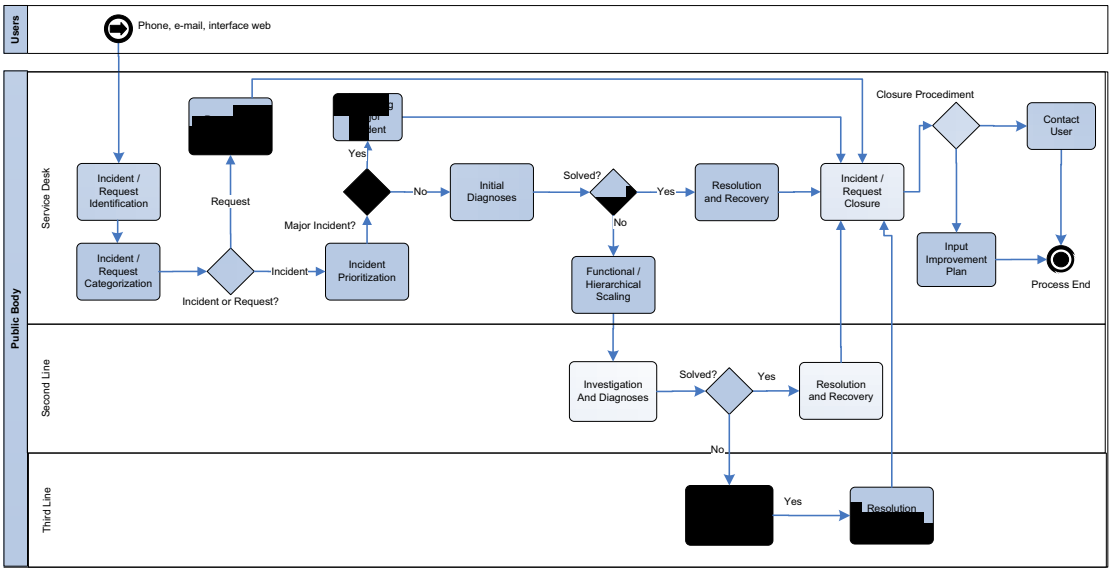
ANNEX A. BPMN METAMODEL (PACKAGE DIAGRAM)



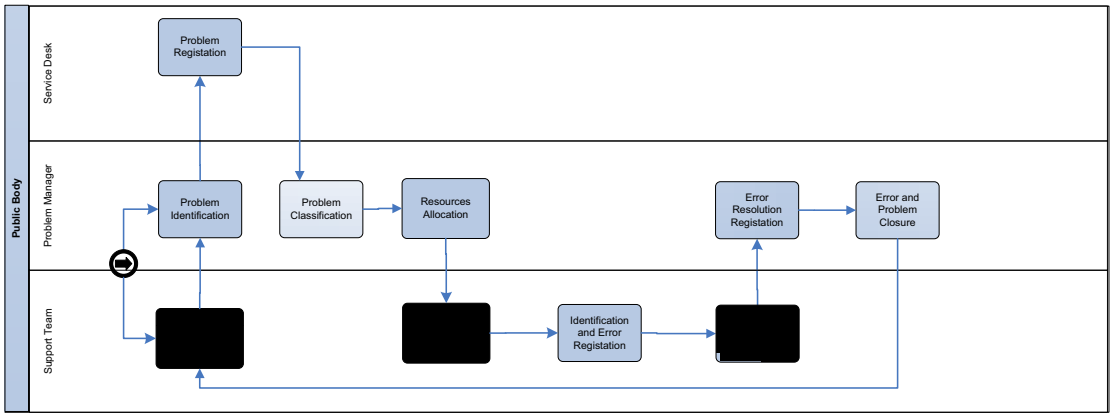
ANNEX B. INCIDENT MANAGEMENT PROCESS: "AS-IS" MODEL



ANNEX C. INCIDENT MANAGEMENT PROCESS: "TO-BE" MODEL



ANNEX D. PROBLEM MANAGEMENT PROCESS: "AS-IS" MODEL



ANNEX E. PROBLEM MANAGEMENT PROCESS: "TO-BE" MODEL