RODS: General Framework for *R*igorous *O*bservation of *D*istributed *S*ystems

Lorenzo Falai (1), Andrea Bondavalli (2)
(1) ResilTech Srl, Via Giuntini, 63, I-56023, Cascina (PI), Italy
lorenzo.falai@resiltech.com
(2) University of Florence, Viale Morgagni 65, I-50134, Firenze, Italy
bondavalli@unifi.it

Abstract

In this work we describe a methodology for monitoring and evaluating distributed systems. This methodology is supported by a rigorous conceptual framework, whose design was made using concepts from metrology science. Current distributed systems are heterogenous systems, characterized by the usage of several kind of hardware, operating systems, middleware software, networks and so on. To support experimental evaluation and monitoring of such heterogeneous systems, the framework here proposed is composed of a sequence of steps and a set of tools that can be used for evaluation and for monitoring of distributed systems composed of different kinds of nodes.

1 Introduction

Our society is faced with an increasing dependence on computing systems, not only in high tech consumer applications but also in areas (e.g., air and railway traffic control, nuclear plant control, aircraft and car control) where failures can be critical for the safety of human beings. Unfortunately, it is accepted that designing large fault-free computing systems is impossible.

As a consequence, the past years have seen a growing interest in methods for studying the behaviour of computer-based systems. Several approaches have been proposed to evaluate the *dependability properties* of a computer-based system. Fault Injection, i.e., the artificial injection of faults into a computer system in order to study its behaviour, emerged as a viable solution, and has been thoroughly investigated by both academia and industry.

In several context we need to observe the behavior of computing systems, mainly i) to monitor their actual behaviour (e.g. to on-line react to events), and ii) to evaluate their behaviour (e.g. to quantitatively assess a system).

What do experimental evaluation and monitoring of distributed systems have in common? We are *observing* and *measuring* system behavior. Thus, methodologies and tools for evaluation and monitoring of distributed systems can successfully use the conceptual framework and mathematical tools and techniques made available by an old science such as metrology, looking these methodologies and tools as what they really are: *measuring instruments*.

Distributed systems offer several advantages - possibility of enhanced reliability due to inherent redundancy, increased performance by concurrency, and better resource utilization through sharing. The observation of the behavior and the measurement of computing dependability characteristics of distributed systems has suffered for lack of an adequate conceptual framework. The main problem of evaluating computer systems in general is their complexity and the fact that they usually have not been designed to be monitored. In distributed systems things are even more complex due to the lack of central control, precise global time and an accurate view of the global state.

2 Background

This Section firstly defines the main concepts related to monitoring and experimental evaluation of distributed systems and the difficulties that characterize these actions. Later, the Section describes main concepts of metrology (science of measurement), that has proved very useful in the application to several fields of science, offering a strong conceptual framework that can be useful also for evaluation and monitoring of distributed systems.

2.1 Observing Computing Systems

In several context we need to observe the behavior of computing systems, mainly i) to monitor their actual behavior and ii) to assess/validate their behavior. In both contexts more precise the observation of the system are, more reliable the collected results are and thus more precise the decisions we are able to take are. In on-line monitoring we are able to react in a more appropriate way, in the evaluation

we are able to take decisions using more precise collected information.

Certain problems associated with observing a centralized system are exacerbated when dealing with distributed systems. A complete overview of problems in monitoring can be found in [10]. Here we enumerate the main problems from the point of view of observing distributed systems, compared to centralized systems:

1. No central point of control
2. No central point of observation
3. Multiple sources of monitoring information
4. No central point of decision making
5. Incomplete observability: in some cases it is not possible to observe certain parts of the system at all or only partially
6. Non-determinism: distributed, asynchronous systems are inherently non-deterministic; this makes the reproduction of errors and the creation of certain test conditions difficult, if not impossible, at times
7. Observation interference: the inclusion of monitoring in a distributed system can alter the behaviour of a program in a manner which is important to the observer
8. Objects and encapsulation: encapsulation, one of the fundamental characteristics of object-oriented systems, is directly opposed to monitoring
9. Objects administer their own management: in contrast to centralized systems which are characterized by a central management entity, management facilities are distributed to the objects.

2.2 Metrology

As modern science is based on experimental evidence [8], the science of measurement, i.e. modern *metrology*, has nowadays reached an adequate level of maturity and has proved very useful in the application to several fields of science. In particular, it has developed theories as well as good practice rules to properly make measurements and evaluate measurement results, and to correctly characterize measuring instruments and assess their characteristics. Also methodologies and tools for evaluation and monitoring of computing systems can successfully use the conceptual framework and mathematical tools and techniques made available by metrology.

Here we briefly present fundamental concepts related to characterize measurement systems and methods according to metrological criteria. Complete digests of metrological terms and concepts can be found in [4], [2], [3] and [8]. In a recent work, [6], the authors analyzed the state of the art of the tools available for experimental dependability evaluation seen from a metrological point of view.

Measuring a quantity (namely the *measurand*) consists in quantitatively characterizing it. The procedure adopted to associate quantitative information to the measurand is called *measurement*. The measurement result is expressed in terms of a measured quantity value and it must always include a related measurement uncertainty.

- **Accuracy** must be intended only in a qualitative way. It represents closeness of the measure to the best available estimate of the measurand value.
- **Uncertainty** provides *quantitative* information on the dispersion of the quantity values that could be reasonably attributed to the measurand and it has to be included as part of the measurement result. [2] contains a comprehensive report of the methods of evaluating and expressing uncertainty in measurements.
- **Resolution** is the ability of a measuring system to resolve among different states of a measurand. It is the smallest variation of the measurand that can be appreciated, i.e. that determines a perceptible variation of the instrument's output.
- **Repeatability** is the property of a measuring system to provide closely similar indications in the short period, for replicated measurements performed i) independently on the same measurand through the same measurement procedure, ii) by the same operator, and iii) in the same place and environmental conditions.
- It is well known that any measurement system perturbs the measurand, determining a modification of its value. Minimizing such perturbation, that is minimizing the system's **intrusiveness**, is therefore desirable when designing a measurement system.

3 RODS

The Section describes RODS, the General Framework for *R*igorous *O*bservation of *D*istributed *S*ystems. In the first part we identify the main goals that we want to obtain in the framework, and later we describe an high level view of RODS.

3.1 Identification of the Goals

The main characteristic of a framework supporting a rigorous monitoring and experimental evaluation of a large class of distributed systems can be summarized as follows.

- The framework must be usable both for monitoring and for experimental evaluation of distributed systems. The difference in the usage of the framework for these two different purposes must be minimal.
- The framework must describe in a rigorous and unambiguous way how to perform monitoring and experimental evaluation of distributed systems.
- The framework must be flexible and its general concepts must be platform-independent. The framework must present how to perform monitoring and experimental evaluation from a high-level perspective; the several steps of the evaluation should work in the largest possible set of nodes that compose the distributed system under analysis.
- The framework must be simple to use.
- The framework must fulfill the following guidelines (identified in [6]):
  1. The measurands should be clearly and univocally defined.
  2. All the sources of uncertainty should be singled out and evaluated.
  3. Uncertainty obtained in the measurement should be evaluated and provided as part of the results.
  4. Intrusiveness of on-line supports should be maintained as low as possible and should always be evaluated.
  5. Resolution should be evaluated.
  6. Comparison of measurement results provided by different tools/experiments should be made in terms of compatibility.
- The framework must be scalable.
- The framework must be efficient and effective.
- The framework must provide ways to perform monitoring and experimental evaluation of distributed systems; the observable components of the system must include the following:
  1. user-level software components (source code can be available and modifiable or not);
  2. operating system-level software components (source code can be available and modifiable or not);
  3. hardware components for which appropriate probing tools are available.
- The results of the analysis obtained using the framework must be portable.
- When the framework is used for experimental evaluation, it must support execution of reproducible and repeatable experiments.

3.2 High Level Overview of RODS

Figure 1 depicts the high level view of the proposed framework. The figure provides a view of the required process for performing *rigorous observations* of distributed systems in terms of a **data flow** diagram. The figure is the guideline for this section. It depicts the necessary steps to rigorously perform monitoring and experimental evaluation of distributed systems. In the following we describe in detail the process corresponding to each step, in terms of input data, processing and output. In each step we present the main differences between performing monitoring or experimental evaluation of a system: as previously depicted, we designed the framework in order to minimize the differences between these two processes. Since several parts are common between monitoring and experimental evaluation processes, we use the word "analysis" instead of "monitoring or experimental evaluation", and the verb "to analyze" instead of "to monitor or to experimental evaluate". The RODS user is identified as "analyzer".

Figure 1 describes the process of analysis of a distributed system following the RODS methodology as a sequence of steps using a *dataflow diagram*: the circles represent **functions**, the boxes represent **input/output**.

The sequence of steps of the analysis is the following:

1. **Definition of the object of the analysis.** This phase is depicted as S1 ("Step 1") in Figure. The analyzer must define:

   SYS: the distributed system to analyze; SYS can be an existing system or a prototype of a future system. It is composed by both hardware and software.

   SYSREQS: it is the description of the requirements of the distributed system to analyze. Requirements should be concise, complete, unambiguous, verifiable, and necessary. SYSREQS can include both functional and non-functional requirements. SYSREQS must be provided in terms of a list of requirements, written in natural language.

2. **Identification of the goals of the analysis.** This phase is depicted as S2 in Figure. In this phase the analyzer has to identify in a clear and unambiguous way what the goals of the evaluation are. In this phase we must identify:
   (a) what we want to analyze and assess
   (b) what data have to be collected from the system
   (c) how to collect the data

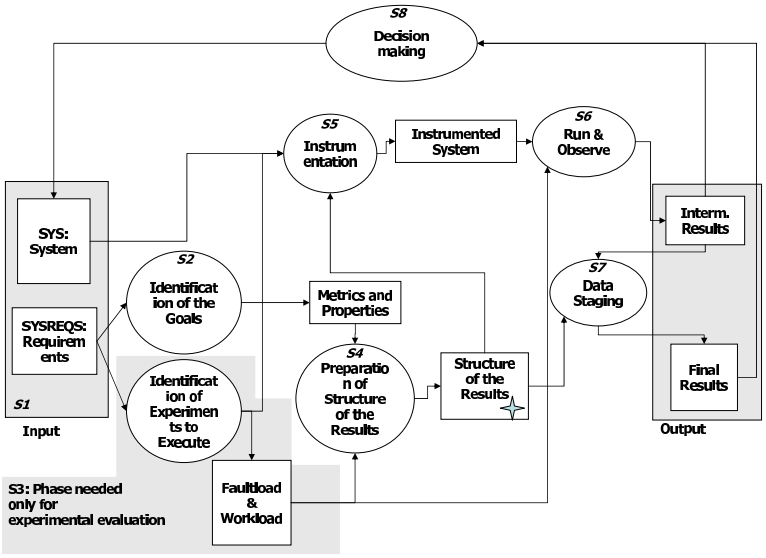

Figure 1. High level overview of the framework: dataflow

The output of this phase is a description of the metrics and properties of the system that we want to analyze. The purposes of the analysis can be mainly identified in two large classes:

(a) Testing of on/off *properties* of the system

(b) Quantitative assessment of properties of the system in terms of *metrics*

In the first case we are interested in the assessment of the validity of *qualitative properties*. Example of such properties are: ”termination”, ”validity”, ”integrity”, and ”agreement”. In the second case the metrics that we identify in this phase depend on the final results that we want to obtain from analysis. Metrics are in general *QoS (Quality of Service) attributes*, including performance and dependability related attributes.

With respect to ”How to collect the data”, we have mainly two ways of obtaining data from the system: i) online automatic logging or ii) human manual logging.

The output of this phase is a description in terms of On/Off properties and Qos metrics that are the purposes of the analysis. These data must be provided in terms of a description containing the following information:

(a) Name of the metric/property

(b) Intent

(c) Description

(d) Motivation for its evaluation

(e) Applicability

(f) Evaluation Procedure

(g) Influencing Factors (if applicable)

(h) Result Type: floating point, boolean, ...

(i) Evaluation Formula (for metrics)

(j) Direct/Composed (for metrics)

This information will be useful for the definition of the structure of the results and for the instrumentation of the system under analysis.

3. **Identification of the experiments to execute.** This phase is identified as S3 in Figure. It is specific only to experimental evaluations. In this phase the analyzer concentrates on the definition of the experiments to execute, in particular:

   (a) the fault injection scenarios to consider

   (b) the characteristics of the workload that we want to use in our assessment

   (c) the system parameter to setup in the experiments.

   When dealing with fault injection experiments, we have to define what kinds of faults are to be injected, how to inject them and what type of target system activity is to be considered. The choice of a particular class of faults to inject has an impact on the choice of the tool to use to inject them (tool used in the ”Instrumentation” phase).

   The workload is the computational load for the SYS. The definition of a workload depends on which system component is analyzed.

In this phase the analyzer also has to define the number of observations for each metric and property, or in any case the duration of the experiments.

4. **Preparation of the structure of the results.** This phase is identified as S4 in Figure. In this phase the analyzer defines the structure of final results of the analysis. In case of experimental evaluation, the final results also contain a description of identified workload and faultload. Recently Madeira et Al. ([11]) proposed the usage of OLAP (On-Line Analytical Processing) and Data Warehousing approaches to store raw results from experiments in a common multidimensional structure; from this structure raw data can be analyzed and shared between several research groups world wide, by means of OLAP tools. In our framework we follow this idea, proposing to structure and save results of performed analysis in a data warehouse. In the usage of data warehousing for analysis of distributed systems, the fundamental idea can be summarized in the following concept: the readouts collected during the experiments represent the data **facts** of the multidimensional model, while contexts in which analysis is performed represent the **dimensions**. Figure 2 depicts the typical conceptual star-schema schema for final results of an analysis. Notice that we can have several "fact" tables, in general one for each different evaluated metric.

   Data warehouse and OLAP tools are really useful for collection and evaluation of large sets of experimental results, and they can be used for collection of historical data (e.g. data of several experiments performed in different scenarios). However, when the purpose of the analysis is monitoring a distributed system in order to be able to promptly react to, these tools can be too intrusive and lighter supports can be used to collect intermediate results to use in the decision making phase (e.g. text log files and/or databases supporting a real-time data stream analysis as described in [1]).

   It appears clear that the definition of the structure of results is made as early as possible. This early definition of the structure of the results is useful since it allows us to clearly define, in an unambiguous way, the purposes (and contexts) of the analysis. The structure of the results is afterwards used as input for the "Instrumentation" phase.

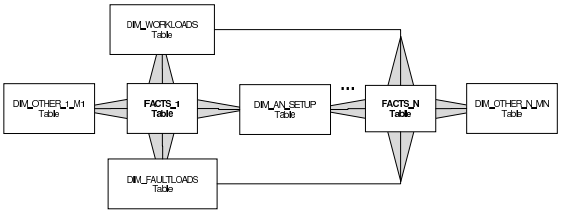

Figure 2. Proposed structure for the evaluation

5. **Instrumentation of the system.** This phase is identified as S5 in Figure. In this phase the analyzer instruments the system with probes. The choice of the probes to use is based on i) components to analyze and ii) properties and metrics of interest (defined in the structure of the results). In case of experimental evaluation, in this phase we also instrument the system to be able to execute it under specified workloads and faultloads. There are several tools already available, made for observing and measuring specific systems and for fault injection of specific classes of faults ([6] provides an overview of several available tools and it provides their characterization from a metrological point of view). Since the kind of distributed systems and the contexts in which they could be analyzed is wide and each tool is specific for a certain class of systems and contexts, we decided not to constrain our framework to a specific tool.

   Whenever we want to obtain direct or indirect measurements in which distributed time interval is involved, we have to pay great attention to uncertainty estimation in obtained results. In case measurements of time interval must be done, the analyzer has to integrate and use a Reliable and Self-Aware Clock component (see Figure 3). [5] contains a complete definition of Reliable and Self-Aware Clock component, and [7] defines how it can be used as support for uncertainty estimation of time metrics. The usage of this component allows the analyzer to obtain "enriched timestamps" instead of standard timestamps. The enriched timestamps contain information on the quality of the actual clock synchronization. It is of uttermost importance to allow the analyzer to have control of the uncertainty of each measure attained through comparison of timestamps obtained in different nodes of the distributed system.

6. **Run and Observe.** This phase is identified as S6 in Figure. In this phase the instrumented system runs and data is collected and exported. The output is represented by collected events. These results are not yet structured using the star-schema proposed for the processing of outcomes.

   Two cases exist:

   - in a monitoring application the output of this phase is a continuous flow of intermediate results that can be exported on file and at the same time they can feed a Decision Making algorithm;

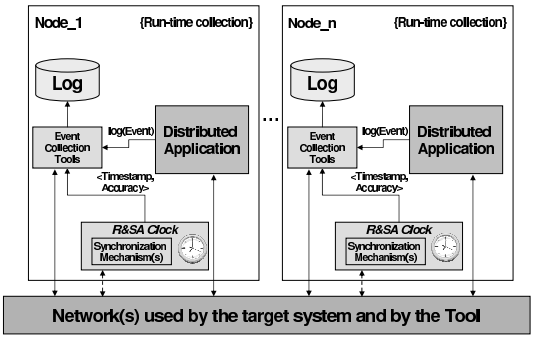

Figure 3. Instrumenting the system with a Reliable and Self-Aware Clock

- in case of experimental evaluation the output of this phase are timestamped events, that are usually stored in a memory support.

7. **Data Staging**: in this phase the collected data are cleaned, combined, archived and eventually exported using the structure for the final results defined in phase "Preparation of the structure of the results". Identified as S7 in Figure.

8. **Decision Making**: in this phase data are analyzed in order to perform appropriate action on the system under analysis. These actions can be done on-line or off-line. If the purpose of the analysis is only an experimental evaluation (e.g. for comparison of QoS of variants of an algorithm), this phase can be optional. Identified as S8 in Figure.

4 Conclusions and Future Work

Summarizing, the framework described in this chapter, RODS, is a general usable way to perform analysis (both monitoring and experimental evaluation) of distributed systems, following a rigorous methodology that allows us to obtain results on which we can trust.

The proposed framework is a conceptual framework, that allows the usage of several different tools and methods available in mathematical and computer science literature. In the framework we identified the several steps needed to perform rigorous analysis of distributed systems, in terms of a dataflow. We used concepts taken from metrology science for its design and assessment: we used concepts like repeatability of measurements, need of an evaluation of the uncertainty and resolution and intrusiveness of used probing tools.

We are currently exploring the usage of the framework in a first case study. The idea is to use RODS for experimental evaluation and comparison of the Quality of Service of a large set of failure detectors, extending a first work made in this direction described in the paper [9].

5 Acknowledgments

This work has been partially supported by the AMBER CA, funded by the EU. We would like to thank also Michele Vadursi and Andrea Ceccarelli, for work made together and for the useful discussions about the possible usage of metrology in computing systems evaluation.

References

[1] Streambase technical documents. Webpage, March 2008. StreamBase Home Page. http://www.streambase.com.

[2] BIMP, IEC, IFCC, ISO, IUPAC, IUPAP, and OIML. *Guide to the expression of uncertainty in measurement*, second edition, 1995.

[3] BIMP, IEC, IFCC, ISO, IUPAC, IUPAP, and OIML. *Guide to the expression of uncertainty in measurement, supplement 1, numerical methods for the propagation for distributions*, 2004.

[4] BIMP, IEC, IFCC, ISO, IUPAC, IUPAP, and OIML. *ISO International Vocabulary of Basic and General Terms in Metrology (VIM)*, third edition edition, 2004.

[5] A. Bondavalli, A. Ceccarelli, and L. Falai. A self-aware clock for pervasive computing systems. In *The Fifteen Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP 2007)*, pages 403–411, Febraury 7-9 2007.

[6] A. Bondavalli, A. Ceccarelli, L. Falai, and M. Vadursi. Foundations of measurement theory applied to the evaluation of dependability attributes. In *DSN-2007 IEEE Int. Conference on Dependable Systems and Networks*, pages 522–533, June 25-28 2007.

[7] A. Bondavalli, A. Ceccarelli, L. Falai, and M. Vadursi. Towards making nekostat a proper measurement tool for the validation of distributed systems. In *Proceedings of The 8th International Symposium on Autonomous Decentralized Systems*, pages 377–386, March 2007.

[8] F. Dyson. The inventor of modern science. *Nature*, 400:27, July 1999.

[9] L. Falai and A. Bondavalli. Experimental evalutation of the QoS of failure detectors on Wide Area Network. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005)*, pages 624 – 633, Yokohama, June 2005.

[10] Y. Hoffner. Monitoring in distributed systems. Webpage, December 1994. ANSA project architecture. http://www.ansa.co.uk/.

[11] H. Madeira, J. Costa, and M. Vieira. The olap and data warehousing approaches for analysis and sharing of results from dependability evaluation experiments. *dsn*, 00:86, 2003.