*10th IMEKO TC7 International Symposium*
*June 30–July 2, 2004, Saint-Petersburg, Russia*

# DISTRIBUTED PROCESSING OF MEASUREMENT RESULTS IN A CONTROL SYSTEM: CAN MOBILE SOFTWARE AGENTS HELP?

*Danila S. Smolko*

Saint-Petersburg State Polytechnic University, Saint-Petersburg, Russia

**Abstract** – Distributed processing of measurement results poses a number of challenges. Among important requirements are security, performance, robustness and ability to withstand network failures, need for flexibility and elegance of a proposed solution. We propose a novel approach to distributed processing, which capitalises on locality of data access. We propose a distributed software architecture that leverages a flexible consistency checking engine and takes advantage of software agency features in order to meet the mentioned distributed processing requirements.

**Keywords:** distributed control systems, verification of consistency relations, software agent architecture

## 1. DISTRIBUTED PROCESSING OF THE MEASUREMENT RESULTS

The measurement processing resulted in modern control systems is characterised by large volumes of data, stringent restrictions on processing time and distributed nature of the measurement process. Functionally, the processing involves verification of pre-defined relations between data, for which we use the term “consistency checking”. A review of existing distributed data processing techniques [3] demonstrates lack of an elegant and simple way to check consistency relations and identify inconsistencies between data in a distributed setting.

## 2. CONTRIBUTION AND NOVELTY OF THE SOFTWARE AGENT ARCHITECTURE

Our novel approach to inconsistency identification performs consistency checks at the location where data resides. There is no need for transmission of complete data sets across networks to a central processor. This factor alone brings numerous advantages to the approach, amongst which data security and processing speed are increasingly important.

We propose and evaluate a distributed architecture for consistency checking, that makes use of mobile and static software agents and can be used to combat GIGO (garbage in-garbage out) effects [1], a common plague of modern distributed measurement and control systems.

The approach facilitates establishment of relations (links) between related distributed resources [6] without transmitting their complete content across the network [3]. It capitalises on locality of access to resources, mobility of the consistency checking agents, and provides an architecture, which establishes the infrastructure necessary to take advantage of access locality and component mobility [2,5]. We present our approach in a scenario, describe the implementation prototype of the agent architecture, and give qualitative and quantitative evaluation to the distributed consistency checking architecture.

## 3. ACHIEVED RESULTS

### *3.1.Consistency checking rule language*

The task of a control system is to use the distributed data as inputs, and produce the output depending on a relation or a set of relations between the inputs. In our approach, measurement data is represented in a structured format, eXtensible Modeling Language (XML) according to a data type definition DTD or an XML Schema. Consistency rules, which specify the required relationships between the data types [6], have to be produced for a particular application domain. The rules are represented in a language [7] that extends XML and is based on first-order logic. Each data location will host a software agent middleware as defined by the distributed software agent architecture.

### *3.2. Scenario: Modelled Relations in an IEC Device Profile Specification*

Expressiveness of the XLinkit rule language [7], which is adopted as an underlying engine for consistency checker for our distributed architecture, allows defining a wide range or constraints and relations between structured data in XML. XML has been eagerly adopted within the measurement and control application domain. Use of XML for IEC Device profiles [8] has been recommended by the IEC Industrial Automation Sector Board [9].

All principal concepts defined in the IEC Device profile specification [8] are available to be modelled in XLinkit language, and therefore become eligible for checking using the proposed approach. For instance, communication and application parts of a “Device profile” define for each particular device a set of

constraints on the data types of the object attributes or block data input, data output or parameter of that device. Data semantics in the Device profile is defined by the characteristic features (parameter attributes) of the application data: this can be data name, data descriptions, the data range, Substitute value of the data, default value.

Consistency rules for the Device profile example express and validate individual values of parameters as well as compare parameter sets during the operation of the device within a control system. As result of consistency check using proposed tooling and architecture, links are created between consistent elements, while constraints are satisfied, or between inconsistent elements if constraints are violated. Our approach is very un-intrusive, as these 'consistent' and 'inconsistent' links are saved separately from the structured XML data that is being checked. It is a clear advantage that our approach allows original data to remain intact. Subsequently, inconsistent links can be automatically navigated by any XPath processor [10], in order to retrieve all required original parameter values from the measurement datasets.

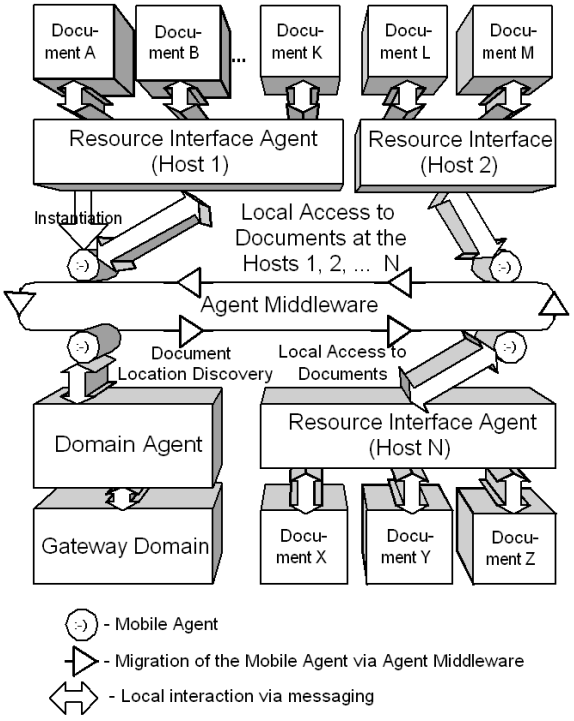


Fig. 1 – Software agent architecture for consistency checking of distributed resources/documents

More complex constraints of the Application Functionality part of the Device profile are defined by specifying the dependencies and consistency rules within the functional elements. Dynamic performance is defined by time constraints that influence the data or the general device behaviour. Both functionality and performance are examples of dynamic constraints, that relate to the behaviour of a device. If such constraints are not covered by expressive capabilities of first-order logic of the XLinkit engine itself, then such constraints can still be modelled in the XLinkit language by algorithmically coding constraints using operators and plugins to encapsulate relationship complexity. The results of dynamic constraint checks are also presented by means of consistent and inconsistent links of un-intrusive nature.

*3.2. Architecture*

The scenario in section 3.1 discussed usage of the XLinkit language for defining constraints, and introduced notions of consistent and inconsistent links that are results of checks of those constraints. Our primary contribution is in constructing and evaluating a distributed software agent-based architecture that drives the consistency checking engine and allows consistency checks to span distributed resources.

We have proposed an open and extensible architecture [5,2,3], based on stationary components-agents that provide services locally at each distributed site, and mobile software agents, capitalizing on autonomy and migration capacity that effectively consume the stationary services during a consistency check of distributed data resources [5,2]. Analysis of some open problems in the field of distributed consistency management [2,3] necessitated functional decomposition of a distributed consistency check into a number of distinct agent roles that could be autonomously executed. Multiple instances of a domain agent, gateway domain for interconnection of domains, resource interface agent and a mobile consistency checking agent comprise the architecture (Fig. 1).

To combat the data-sizing problem, we have proposed an incremental checking approach that drastically improves performance of distributed checks while guaranteeing correctness of all established consistency relations [5].

## 4. EVALUATION

We have conducted a comprehensive evaluation of the software agent architecture for distributed consistency checking [5]. A developed state transition model expresses the nature of algorithms of each component and their interactions. An implementation prototype of the architecture and its operation is described in detail in [5] on the three practical scenarios of varying data sizes. These scenarios represented real world distributed system configurations of increasing complexity.

*4.1 Data access locality*

Access to measurement data and identification of consistency relations between elements occurs at the location of the data, rather than in a distributed fashion across the network. The proposed software agent architecture enables to partition consistency checks that span multiple distributed resources into sets of

local checking activities, at the location of each resource. Fig. 1 demonstrates a consistency check that spans several hosts in the domain (Host 1 ... Host N) that are accessible to mobile checking agents through agent middleware. Mobile agents extract all data for rule checks from documents/resources that are accessible at the current Host, where mobile agent is running: documents are never accessed remotely.

Local checks ensure maximum checking performance without network transfer overhead, and allow the architecture to answer to stringent security constraints.

*4.2 Distributed checks and data access locality*

Inter-resource relations are checked by mobile agents that migrate between nodes in a network. The migration algorithm of a mobile consistency checking agent is based on sending the current state of agent code and any non-transient data across the network in a form of a message. Significant performance advantages of this novel approach are achieved in comparison to the traditional migration, when transmission of the agent's code itself occurs. At the receiving end, a running listener initialises a "child" instance of a mobile checking agent. This new instance restarts the checking algorithm from the same state as its parent, and communicates results back to the parent via messaging. In short, the approach taken by mobile agent architecture selectively leverages advantages of "strong" mobile agent migration, without compromising performance.

*4.3 Rule prioritisation helps achieve real-time checks*

Consistency rules that are checked by software agents are prioritised in a scenario according to rule importance, which is set by system administrator. In a conducted performance evaluation, performance of higher priority rules was satisfactory to meet real-time requirements for data control. Incremental consistency checks, which only take into account data that actually changed since the last run, allowed additional significant improvement of the checker performance. It was not uncommon to achieve 10Mb/sec throughput on XML data during a repeated incremental check in the conditions of streaming XML data.

*4.4 Disconnected operation*

One of unique advantages of the proposed architecture is its capability to support disconnected operation. A domain-based model groups more tightly related resources into domains, where majority of consistency checks happen within this domain. If a network failure or a planned disconnect occurs, any in-domain checks are not affected by disconnection. Inter-domain checks complete all required processing of in-domain resources and are saved in a queue awaiting reconnection of external domains. The partial results of inter-domain checks are available to be consulted within the disconnected domain.

This disconnected operation capability ensures that once a distributed measurement system is configured in such a way, that most related resources are grouped in a logical domain, disconnection of this domain will not affect internal checks. It important to take into account physical network configuration. Since the largest part of messaging traffic (subsection 4.2) is concentrated within the domain, physical network links between domain nodes should be able to meet quality of service requirements.

*4.5 Quantitative evaluation*

The conducted evaluation has confirmed elegance and flexibility of the proposed software agent architecture, established by the degree of intuitiveness and autonomy of mobile checking agents. Quantitative performance evaluation of the prototype has defined the conditions, where significant performance advantages of the incremental checking approach are realised [5].

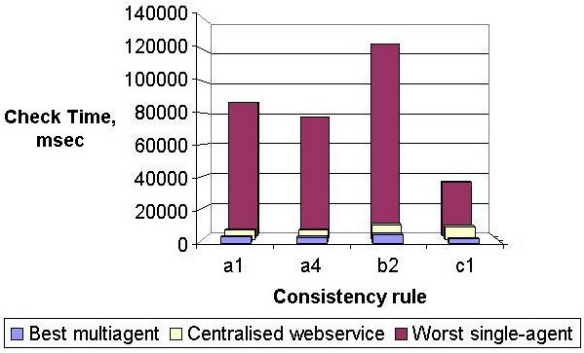


Fig. 2 – Multi-agent cooperative check performance

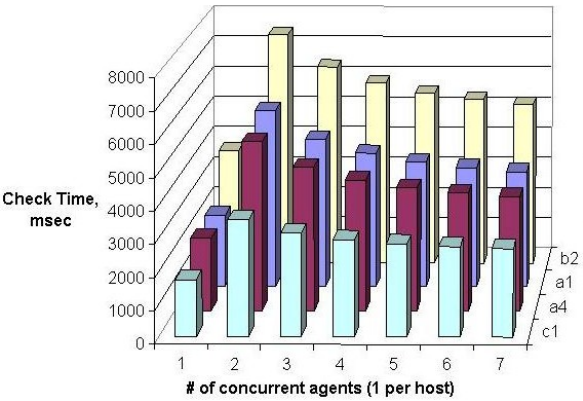


Fig. 3 – Quantitative performance evaluation of distributed multi-agent architecture.

Multiple-agent concurrent distributed checks have proven to achieve performance improvements [5] over traditional centralised, server-side processing techniques for consistency checking [7]. Fig. 2 compares performance data for best multi-agent concurrent check of a consistency rule that spans multiple distributed resources, with centralised check of the same rule (a single server, or so called

"webservice" check). The worst single-agent result is also provided (Fig. 2), where performance in case of multiple migrations of a single checking agent is in orders of magnitude worse than similar work done in a concurrent and cooperative fashion in a multi-agent mode. However, a multi-agent check performance appears to deteriorate after a certain number of concurrent agents, and this number depends on the actual content of a consistency rule being checked, number of nodes where related resources are located, and performance of the nodes themselves (Fig. 3). An algebraic performance model of a distributed cooperative multi-agent check has been proposed [5] to allow more formal estimation of performance advantages of a multi-agent check.

## 5. CONCLUSIONS AND FUTURE WORK

The proposed software agent architecture for distributed checking of consistency relations is a novel application of the concept of mobility in the domain of distributed measurement and control systems. The proposed technology is particularly suitable to this domain, in which the sheer size, confidentiality or timeliness of the data render transport of this data to a central location for processing too expensive or impossible. The suggested architecture tackles many facets of this problem by ensuring local data access and processing.

## REFERENCES


[1] Smolko D.S., Mobile Agent Architecture For Portfolio Management, In Proceedings of International Conference SCM-2001, St. Petersburg, Russia, pp. 208-211, v.1, 2001.

[2] Smolko, D., Design and Evaluation of the Mobile Agent Architecture for Distributed Consistency Management. In Proc. 23rd Int. Conference on Software Engineering, Toronto, Canada. Doctoral Symposium, pp. 799-801, IEEE Computer Society, 2001.

[3] Smolko, D., Consistency Issues in Document Management of Distributed Group Work. In Proceedings of Doctorate Symposium, Automated Software Engineering '99, 14th IEEE International Conference, Orlando, USA, IEEE Computer Society, 1999.

[4] Finkelstein, A. and Smolko, D., Software Agent Architecture for Consistency Management in Distributed Documents. In Proceedings of SCI/ISAS 2000 - 6th International Conference on Information Systems Analysis and Synthesis, Orlando, USA, vol. IX, pp. 715-719, International Institute of Informatics and Systemics, July 2000.

[5] Smolko, D.S., Software Agent Architecture for Consistency Checking of Distributed Documents, Ph.D. Thesis, University College London, University of London, UK, January 2002.

[6] Zisman, A., Finkelstein, A., Ellmer, E., Smolko, D., and Emmerich, W., Method and Apparatus for Monitoring and Maintaining the Consistency of Distributed Documents. International Patent, The U.K. Patent Office Patent No GB2351165, University College London (Application No GB9914232.5, 18/06/1999).

[7] Nentwich, C., Emmerich, W., and Finkelstein, A., http://www.xlinkit.com and http://www.systemwire.com, 2000-2004.

[8] International Electrotechnical Commission (IEC), TC65: Industrial Process Measurement And Control: Device Profile Guideline. Document 65/290/DC, March 2002, available from http://www.holobloc.com/stds/iec/tc65wg6/liaison/dptf/65_290e_DC.pdf

[9] ISO/TC 184/SC 5, Em de la Hostria, Rockwell Automation, USA, Enterprise-reference architectures, Application integration frameworks and Interoperability profiles. Business Object Summit (BOS), 27-28 November 2000, United Nations, Geneva, CH.

[10] XML Path Language (XPath) Version 1.0, W3C Recommendation, 16 November 1999, available from http://www.w3.org/TR/xpath



**Author:** Dr. Danila S. Smolko, Saint Petersburg State Polytechnic University, Computer Science and Engineering Faculty, Department of Information and Control Systems, 195251 Saint Petersburg, ul. Polytechnicheskaya 21, Saint Petersburg, Russia. Electronic mail: smolko@list.ru.