A Constraint-Based Approach to Fault Management for Groupware Services

M. Sabin[1] *A. Bakman*[2] *E. C. Freuder*[1] *R. D. Russell*[1]

[1]*Computer Science Department*
University of New Hampshire
Durham, NH 03824
mcs,ecf,rdr@cs.unh.edu

[2]*Candle Corp.*
75 Market Street, Suite 302
Portland, ME 04101
Alex_Bakman@candle.com

Abstract

There is no standard model at the service layer. However, fault management to distributed services and applications needs to construct and utilize complex models of the participating objects and their interdependencies. Thus, model-based fault management tools can predict the correct behavior of diagnosed systems and use the resulting predictions to identify faults. When used on-line in real systems, diagnostic tools based on such models should be able to provide prompt response and accurate, comprehensive explanations of the root causes of faults. In this paper we propose to address these requirements: modeling, proactive diagnosis, and explanation. We apply a recent extension to the constraint satisfaction paradigm, called *composite constraint satisfaction*, to facilitate modeling of complex systems, and we use *constraint propagation techniques* to support proactive diagnosis and explanation. We demonstrate the applicability of our approach on an example of a basic groupware service, namely, distributed database replication.

Keywords

Fault management, constraint satisfaction, groupware service, modeling, proactive diagnosis, explanation.

1. Introduction

Modeling network resources represents the first, necessary step to developing network and system management tools. However, the Managed Objects (MOs) are not isolated from each other. They exhibit relationships that reflect the dependencies among the underlying network resources. The OSI General Relationship Model (GRM) provides the relationship construct to support the description of the behavior of the interrelated MOs that participate in a relationship. Work by [1], [2], and [3] demonstrates how GRM can leverage management and conformance testing, and will most likely speed up the standardization process. Some questions, however, remain open. As has been pointed out in [1], modelers of management information have still to decide "when and where relationships have to be defined". In

other words, what collection of relationships best reflect and respond to management problems? Moreover, can the management information modelers benefit from a knowledge-based language embedded in the standardized specifications? These questions suggest that the design of suitable management services should consider the representation and reasoning techniques used by knowledge-based management systems. Among such systems, model-based systems have gained a rapidly increasing acceptance for use in event and alarm correlation [4], [5], [6], as well as for fault and configuration management of distributed services and applications [7], [8]. The latter category poses new challenges, as there are no standard models at the service layer. Therefore, management tools should provide flexible and powerful modeling schemes along with effective reasoning support.

The constraint satisfaction paradigm meets these challenges. In order to address hard combinatorial problems, such as diagnosing network services, constraint satisfaction integrates modeling, search, and propagation. The contribution of this paper is to:

1. facilitate modeling of complex systems such as groupware services, by employing advanced constraint satisfaction modeling concepts: composition, local models, and parameterized relationships or ports,
2. use constraint propagation to (1) support proactive diagnosis of the configuration settings and the running service, and to (2) provide explanations in terms of the root causes of the signaled or predicted fault.

The proposed approach enhances the fault management capabilities of the diagnosticians described in our previous work [9] as follows. First, the modeling scheme, called *composite constraint satisfaction* [10], embeds object-oriented design principles to facilitate modeling of complex network services. Second, we use *constraint propagation* in the constraint model as a natural way to report the consequences of observed inputs before a fault actually manifests itself in the real system. In [9] the diagnoses are produced *after* a fault is signaled, by finding all the constraint violations which are precise indicators of the deviations of the constraint model from the actual behavior of the diagnosed system. In this paper, besides search-based diagnosis, we consider also the consequences of observed inputs as these inputs are monitored (as events) or configured (as configuration settings), *before* the real system reaches an invalid state. In addition, constraint propagation supports explanation, by recording which observations produce certain propagation effects. Finally, we propose to implement this approach using the instrumentation capabilities of the IntelliWatch[1] Monitor tool.

The paper is organized as follows. In Section 2 we present a sample management problem, the replication service. We use this example of groupware services as a running example throughout the paper to describe our approach to fault management for groupware services. In Section 3 we introduce the *constraint satisfaction problem* (CSP) paradigm, highlight its strengths, and discuss its applicability to fault and configuration diagnosis. In Section 4 we present the composite CSP extension.

[1] IntelliWatch and IntelliWatch Monitor are trademarks or registered trademarks of Candle Corp.

Section 5 presents how constraint propagation can be used for proactive diagnosis and explanation. Section 6 reviews other model-based approaches to network management. In the last section we summarize the contribution of our work and discuss the proposed implementation support.

2. Sample Management Problem: Replication Service

Groupware services and applications allow widely dispersed groups of people to communicate and collaborate with each other on projects. One example is Lotus Notes[2] [11], a large, versatile, and complex groupware environment of shared, distributed, and email-enabled databases. *Replication* is one of its basic groupware services. Users working with different instances of a database are able to access and modify the same information in a manner such that each update is propagated to all users. In the Lotus Notes environment multiple instances of the same database residing on different servers and/or clients are called *replicas* and are identified by unique replica ID numbers.

Topology	Connection type: LAN (status, bandwidth), serial, etc.
	Source server
	Destination server
	Direction: one-way or two-way
Schedule	Days of the week
	Time limit
	Time: specific, list, or range
	Repeat interval
Data	Priority: low, medium, or high
	Files and Directories

Table 1: Elements of the Connection Document for scheduled replication

Our working example refers to *scheduled replication*, whose controlling features are partly summarized in Table 1. The configuration element central to setting up a scheduled replication is the *Connection Document*, through which the Lotus Notes administrator: (1) establishes the lines of communications among the participating servers according to some topology information, (2) schedules the times when replication events are triggered, and (3) decides what data get exchanged.

Among possible topologies, *hub-and-spoke topology* is the most recommended. In a hub-and-spoke topology, the hub server initiates replication and starts communicating with each spoke server in turn. A hierarchical organization of hub-and-spoke topologies is shown in Figure 1a, where H1 is both the hub of the A, B, and C spokes, and one of the spokes of the H2 hub. Connection documents created on a hub designate the hub as the source server and the spokes as destination servers. The topology information also specifies the connection type and the replication direction. The hub can either request reception of (pull) changes from the spoke (Figure 1b), or request transmission of (push) its own changes to the spokes, or do both. Replication occurs

[2]Lotus Notes is a trademark or registered trademark of Lotus Development Corporation.

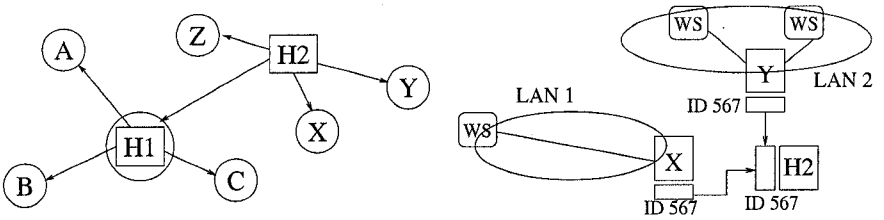


Figure 1: a) (left) Hub-and-spoke topologies for scheduled replication
b) (right) One-way, pull-only replication from spokes X and Y to hub H2

at scheduled times. In between these scheduled times, as users add, edit, or delete documents in a database, the replicas are not synchronized. Finally, another level of control of the replication process includes the specific databases and/or directories a user chooses to replicate.

3. Constraint Satisfaction

3.1 Framework

Constraint satisfaction is a well-known paradigm in the Artificial Intelligence community, and has been applied successfully for more than two decades to a variety of tasks, such as scheduling and resource-allocation, design and configuration, verification and diagnosis [12]. At the center of this paradigm lies the concept of the Constraint Satisfaction Problem (CSP). A CSP is expressed declaratively as a set of *variables* with associated domains of *values*, and a set of *constraints* on some of the variables, which restrict the values the constrained variables can take. Finding a solution to a CSP means to assign values to all variables such that all constraints are satisfied. A CSP solving technique is based on *searching* for such an assignment.

Modeling in CSP terms is straightforward. It simply requires the specification of the entities of interest to the problem being solved, or the CSP variables with associated values, and the interdependencies among these entities, or the CSP constraints.

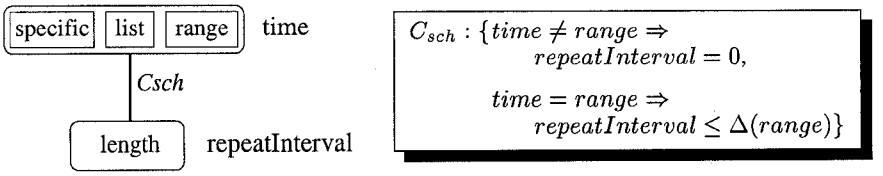


Figure 2: CSP description of scheduled replication elements

The CSP description in Figure 2 defines two variables corresponding to two elements in the Connection Document in Table 1, and expresses a constraint between them. Variable time can take one of three possible values, while variable repeatInterval can take a nonnegative integer. CSP values in general can be of type numeric, string, time, or interval and sets of these types. The constraint C_{sch} re-

stricts the value assignments to the two variables, and is written as a set of logical implications. Constraints return boolean values, thus they can be represented by any boolean function. For the C_{sch} constraint to be satisfied, at least one of the logical implications has to be found true. Auxiliary functions on the currently assigned value can be used in a constraint. The $\Delta(range)$ function in C_{sch} returns the span time of the range interval.

One essential strength of the CSP paradigm is that, on one hand, *constraint modeling* is the natural language of discourse for many applications, and, on the other hand, its declarative character totally decouples the resolution techniques from the design phase. Another advantage is the wealth of thoroughly examined *CSP algorithms*, from which the most effective for the problem at hand can be chosen. From this perspective, CSPs scale well: their profiles in terms of semantics of constraints and the number of variables and constraints can indicate the appropriateness of the algorithm to be used. Finally, constraint technology supports *constraint propagation*, or consistency inference, an efficient means of narrowing down the search space and finding partial solutions, that is, consistent value assignments to subsets of the CSP variables. Constraint propagation can be used prior to searching for solutions (e.g. arc or path consistency preprocessing) or can be embedded within the search algorithms (e.g. forward checking or maintaining arc-consistency algorithms [13]). Eliminating inconsistent values, either before or during search, can lead to a significant improvement in efficiency. Moreover, constraint propagation is a direct means for automatically producing all the consequences of a change occurring in the model.

3.2 Fault and Configuration Diagnosis

The CSP framework is model-based. To diagnose a system within the CSP framework, one must first represent the model of the system structure and correct behavior. Recent research has demonstrated the success of model-based approaches in the area of diagnosis [14]. In the constraint-based approach applied to diagnosing problems with network services [9], the structural elements of the system being diagnosed, or the MO descriptions, are modeled as CSP variables, and the behavioral relationships among the constituent components are expressed as constraints. Instantiating the model based on the observed values collected dynamically from the operational network provides us with possible deviations from the predicted, correct behavior of the model. The discrepancies are the violated constraints found by the CSP search algorithm, and they indicate the inconsistencies between the model predictions and system observations.

The solution in [9] relies on two extensions to the basic CSP framework. First, if a fault exists, the algorithm does not settle for "no solution found". Instead, it records as *diagnoses* the partial solutions for which some constraints are violated. In the example shown in Figure 2, one diagnosis can be indicated by the violation of the constraint C_{sch} when the observed values for the two variables do not satisfy the constraint. Second, the distributed nature of network services and applications, as well as dynamic changes in network configuration, imply that not all the MO

descriptions participate in diagnosing the observed problem. The active portions of the model that capture dependencies related exclusively to the observed symptoms and actual configuration can be isolated at run-time by the mechanism of activity variables and constraints. The example in [9] of diagnosing configuration problems with the Internet domain name service illustrates the use of a CSP framework that embeds partial solutions and dynamic control of model activation.

4. Composite Constraint Satisfaction

CSP modeling as presented above does not cope with complex and heterogenous management data through means such those employed in object-oriented approaches. Thus, there is no explicit support for a generic relationship that models *aggregation* or composition of subcomponents into larger components. Another essential modeling principle that controls complexity and enables reusability is *abstraction* or classification. Types of components that exhibit some commonality can be specialized or refined through is-a relationships, from a common, higher-level component type.

4.1 Composite Values

A natural way to extend the basic CSP to address these limitations is to allow values to be *composite values*, which are, in fact, entire constraint satisfaction subproblems. A composite CSP is defined in the same manner as a basic CSP [10]. The difference is that the values a variable can take are not restricted to a simple type. Instead, a value can stand for an entire CSP subproblem. Instantiating a variable to a composite value leads to dynamically modifying the original CSP with the new subproblem. While variable instantiation offers the mechanism of specialization, a composite value captures directly the concept of aggregation.

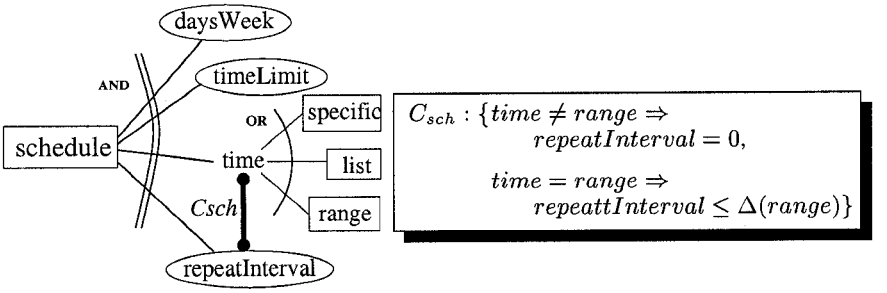


Figure 3: Variables and constraints of the schedule CSP model

In Figure 3 we show that the composite value schedule, corresponding to the schedule field in the connection document (Table 1), introduces the variables daysWeek, timeLimit, time, and repeatInterval. We distinguish the values from the variables by drawing values within rectangles, while variables are drawn within ovals; both are labeled with their names.

The graphical representation of the CSP submodel for the replication schedule shown in Figure 3 highlights the object oriented design principles of *composition* and *abstraction* incorporated in the composite CSP approach. Since the direct descendents of a composite value are all CSP variables which extend the current model, we use the graphical convention of an **AND** double arc. For abstraction, on the other hand, we use the **OR** single arc to show that a variable can be instantiated to one of its values only. The variable time can be instantiated to either specific, list, or range. Note that at the bottom of the abstraction hierarchy lie variables that take elementary values. Both timeLimit and repeatInterval have integer values, while daysWeek takes sets of string values. How is this CSP model utilized by the CSP search algorithm? When the composite value schedule becomes part of a CSP model at run-time, its variables and constraints are dynamically added to the current model.

4.2 Local Models and Ports

Two important additions to the modeling capabilities of the basic CSP are introduced by the composite CSP: the concept of a *local model*, and a mechanism for defining *types of relationships*. A composite value defines only constraints among the variables it contains. Therefore it represents a context-independent *local model*. A subset of the variables of a local model are made visible through the model interface. Whenever a particular instance of a local model is used in a specific context, additional constrains can be imposed on the interface variables as required by the context.

The schedule CSP model which contains only the constraints among schedule variables is a local or context-free model. The context in which this local model is used does not affect the model's structure and behavior. However, when "plugged" into different larger models, schedule can introduce different constraints between its interface variables and the variables of the model that encompasses it. For example, the schedule model can be used by both the replication service and the mail routing service. In either case it preserves its local variables and constraints, but it instantiates different contextual information, that is, constraints, based on the model's usage. In the case of replication for example, we define the C_{sch_repl} constraint to link schedule elements to topology and database information. The C_{sch_repl} constraint is defined on four variables: bandwidth, connTime, timeLimit, and sizeAllReplicas:

$$C_{sch\_repl} \ : \ sizeAllReplicas/bandwidth + connTime \leq timeLmit$$

The sizeAllReplicas variable belongs to the sourceServer composite value, while bandwidth and connTime characterizes connectionType. The constraint C_{sch_repl} ensures that the transfer of all replicas on the source server meets the schedule requirements while using the configured connection. In Figure 4 we show how the C_{sch_repl} constraint models the interaction among topology and schedule elements. We also provide a more detailed CSP model of the scheduled replication service represented by the scheduledRepl composite value. This value expands

to the CSP model that contains the variables: connectionType, sourceServer, destServer, direction, filesDirectories, priority, and the composite value schedule. To avoid cluttering the picture, we marked only the **AND** edges; all the others are assumed to be **OR**. Also, to keep the example simple, some of the bottom nodes in the hierarchy, such as sourceServer or serial, are not expanded to their CSP local models.

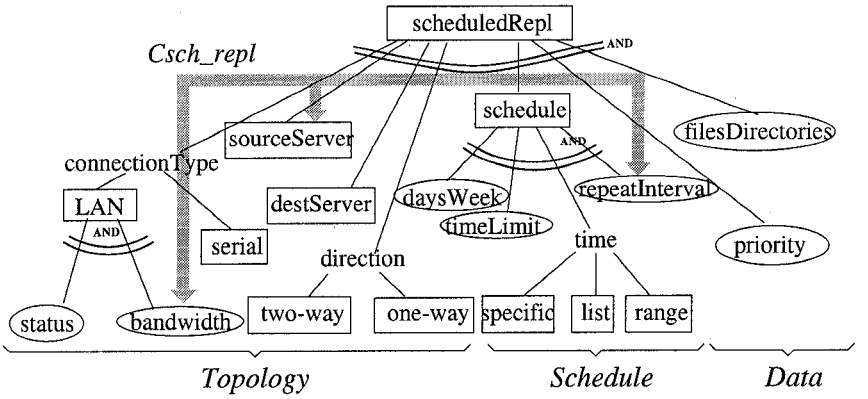


Figure 4: CSP model of scheduled replication

Types of relationships can be defined in the composite CSP approach through a special type of variable, called *port*. In general, a port is described by: (1) the type of values (possibly composite) which are allowed to connect to the port, (2) the cardinality of the port, and (3) additional constraints that filter further the values received by the port. A complete description of this modeling concept and its application within a constraint-based framework can be found in [10].

5. Constraint Propagation

Constraint-based fault management provides the set of minimal diagnoses by systematically instantiating variables to values according to the CSP model and the current observations collected from the running network. The constraint violations found in the search process correlate the triggered events and indicate the possible causes [9]. This model-based method has the important advantage of reporting unexpected faults without necessitating time for acquiring the empirical associations, known as rules, between symptoms and causes. In a domain so dynamic and heterogenous as computer networks, building such expertise has been proven to be a severe limitation to the traditional rule-based management systems.

Although diagnosing network services is essential to fault management, aspects such as proactive diagnosis and explanation considerably improve the quality of fault management. The method proposed in [9] addresses only diagnosis: in response

to a network service error, an automatically generated constraint-based diagnostician produces the set of minimal diagnoses based on observations, in a batch process. What we propose in this paper is to enhance the diagnostician capabilities with proactive diagnosis and explanation. This task calls for integration of the constraint propagation techniques with the basic search engine.

Constraint propagation allows for narrowing down the value domains by removing inconsistent values without ruling out any solution in a CSP. In this section we discuss how to exploit the facility of propagating information in a constraint model to enable proactive and explanatory support for diagnosis.

5.1 Proactive Diagnosis

The idea behind constraint-based proactiveness is simple. The constraint model of the monitored network system mirrors the current state of the system: changes in the real system translate into changes in the CSP model. Constraint propagation produces in the model the consequences of observed inputs. If the model reaches an invalid state, that is, it becomes inconsistent and no value assignments are possible for some variables to satisfy their constraints, it means that a fault is discovered in the model before the fault is actually signaled.

We consider again the very simple example of the four-variable constraint C_{sch_repl} defined previously. The constraint requires that the replication time (transfer and connection overhead time) be no larger than timeLimit, the configurable parameter in the connection document. One possible scenario could be that the connection time is within normal limits, i.e., does not exceed the threshold value of connTime for the configured bandwidth value, but there is more data than can be transferred at the given bandwidth within the remaining time. If this happens, timeLimit will stop the replication process without reporting any replication error in the server's log. Since replHistory is not updated, we need to introduce the constraint:

$$C(replHistory, endReplTime) : \ replHistory = endReplTime$$

so that violation of this constraint will indicate that replication did not have enough time to complete.

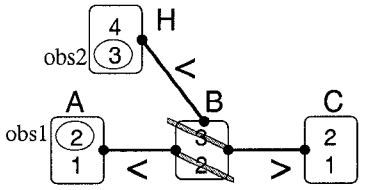
It would be more useful to give a warning *prior* to stopping replication, to allow for some corrective action. This is possible if the values of the three variables: bandwidth, connTime, and timeLimit are propagated according to the C_{sch_repl} constraint as soon as they are known from the system, usually at configuration time. The constraint propagation will set up an upper bound value for sizeAllReplicas. Dynamically monitoring a larger value for this variable during system operation would render the four-variable submodel inconsistent, and the constraint violation could be reported before the replication process fails to complete. This example is limited to a single constraint, but the same scheme is applied when an inconsistency is found as a result of propagating through several constraints, and of advancing

further in the model with predicted values of correct behavior. At any point in the process, a contradiction between observed values and propagated values may arise before the real system reaches the inconsistent state.

Another form of proactive diagnosis is the validation of configuration settings or changes. "What-if" scenarios are possible because the model responds to the configuration decisions, and reports incorrect configurations before their effects impact the functioning of the network service.

5.2 Explanation

In this section we show a simple example of how constraint propagation is used for explanation. Let us assume that in a hub-and-spoke topology, where the hub server H initiates replication with the spoke servers A, B, and C, the access control levels (ACLs) of the same replica, residing on all servers, should be set up according to the following policy. Replica on server H can receive from A and C either new documents (`Author` access), or both new documents and updates to the other documents (`Editor` access). From server B, it can receive not only document updates, but also changes to the database design elements (`Designer` access). H is authorized, according to the ACL information of the replicas on spokes, to send all the types of updates mentioned so far, as well as changes to the replica ACL (`Manager` access). The access policy also imposes the following binary constraints: B is allowed higher access than both A and C, and H is allowed higher access than B.



Obs.	Effect of propagation	through Constraint
$A = 2$	$B \neq 2$	$C_{AB} : A < B$
$H = 3$	$B \neq 3$	$C_{BH} : H < B$

Figure 5: Constraint propagation supports explanation

In Figure 5 we illustrate the allowed domains of values permitted for each server, and the three binary constraints represented as inequality constraints: C_{AB}, C_{BC}, and C_{BH}. For example, C_{BH} says that a valid value assignment to B and H satisfies the constraint that the value at B is smaller than the value at H. We assume that the replica ACLs: `Manager`, `Designer`, `Editor`, and `Author` can be represented as numerical values 4,3,2, and 1, respectively.

If it is observed first that `Editor` access is allowed to A, that is, $A = 2$, the effect of propagating this value instantiation through the constraint C_{AB} is to eliminate value `Editor` from B ($B \neq 2$). This happens because $B = 2$ would violate $C_{AB} : A < B$ when $A = 2$ ($2 < 2$ is false). When the second observation is made and `Designer` access is granted to H, that is, $H = 3$, C_{BH} propagates this input to B and, as a consequence, value 3 is eliminated from the domain of B ($B \neq 3$). Since no possible value at B is left to satisfy both C_{AB} and C_{BH} in the presence of the two observations, the diagnosis tool reports the fault. Note that, using propagation,

the fault is reported before an actual value is assigned to B. Moreover, a better explanation can be built, rather than reporting one constraint violation. The explanation takes into account how the two observations correlate in rendering any instantiation at B inconsistent. Each eliminated value at B records the observation that causes this propagation effect. When no value is left for B, an explanation is produced from concatenating the reasons for all value eliminations: "neither `Designer` nor `Editor` will work for B because A has `Editor` and H has `Designer` access". This way, the administrator correlates all the factors contributing to the diagnosed inconsistency.

6. Related Work

Several other model-based approaches have focused on modeling interobject dependencies and have developed appropriate reasoning techniques to perform event correlation or fault and configuration management of distributed services and applications. We present here some of these approaches and emphasize the key concepts in their solutions: active configuration and alarm knowledge, relation exploration, model instantiation and expansion, service graph of functional and environmental dependencies. More important, we observe that the advances of knowledge-based systems in the domain of integrated network management has led to the crystallization of a recurring theme: expressive models which dynamically map to, mirror, and respond to the managed system. Once these models have been constructed, they are used as the basis of on-line tools that monitor operating networks to detect and diagnose faults as they occur.

The solution to event correlation and fault isolation problems presented in [4] and [5] is based on modeling network resources as objects in a generic object model, and on developing relationships between them. The relationships model links across layers of the protocol stack, and between connections at the same layer. To correlate the network objects reporting fault events with the same root cause, the relationships are used to "navigate" the model and build an equivalence relation on the network objects. Similarly, the constraints in our approach are used to propagate the observations in the model. Model navigation in [4] can be obtained by propagating fault events along equality constraints that would model the specified interobject relationships. To isolate the root cause of those faults that belong to the same equivalence class, [4] apply heuristic rules to navigate the causality relationships between objects within each class.

Management of distributed services and applications poses additional challenges with regard to what to represent and how to reason on the designed representation. In the absence of standard models at the service layer, management systems rely on complex, hierarchical structure of objects representing components and dependencies among them [5]. Two effective frameworks are the work presented in [7] for fault and configuration diagnosis of distributed applications, and the work in [8] for service availability.

Based on the idea that to manage a system one should construct a description of

it, [7] propose Dolphin, an object-oriented modeling language for networked systems and distributed applications management, in which the modeler provides a precise description of the correct functioning of the system. Objects described in Dolphin include fundamental components of the system to be managed. They are characterized by basic attributes, or elementary information about the objects' properties, and by derived attributes, also called rules, which give higher-level information about the object behavior. The rules are in fact constraints, and the diagnostic task performs constraint checks as observed values instantiate the constraints.

The modeling schema proposed in [8] for the availability task of distributed applications starts with a generic service graph of the functional dependencies among the application's underlying services. This graph is then gradually refined into a more comprehensive availability graph that involves components and their associated functional and environmental dependencies. The same function is obtained in our constraint-based approach through composite values, which bring in more details about the management information as needed. Finally, the parameterized availability graph is instantiated in the concrete environment in which the availability tests and calculations of the involved components are performed.

7. Conclusion and Future Work

In this paper we have introduced a constraint-based approach to fault management, and illustrated its application to the replication service of the Lotus Notes system. We have shown that our approach provides a flexible and powerful framework for modeling network services by aggregating context-independent models. Based on the model and the observations collected through IntelliWatch monitoring tools, the inference engine performs both reactive and proactive diagnosis, and provides explanation capabilities.

A prototype based on our approach is currently under development. At the heart of the system lies a library of C++ classes for representing variables and constraints. The domains of variables can be both discrete and continuous. The library offers a large variety of predefined constraint operators, such as arithmetic, logical, relational, set membership, etc., and specialized constraints. In addition, the modeler can create new types of constraints, expressed either extensionally, as a set of tuples of allowed/disallowed values, or intentionally, by combination and derivation from predefined constraint classes. The search engine uses one of the most performant general-purpose CSP search algorithms [13]. While the minimum level of consistency maintained by the prototype system is arc-consistency, more powerful forms of inference are performed for the specialized constraints.

In order to perform its tasks of reactive and proactive diagnosis and explanation, the system needs both the constraint-based model and some inventory tools which supply observations about the replication service. As inventory tools we use the IntelliWatch Monitor, a monitoring and management tool for Lotus Notes systems.

Acknowledgment

This material is based upon work supported in part by Candle Corporation, National Science Foundation under Grant No. IRI-9504316, and Trilogy. The paper has benefited from many discussions of the members of the Constraint Computation Center at UNH, and from the insightful comments of the reviewers.

References

[1] E. Nataf, O. Festor, and L. Andrey, "RelMan: A GRM-based relationship manager," in *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 661–672, 1997.

[2] R. Eberhardt, S. Mazziotta, and D. Sidou, "Design and testing of information models in a virtual environment," in *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 461–472, 1997.

[3] B. Baer and A. Clemm, "Testing of relationships in an osi management information base," in *Proceedings of the Forth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 578–591, 1995.

[4] J. Jordaan and M. Paterok, "Event correlation in heterogeneous networks using the osi management framework," in *Proceedings of the Third IFIP/IEEE International Symposium on Integrated Network Management*, pp. 683–695, 1993.

[5] S. Katker and M. Paterok, "Fault isolation and event correlation for integrated fault management," in *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 583–596, 1997.

[6] D. Ohsie, A. Mayer, S. Klinger, and S. Yemini, "Event modeling with the MODEL language," in *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 625–637, 1997.

[7] A. Pell, K. Eshgi, J.-J. Moreau, and S. Towers, "Managing in a distributed world," in *Proceedings of the Fourth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 94–105, 1995.

[8] G. Dreo Rodosek and T. Kaiser, "Determining the availability of distributed applications," in *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 207–218, 1997.

[9] M. Sabin, R. Russell, and E. Freuder, "Generating diagnostic tools for network fault management," in *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management*, pp. 700–711, 1997.

[10] D. Sabin and E. Freuder, "Composite constraint satisfaction for configuration," in *Working Notes of the AAAI'96 Fall Symposium* (E. Freuder, ed.), 1996.

[11] Lotus Notes, *Getting Started with the Domino Server*, 1997.

[12] E. Freuder and A. Mackworth, eds., *Constraint-Based Reasoning (Special Issue of Artificial Intelligence: An International Journal*. MIT Press, 1994.

[13] D. Sabin and E. Freuder, "Understanding and improving the mac algorithm," in *Proceedings of the Third International Conference on Principles and Practice of Constraint Programming*, pp. 167–181, 1997.

[14] W. Hamscher, L. Console, and J. de Kleer, eds., *Readings in Model-Based Diagnosis*, (San Mateo, CA), Morgan Kaufmann Publishers, 1992.

Biography

Mihaela Sabin is currently pursuing her Ph.D. degree at the Department of Computer Science, University of New Hampshire. Her research interests include constraint satisfaction, modeling, configuration, and network management. Her home page is `http://www.cs.unh.edu/~mcs`.

Alex Bakman is vice president of Candle Corp. He is recognized as one of the leading experts in Notes systems management and is a member of Lotus Technical Advisory Group. Alex has authored many articles on Notes systems management, Candle's Lotus Notes administrator survival guides and a book called "How to Deliver Client/Server Applications that Work" published by Prentice Hall.

Eugene C. Freuder is a professor in the University of New Hampshire Department of Computer Science and Director of its Constraint Computation Center. A Fellow of the American Association of Artificial Intelligence he is the founding editor-in-chief of *Constraints*, An International Journal (Kluwer Academic Publishers) and the executive chair of the Organizing Committee of the International Conference on Principles and Practice of Constraint Programming. His home page address is `http://www.cs.unh.edu/Personal/ecf.html`.

Robert D. Russell is an associate professor in the University of New Hampshire Department of Computer Science. His research interests include network protocol development, and network management. He is a member of IEEE and ACM. His home page is `http://www.cs.unh.edu/~rdr`.