An Investigation on Component Based Adaptation for Feature Models

R.Saradha[1] Dr. X. Mary Jesintha[2]
[1]Research Scholar [2]Professor
[1,2]Department of MCA
[1]Saradha Gangadharan College, Pondicherry, India
[2]Vivekananda Institute of Engineering, Thiruchengode, India

Abstract— To support media applications in portable situations, it will be important for applications to know about the hidden natural conditions, furthermore to have the capacity to adjust their conduct and that of the real stage all things considered conditions change. Numerous current scattered frameworks stages backing such adjustment just in a fairly irregular way. This paper presents a principled approach to supporting adaptation through the use of reflection. More specifically, the paper introduces a language independent, component-based reflective architecture featuring a per-component meta-space, the use of meta-models to structure meta-space, and a consistent use of component graphs to represent composite components. The paper additionally gives an account of a nature of administration stage, giving better backing than checking and adjustment capacities. At long last, the paper clarifies a model execution of the design utilizing the article arranged programming dialect Python

Keywords: component-based, specifically, adaptation, object-oriented, reflection, environments, prototype

I. INTRODUCTION

In heterogeneous circulated situations, the other vital element is openness. Specifically, it is critical that open frameworks guidelines are xploited wherever conceivable to allow both interoperability and transportability of (portable) applications. In this paper, we are along these lines worried with backing for adjustment in open disseminated frameworks. We show a principled answer for this issue taking into account thoughts from the fields of open usage and reflection. We are especially eager about supporting circulated sight and sound applications in such situations then expand this engineering with nature of administration (QoS) administration abilities. Our usage of the intelligent engineering (counting QoS administration) is then quickly examined.

A. Basics of Adaptation:

Adjustment is required at different levels in a conveyed framework. For instance, at the system level, it may be important to switch between accessible systems joins relying upon nature of administration at present being advertised. Greater up the convention bundle a scope of adjustments are likewise conceivable, for example, changing to an alternate transport convention, changing the parameters of a specific convention component, or bringing out new convention components. It might likewise be helpful to change the encoding of information, e. g. by presenting pressure and de-pressure components. Significantly, numerous parts of adjustment are additionally done either in the application or with the inclusion of the application. For instance, a project may rebuild its movement by off-stacking some preparing to a settled part of the system. All the more for the most part, adjustment is material to a variety of perspectives at various levels of the framework. These angles incorporate the interchanges perspectives clarified above additionally incorporate issues like the assets distributed to a movement and the arrangement of outside administrations getting utilized. Besides, this adjustment ought to be driven by familiarity with an assortment of issues including showcasing and deals correspondences execution, asset use, area, cost, and battery life and application inclination.

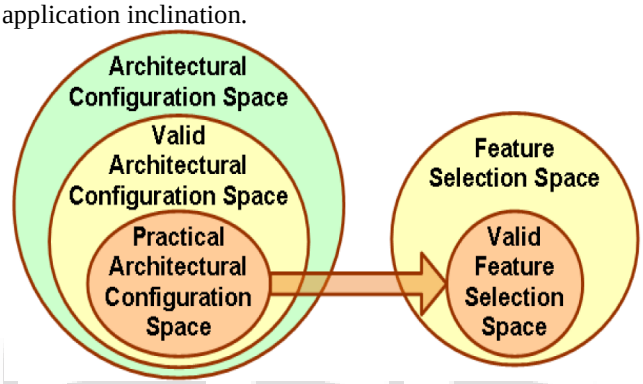

Fig. 1: Self Adaptive Software System

B. Adapting methods:

Adjustment is required at different levels in a conveyed out framework. For instance, at the system level, it may be important to switch between accessible systems joins relying upon nature of administration getting advertised. Greater up the convention bundle a scope of adjustments are additionally conceivable, for example, transitioning to another vehicle convention (or gathering of smaller scale conventions [14]), changing the rules of the specific convention part, or bringing out new convention components. It might likewise be valuable to supplant the encoding of information, e. g. by bringing out pressure and de-pressure components. Vitally, viewpoints worth considering of variety are likewise done either in the application structure or with the inclusion of the application. For instance, a project may rebuild its action by off-stacking some control to an altered segment of the system. Much all the more for the most part, adjustment can be connected to a variety of viewpoints at various levels of the framework. These perspectives incorporate the interchanges angles clarified above yet incorporate issues like the assets dispensed to an action and the arrangement of outside administrations utilized. Besides, this adjustment ought to be driven by comprehension of an assortment of issues including showcasing and deals correspondences execution, asset utilization, area, cost, and battery life and application inclination.

C. Adapting Frameworks:

Since specified above, adjustment is important at all levels of a framework, from the application level conceivably directly down to the working framework level. However , this promptly shows various issues. To get case, adjustment at the working framework level can be entirely perilous in states of influencing respectability and fulfillment. Moreover, the software engineer would definitely need to depend on working framework specifics to achieve variety, in this manner trading off the transportability of utilizations. The option compelling of leaving all adjustment to the application is additionally evidently unsatisfactory, as this would present an unsuitable weight for the product essayist. Our answer is to present a stage for overseeing adjustment at the middleware level of the framework.

In allotted frameworks, middleware is depicted as a layer of programming dwelling in every machine and sitting between root (heterogeneous) working framework stages and appropriated applications/administrations, offering a stage free programming model to software engineers.

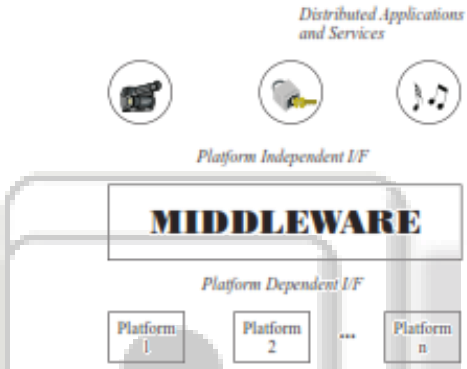

Fig. 2: The Role of Middleware.

Know that this methodology is bolstered by late research in the working frameworks group which advocates that working frameworks ought to be littler and more arrangement free [12].

This infers bunches of capacities at initially contained in the

II. Literature Review

There is developing enthusiasm for the utilization of reflection in circulated frameworks. Spearheading work here was done by McAffer et al. As for middleware, specialists at Illinois have completed introductory trials on reflection in Article Ask for Merchants. The level of reflection however is coarse-grained and limited to conjuring, marshaling and dispatching. Also, the work does not consider key territories, for example, support for gatherings or, all the more for the most part, ties. Scientists at APM have built up an exploratory middleware stage called FlexiNet et al . This stage permits the software engineer to tailor the fundamental correspondences foundation by embeddings/evacuating layers. Their answer is, be that as it may, language specific, i.e. applications must be composed in Java. Manola has completed work in the outline of a "RISC" object model for dispersed registering , i.e. a negligible item demonstrate which can be specific through reflection. Analysts at the Ecole des Mines de Nante are additionally exploring the utilization of reflection in intermediary instruments for Circles.

Our configuration has been influenced by various specific reflective dialects. As expressed over, the idea of multimodels was gotten from AL/1-D. The basic models of AL/1-D are however entirely distinctive; the dialect underpins six models, specifically operation, asset, measurements, movement, conveyed environment and framework.

From this rundown, it can be seen that AL/1-D does however bolster an assets model. This asset model backings reification of planning and trash accumulation of articles (however in a moderately restricted manner contrasted with our methodology).

Our progressing research on nature and embodiment meta-models is likewise intensely influenced by the outlines of ABCL/R and Cod A . Both these frameworks highlight deteriorations of meta-space as far as the acknowledgment of messages, setting the message in a line, their determination, and the resulting dispatching. At long last, the outline of ABCL/R2 incorporates the idea of gatherings. Be that as it may, bunches in ABCL/R2 are more prescriptive in that they uphold a specific development and translation on an item.

The gatherings themselves are likewise primitive, and are not helpless to reflective access. Our utilization of segment charts is propelled by specialists at JAIST in Japan In their framework, adjustment is taken care of using control scripts written in TCL. Despite the fact that like our proposition, the JAIST work does not give access to the inside points of interest of correspondence items.

Moreover, the work is not incorporated into a middleware stage. Comparable methodologies are pushed by the originators of the VuSystem [22] and Pound . The same reactions however additionally apply to these plans. Microsoft's ActiveX programming additionally utilizes segment charts. This product, in any case, does not address conveyance of part diagrams. Also, the diagram is not re-configurable at runtime.

At long last, various specialists have considered the effect of rising application ranges on middleware. For instance, various middleware stages have been created to bolster sight and sound . Most prominently, specialists at CNET have built up a developed CORBA stage to bolster mixed media; this stage includes the idea of recursive ties and has been profoundly influential in our exploration, a related stage is likewise reported in . Concentrates on have likewise been done in the territories of portability, constant and gathering support . While a considerable lot of these plans bolster a level of configurability, they don't offer the level of openness and adaptivity that we look for in our exploration.

III. Feature Models for Runtime Adaptation

A feature model is a compact representation of all possible products or configurations, for instance, of an SPL. These models are visually represented as features and relationships among them. Features correspond to selectable concepts of the system at any abstraction level: functional and non-functional (or quality-attribute) requirements, environment and context restrictions, runtime components, implementation modules, etc. A feature model is arranged in a hierarchy that forms a tree where features are connected by:

- Tree constraints: relationships between a parent feature and its child features (or sub-features). Tree constraints include mandatory, optional, xor (alternative) and or relationships between parents and sub-features.
- Cross–tree constraints: typically inclusion or exclusion statements of the form "if feature F is selected, then features A and B must also be selected (or deselected)".

The root feature of the tree represents the concept being described, generally the system itself, and the remaining nodes denote branches and sub-features that disaggregate the main concept into several elements and concerns. Several extensions to this basic notation have been proposed [5], including propositional formulas for cross-tree constraints [6], and extended or attributed feature model [7]. We use these two extensions for modeling system element dependencies and specifying quality-attribute properties in our work. Generic propositional

Formulas allow us to define constraints and dependencies among features, such as "features A and B imply not C".

The second extension refers to a type of feature model in which additional information is added as feature attributes. An Attribute consists of a name, a domain, and a value. Attributes are often used to specify extra information, such as cost, response time, or memory required to support the feature, among others.

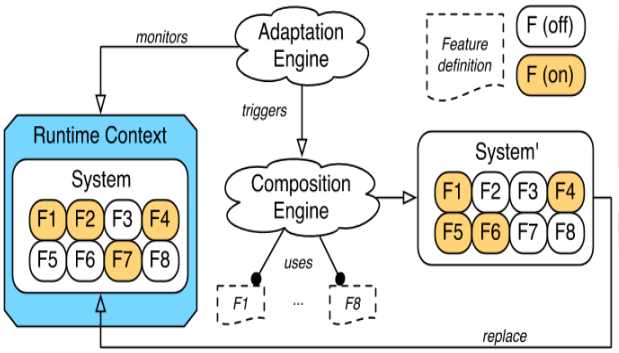

Fig. 3: A model to support software adaption

A. Formal Verification Methods:

State space (or reachability) analysis provides a promising and automated method for the static analysis and verification of systems. However because of the complexity of the state-space explosion, efficient analysis by state space is restricted to small system models. With increasing computing power, larger and more complex systems are continuously developed.

Techniques and tools need to be developed to analyze on large-scale systems. However, formal verification method is more applicable to architecture level of a software system since this level is closer to the problem domain and more abstract. Many techniques have been proposed to cope with the state explosion problem. Equivalence is the main concept shared in these techniques. Two system models are considered to be equivalent if no distinction can be made between them.

Based on the concept of equivalence, it is possible to analyze a system model using another equivalent system model or state space. For example, Petri-net models can be analyzed using reach-ability graphs because Petri nets and reachability graphs are equivalent with respect to almost all dynamic behaviors and interesting properties. The notion of equivalence is very useful when we focus on particular behaviors/properties of systems. The following summarizes existing approaches to the state explosion problem.

Transformation of System Models. Many transformation techniques have been developed from the concept of equivalence in order to synthesize and/or reduce system models [3], [4]. A synthesized system model is equivalent to the original one. Therefore, the verification of synthesized system models can be eliminated. On the contrary, structural reduction is used to generate a reduced system model which is equivalent to the original one. Consequently, the analysis can be performed on a reduced system model. Synthesis and structural reduction of system models are powerful, but in some cases they are applicable only to special situations or special subclasses of specification formalisms.

Bit-Hash Technique. Holzmann [5] proposed a bit-hash technique to significantly reduce the memory requirement of state space analysis. In his technique, a hash function is employed to give a compact representation of state information. However, the analysis result may not be reliable because of hash collisions in some cases.

Symmetry Methods. Symmetry methods [6] exploit regularity of system structure and store mutually symmetric states as one. Symmetry methods require more complex analysis algorithms and cannot facilitate the analysis of nonsymmetrical systems. Symbolic Model Checking. Symbolic model checking [7], [8], and [9] uses a fixed-point algorithm over a symbolic representation of state and transition relation (BDDs [10]). Symbolic model checking has been shown useful especially for the verification of hardware. However, its application to software systems may be affected by the effort in finding a compact encoding.

Partial Order Techniques. Partial order techniques investigate only a partial ordering of concurrent and independent events. The explored state space is equivalent in terms of certain properties, e.g., deadlocks and event occurrences. Persistent (or stubborn) sets [11] and sleep sets [12] are two main techniques proposed in the literature. Nevertheless, in the worse case, the state space remains exponential in the size of processes. A lot of research is focusing on the design of heuristic rules to reduce analysis complexity.

Compositional Verification Techniques. In compositional verification techniques, a system is considered as a collection of subsystems (processes). The main goal of compositional verification techniques is to hierarchically and efficiently generate equivalent and much smaller state spaces, which are more amenable to analysis. There have been a considerable amount of studies on the compositional verification, especially in the area of process algebras [13], [14], [15]. However, current techniques are effective only for loosely coupled modules (i.e., subsystems with simple interfaces).

Incremental Verification Techniques. Many systems frequently undergo modifications to suite new user requirements or to satisfy the desired properties. Their specifications and hence, internal representations keep changing from time to time. However, due to the huge size of the state-space, it is very expensive to verify the Requirements and constraints of the system after every

small modification to the specification. The incremental approach is to make the cost of verification of the modified system proportional to the size of the change made to it rather than the size of the representation (the state-space) of the whole system [16], [17].

Hybrid Techniques. It has been suggested that more cost-effective techniques can be derived by integrating different verification techniques. For example, partial order techniques and structural reduction can be combined to further reduce analysis cost [18]; symbolic state space search is combined with partial-order reduction methods for invariant checking [19], etc.

B. Compositional Verification :

Among the formal verification techniques discussed above, compositional verification is considered more suitable for verifying component-based software systems or architecture specifications of a target system [20]. In this paper, we will focus on the important concepts and techniques in this approach. In compositional verification, a system is first divided into hierarchical modules (subsystems). State spaces of modules are then recursively constructed from reduced state spaces of submodules. As a reduced state space could be much smaller than the original one, the compositional verification technique provides an attractive approach for analyzing large-scale systems. Compositional verification techniques resemble structural-reduction techniques for system models [3]. Nevertheless, compositional verification techniques are applied to state spaces instead of system models. Compositional verification techniques are more effective than structural-reduction techniques since state spaces explicitly describe the dynamic behaviors of systems.

The main goal of compositional verification techniques is to hierarchically and efficiently generate equivalent and much smaller state spaces. However, the concept of equivalence does not consider compositional mechanisms. It is possible that the interesting properties of systems are not preserved after we replace one module with another equivalent module. In order to hierarchically generate an equivalent state space, the concept of congruence is essential. to the old one. Various compositional verification techniques have been proposed for the algebraic treatment of processes by using equations and inequalities for processes expressions. These approaches to the algebraic treatment of processes can be identified by an acceptance of synchronized communication as the primitive means of interaction among processes [21].

Primary process algebras include for example, Milner's CCS [21], Hoare's CSP [14], and Bergstra and Klop's ACP [15]. Equivalences and congruences of processes are well founded in those process algebras for Event-based systems.

IV. QUALITY ATTRIBUTES ON FEATURE MODELS

A quality attribute is a key aspect (or property) of the system that is used by its stakeholders to judge its operation, rather than specific functional behaviors [8]. Quality-attribute properties of a system are typically quantified by quality metrics, particularly those with runtime characteristics. Systems often fail to meet stakeholder's needs regarding these attributes when they focus on some aspects without considering the impact on others. For instance, when system adaptation is required, it might not be possible to select a configuration that minimizes the reconfiguration time while at the same time maximizes the overall quality of service (QoS) since for this we probably need to setup and tune additional components that impact negatively on the reconfiguration time. Furthermore, the overall QoS is defined based on a combination of conflicting runtime properties (e.g., response time, accuracy, availability, security).

For example, replicating communication and computation to achieve availability, or including a shadow removal component to achieve high accuracy for event recognition, might conflict with performance requirements (e.g., low response time) or resource restrictions (e.g., maximum memory consumption). Stakeholders generally find it difficult to quantify their preferences in such conflict situations.

The goal of our model-based approach for managing quality attributes is to quantitatively evaluate and trade-off multiple quality attributes to arrive at a better overall system configuration. We do not look for a single metric but rather for a quantification of individual attributes and for trade-offs among those metrics. The problem is formalized as the optimization of an objective function that aggregates the metrics and quantifies stakeholders' preferences for individual attributes. In our approach, the management of runtime quality attributes involves three steps, namely:

1) specification,
2) measurement, and
3) optimization.

Specification deals with the representation and assignment of quality-attribute properties of individual system elements to features. Measurement implies the design of metrics for feature models to assess this quality attributes at the system level. At last, optimization deals with the maximization (and/or minimization) of conflicting attributes, which are assigned to different weights in order to consider stakeholders' preferences, while still meeting configuration rules and resource restrictions.

A. Specification:

Quality-attribute properties must be specified at design time for system architects. A wide range of properties exists to evaluate the runtime operations of a system. Some attributes are common to most adaptive systems, like reconfiguration time, response time, memory consumption, availability, among others. Besides, video-surveillance systems exhibit specific attributes, namely: accuracy and sensitivity of detection or tracking algorithms, relevance of object classification, frame rate, among others. These properties can be categorized into the following classes, depending on how they are specified on the feature models:

1) Direct assigned attributes: this class contains attributes that are representable as features, because they can be directly selected by stakeholders during the product configuration phase at development time. At runtime, the selection and deselection of these features can be triggered by events coming from context changes. In our model, these features correspond to the Quality of Service (QoS) branch.

2) Quantitative attributes: this category contains feature attributes that can be measured on a metric scale, such as: response time, reconfiguration time, accuracy, among others. These attributes define properties of individual features, but one can infer a measure for the overall configuration using some metric function able to aggregate the values of individual elements. For example, the system memory consumption can be calculated as a sum of the required memory for each running component.

Qualitative attributes: this category includes attributes of features that can only be described qualitatively using an ordinal scale, i.e. a set of qualifier tags like low, medium, and high for usability, security, or camera resolution, among others. In this case, there is no metric for deriving quantifiable measures of the overall configuration. However, a mapping function from qualifier tags onto real values can be used in order to handle these attributes as quantitative properties.

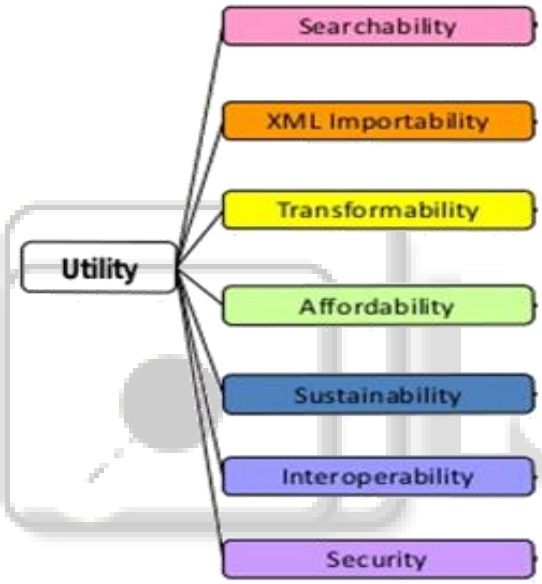

Fig. 4: Common Quality Attributes

Quality Attribute Examples
Required memory reconfiguration and response time (sequential execution)
accuracy, availability
reconfiguration and response time (parallel execution)
Security, usability (using a metric scale for qualifier tags)

Table 1: Quality Attributes used

V. Conclusions

This paper has addressed the issue of distributed systems support for mobile applications. It is now well recognized that this equates to support for adaptation. However, in our opinion, many current solutions address adaptation in a rather ad hoc manner; we believe a more principled approach is required in terms of open access to underlying systems components, and QoS management of such components. We also believe that middleware is the right place to place such functionality. The paper has presented the design and adaptation of a reflective middleware architecture that, we argue, provides such principled support for mobile applications. The architecture is supported by a component framework featuring an open and extensible set of primitive and composite components. In addition, our approach to QoS management has a number of interesting properties in terms of I) offering support for the dynamic insertion and adaptation of QoS management components, ii) allowing policies to operate over all meta-models (including crucially the resource meta-model), and iii) enabling the creation of sophisticated management structures featuring, for example, policies and meta-policies. address issues of performance in reflective middleware platforms.

References

[1] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson, "Feature-oriented domain analysis (food) feasibility study," Software Engineering Institute, Tech. Rep. CMU/SEI-90-TR-21, Nov. 1990.

[2] S. Hallsteinsen, M. Hinchey, S. Park, and K. Schmid, "Dynamic software product lines," Computer, vol. 41, no. 4, pp. 93–95, April 2008.

[3] S. Moisan, J. Rigault, M. Acher, P. Collet, and P. Lahire, "Run time adaptation of video-surveillance systems: A software modeling approach," in Computer Vision Systems, ser. Lecture Notes in Computer Science, J. Crowley, B. Draper, and M. Thonnat, Eds. Springer Berlin Heidelberg, 2011, vol. 6962, pp. 203–212. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23968-7_21

[4] L. Sanchez, S. Moisan, and J.-P. Rigault, "Metrics on feature models to optimize configuration adaptation at run time," in Combining Modelling and Search-Based Software Engineering (CMSBSE), 2013 1st International Workshop on, May 2013, pp. 39–44.

[5] P. Schobbens, P. Heymans, J. Trigaux, and Y. Bontemps, "Generic semantics of feature diagrams," Computer Networks, vol. 51, no. 2, pp. 456 – 479, 2007, feature Interaction. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128606002179

[6] D. Batory, "Feature models, grammars, and propositional formulas," in Proceedings of the 9th International Conference on Software Product Lines, ser. SPLC'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 7–20. [Online]. Available: http://dx.doi.org/10.1007/11554844_3

[7] D. Benavides, P. Trinidad, and A. Ruiz-Cortés, "Automated reasoning on feature models," in Advanced Information Systems Engineering, ser. Lecture Notes in Computer Science, O. Pastor and J. Falcão e Cunha,Eds. Springer Berlin Heidelberg, 2005, vol. 3520, pp. 491–503.[Online]. Available: http://dx.doi.org/10.1007/11431855_34

[8] L. Bass, Software architecture in practice, 2nd ed. AddisonWesley, Apr. 2003. [Online]. Available: http://www.worldcat.org/isbn/0321154959

[9] F. Rosenberg, P. Celikovic, A. Michlmayr, P. Leitner, and S. Dustdar, "An end-to-end approach for qos-aware service composition," in Enterpris Distributed Object Computing Conference, 2009. EDOC '09. IEEEInternational, Sept 2009, pp. 151–160.

[10] R. Marler and J. Arora, "The weighted sum method for multiobjective optimization: new insights," Structural

and Multidisciplinar Optimization, vol. 41, no. 6, pp. 853–862, 2010. [Online]. Available: http://dx.doi.org/10.1007/s00158-009-0460-7

[11] P. Guerra, C. Rubira, and R. de Lemos. An idealized faulttolerant architectural component. In R. de Lemos, C. Gacek,and A. Romanovsky, editors, Proceedings of the Workshop on Architecting Dependable Systems, ICSE'2002, Orlando,FL., May 2002.

[12] P. Guerra, C. Rubira, A. Romanovsky, and R. de Lemos. Integrating COTS software componentes into dependable software architectures. In Proceedings of the 6th International Symposium on Object-Oriented Real-Time Distributed Com- puting (ISORC'03)., pages 139–142, 2003.

[13] V. Issarny and J. P. Banatre. Architecture-based exception handling. In Proceedings of the 34th Annual Hawaii Inter-national Conference on System Sciences. IEEE, 2001.

[14] M. Hayden. The Ensemble System. PhD thesis, Dept. of Computer Science, Cornell University, USA, 1997.

[15] R. Hayton. FlexiNet Open ORB Framework. Technical Report2047.01.00, APM Ltd., Poseidon House, Castle Park, Cambridge, CB3 ORD, UK, October 1997.

[16] A. Hokimoto and T. Nakajima. An Approach for Constructing Mobile Applications using Service Proxies. In 16th International Conference on Distributed Computing Systems(ICDCS'96), May 1996.

[17] Interactive Multimedia Association. Multimedia System Services - Part 1: Functional Specification (2nd Draft). IMA Recommended Practice, September 1994.

[18] Interactive Multimedia Association. Multimedia System Services - Part 2: Multimedia Devices and Formats (2ndDraft). IMA Recommended Practice, September 1994.

[19] R. Katz. Adaptation and Mobility in Wireless Information Systems. IEEE Personal Communications, 1(1):6–17, Quarter 1994.

[20] G. Kiczales, J. des Rivi`eres, and D. Bobrow. The Art of Meta Object Protocol. MIT Press, 1991.

[21] T. Ledoux. Implementing Proxy Objects in a Reflective ORB. In ECOOP'97 Workshop on CORBA: Implementation, Use and Evaluation,Jyv¨askyl¨a, Finland, June 1997.

[22] C. Lindblad and D. Tennenhouse. The VuSystem: A Programming System for Computer-Intensive Multimedia.IEEE Journal of Selected Areas in Communications,14(7):1298–1313, 1996.

[23] E. Madelaine and R. de Simone. FC2: Reference Manual Version 1.1., 1994. See http://www.inria.fr/ meije/verification/doc.html.

[24] S. Maffeis and D. Schmidt. Constructing Reliable Distributed Communication Systems with CORBA. IEEE Communications Magazine, 14(2), February 1997.

[25] F. Manola. MetaObject Protocol Concepts for a "RISC" Object Model. Technical Report TR-0244-12-93-165, GTE Laboratories, 40 Sylvan Road, Waltham, MA 02254, USA, December 1993.

[26] S. Matsuoka, T. Watanabe, and A. Yonezawa. Hybrid Group Reflective Architecture for Object-Oriented Concurrent Reflective Programming. In European Conference on ObjectOriented Programming (ECOOP'91), LNCS 512, pages 231–250, Geneva, Switzerland, 1991. Springer-Verlag.

[27] J. McAffer. Meta-Level Architecture Support for Distributed Objects. In G. Kiczales, editor, Reflection 96, pages 39–62, San Francisco, 1996. Available from Dept of Information Science, Tokyo University, 1996.

[28] S. McCanne, E. Brewer, R. Katz, L. Rowe, E. Amir, Y. Chawathe, A. Coopersmith, K. Mayer-Patel, S. Raman, Schuett, D. Simpson, A. Swan, T.-K. Tung, and D. Wu. Toward a Common Infrastructure for MultimediaNetworking Middleware. In 7th Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '97), St. Louis, Missouri, May 1997.

[29] Object Management Group. AV Strems RTF, 1998. Available from http://www.omg.org/library/ schedule/AV_Streams_RTF.htm.

