Semantic Foundations for Survivable System Analysis and Design

Richard Linger
CERT Coord. Center
Software Engrg. Inst.
Carnegie Mellon Univ.
Pittsburgh, PA

Alan Hevner
Information Systems &
Decision Sciences Dept.
Univ. of South Florida
Tampa, FL

Gwendolyn Walton
Electrical & Computer
Engineering Dept.
Univ. of Central Florida
Orlando, FL

Ann Sobel
Computer Science &
Systems Analysis Dept.
Miami University
Oxford, OH

A Survivability Research Agenda

Survivability is the capability of an information system to support critical enterprise missions in adverse environments of attacks, failures, and accidents [Ellison et al 1999]. A research program in survivability must therefore address both systems and the environments within which they operate. Survivability is a combination of quality attributes, including security, reliability, safety, fault tolerance, dependability, and others [Mead et al 2000]. The SEI CERT Coordination Center, in cooperation with other researchers, has embarked on a multi-year, dual-thread research program, one thread to create engineering practices for survivable system design and development, the other to create engineering practices for analysis and definition of adverse environments. We believe that lack of theoretical foundations in both areas has been a serious impediment to survivable system development. In essence, we seek to move beyond natural language descriptions of survivability to a computational capability for engineering analysis of survivability properties. Accordingly, our agenda is to progress from theoretical foundations, to formal language representations, to engineering practices. We take it as an article of faith that to be effective, engineering practices must be based on rigorous foundations.

At the same time, it is important to target foundations and engineering practices to the present reality and future evolution of information system architectures and technologies. Today's large-scale infrastructure systems are characterized by the behavioral complexity of asynchronous operations and the semantic complexity of data and service interoperability. Loss of intellectual control is a common occurrence in development and operation of such systems, whose complexity often exceeds current engineering capabilities for abstraction and analysis. In the absence of engineering methods that scale to systems of such magnitude, it is difficult to assess and improve survivability properties. In the future the trend to very-large-scale systems-of-systems integration will reach global proportions, enabled by new standards and technologies. This trend will establish an unbounded global computation and communication grid far more capable than present-day networks, a grid populated by data and services subject to real-time function composition and interoperability on a massive scale. Concepts of bounded system architectures, while still necessary, are themselves insufficient to support specification, design, and development in this environment.

In particular, we see a need for rigorous scale-free engineering methods to specify the behavior of large-scale network systems and their parts, and to refine and reconcile these specifications in designs and implementations often populated by existing services. While new system services will always require design and implementation or acquisition, the process of design in future systems will extend to an operational capability for real-time discovery and composition of existing services as on-the-fly designs that satisfy specified enterprise operations. Our initial research points up the value of three entities, namely, flows, services, and quality attributes, as primary engineering artifacts for analysis and design in a world of such large-scale network integration and interoperability. Flows are user task flows and their corresponding architecture traces, services are system services composed to satisfy user flows, and quality attributes are familiar properties of security, survivability, reliability, etc., associated with flows and realized by compositions of attribute-enabled system services. The Flow-Service-Quality (FSQ) Framework [Hevner et al. 2001, Hevner et al. 2002] provides engineering representation and reasoning methods for developing these complex systems. In particular, the framework provides a foundation for assessing and managing system survivability. The objective of FSQ research and development is to provide theoretical foundations, language representations, and rigorous yet practical unified engineering methods to represent and reason about system flows as essential artifacts of system specification, design, and operation. In operation, quality attributes are treated as computational system capabilities for evaluation against quality requirements of requested flows. It is these three first-class concepts, namely, flow, service, and quality that can form a basis for unified engineering of large-scale network systems.

The Flow-Service-Quality Framework

Distributed network systems exhibit extensive asynchronous behavior in their virtually unknowable interleaving of communications among system components. In addition, system complexity is compounded by indeterminate boundaries, ever-changing couplings to other systems, and continual requirements for functional evolution. The development of these systems is essentially a large-scale integration activity that seeks to reconcile and satisfy user requirements through combinations of service components, often within a framework of pre-defined environments, enabling technologies, and domain architectures. The central issue in modern systems development is how to maintain intellectual control over such complex structures and the asynchronous behaviors they produce. In short, what are the stable and dependable anchors for specification, design, and operation that can provide a unified engineering discipline for large-scale network system development? Our research is focused on developing an effective answer to this question and applying it to survivability analysis and design.

Distributed systems are usefully viewed as networks of asynchronously communicating components, where the components provide services whose functions can be combined in various patterns to satisfy business requirements. The sequencing and alternation of services in user task flows can be mapped into compositions of network component functions. These compositions are end-to-end traces that define slices of network architectures whose net effect is to carry out operations that satisfy user requirements. Large-scale systems support many concurrent users in many roles with many possible flows, and particular services may appear over and over in their definitions. In fact, a principal design objective in large-scale systems is coordination and synchronization of multiple uses of particular services specified by control flows. In dynamic networks with constantly varying function and usage, flows and their corresponding traces of services serve as stable foundations for functional and non-functional (quality attribute) specification and intellectual control. Flows are composed of services and are evaluated against a set of quality attributes:

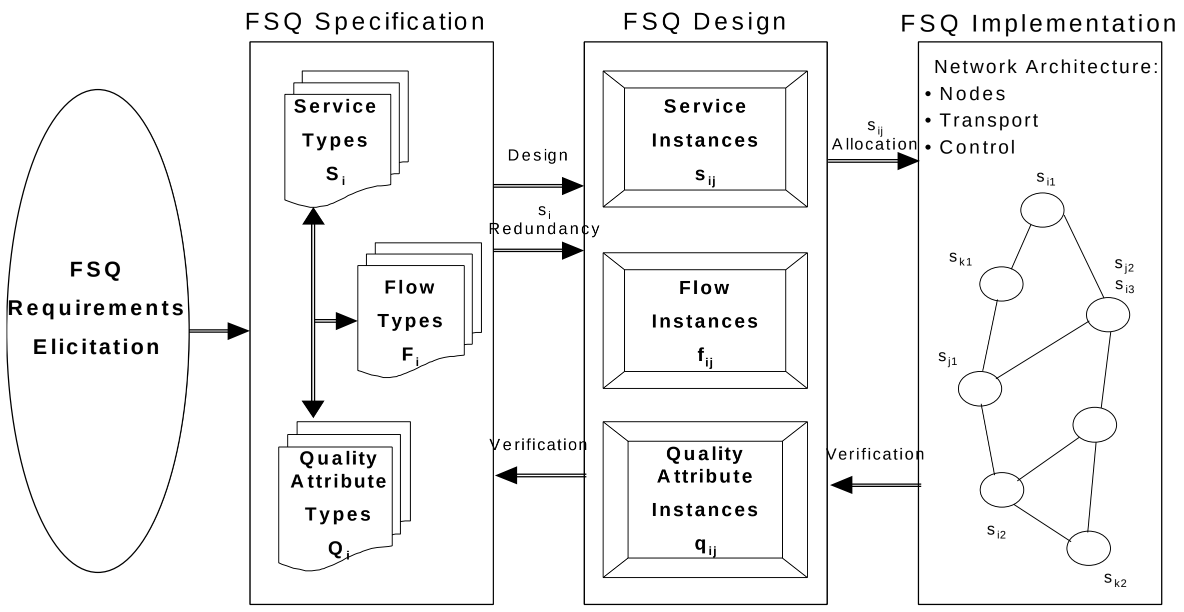
Flow: A flow is expressed using a flow structure language that specifies user requirements in precise terms. Flows can be represented as procedural structures composed of nested and sequenced service invocations that are expressed in terms of sequence, alternation, iteration, and concurrent structures. Flows also define required levels of quality attributes for themselves, as well as for execution of the individual services they reference. The semantics of the flow structure language preserves powerful reasoning methods of composition and referentially transparent refinement, abstraction, and verification.

Service. A service is a function provided by a system component or set of components. Services can be specified as state machine transition functions, where an input stimulus and the current state of the service results in an output result and a new state. Services can be composite or primary. Composite services are refined into sets of flows in a recursive process, whereas primary services perform basic functions and are not considered for refinement. It is necessary in a flow specification to define the behavior of each constituent service in terms of all possible responses resulting from all possible states generated by its uses in all the flows that reference it. Relational specifications over equivalence classes are suitable for this purpose. A relational specification can define the set of all possible responses generated by all the uses of a service in all the flows within which it appears. In this way, completeness and consistency can be achieved in flow definitions. Compact specifications are possible through equivalence classes that reduce the number of states to be considered.

Quality Attributes. Attributes of quality, such as security, survivability, reliability, and availability are defined as computational functions and are associated with both flows and services. Substantial effort has been devoted in the past to the development of descriptive and often subjective characterizations of non-functional system properties, or quality attributes, for example, survivability attributes [Ellison et al. 1999]. Rather than focusing on descriptive methods, we adopt an alternate approach and ask how such properties can be defined, computed, and acted upon as dynamic characteristics of system operation. That is, we wish to define quality attributes as functions to be computed, rather than as subjective descriptions of capabilities to be achieved. While such functions rely on what can be computed and may differ thereby from traditional views of non-functional properties, they may permit new approaches to property analysis and operational evaluation.

The FSQ Engineering Process

The FSQ Framework is composed of a structured flow language (SFL) and an engineering process for 1) specification of user task flow types in terms of required service types and quality attributes, 2) design and verification of architecture trace refinements in terms of services and quality attributes that satisfy specific instantiations of user task flows, and 3) implementation and verification of network refinements in terms of system hardware, software, and communications that satisfy architecture traces of user flows.



As shown in the figure, there are four development stages in the FSQ Framework:

- **FSQ Requirements:** FSQ requirements can be expressed in terms of necessary services, plus their quality attributes. Services can be described in functional terms at their highest level of abstraction. Services may themselves require flow refinements that are dictated by the user's business rules. Each service will require a set of quality attributes for itself and any system implementation that provides the service.
- **FSQ Specification:** A thorough system requirements analysis will allow formal specification of flows, services, and quality attributes. The close interactions among the three types will result in rigorous closure verifications. For example: Are the set of services necessary and sufficient for all user task flows? Do the

flows and services define their quality attributes in ways supportive of clear selection of service implementations?

- **FSQ Design:** The design stage will generate flow, service, and quality attribute instances. Each instance is a fully realized design of the corresponding FSQ specification. Quality criteria (e.g. security, survivability) may dictate design of redundant service and flow instances, which can be verified against FSQ specifications.
- **FSQ Implementation:** The network architecture provides the foundation for the allocation of service instances onto the distributed system. The network architecture consists of the system nodes, the transport connections between nodes, and the distribution of system control and communication among the nodes. The implementation of the distributed system can be verified against the FSQ designs.

The FSQ Research Vision

The following observations summarize our research vision:

- The FSQ Framework permits unification of network system engineering through uniform, scale-free semantic structures for requirements, specification, design, and implementation; and generic control architectures for managing system operation, service quality, adaptation, and evolution.
- The FSQ engineering view of system development is based on a seamless hierarchical decomposition beginning with user flow and service requirements and ending with flow and service implementations, with intrinsic traceability from requirements through implementations.
- User task flows of services supporting enterprise operations drive FSQ system engineering. Systems are composed in terms of user views of services, as opposed to strictly functional decomposition or object-based composition.
- The FSQ Framework provides a recursive development process wherein flows define sets of services and services define sets of flows.
- Asynchronous behavior is effectively dealt with by relational specifications of services. Every service use in a flow must account for all possible responses resulting from asynchronous uses of that service in all the flows in which it appears, with the specific response obtained determined by a postfix predicate on every use. This use-predicate structure bounds the inherent non-determinism of asynchronous execution, and enables reasoning methods of referentially transparent composition, abstraction, and refinement.
- The effort required for FSQ system scale-up through addition of user flows and services is no more than linear due to the inherent localization and absence of transitive propagation of effects of such additions.
- Any flow can be constructed from a basis set of single-entry/single-exit primitive structures that provide service uses in sequence, alternation, iteration, and concurrent forms. Flows expressed in this form can be refined, abstracted, and verified with mathematical precision
- FSQ supports modeling of quality attributes as functional, computational entities. Concepts of system quality attributes are open to improved understanding and optimization in this representation. Survivability is targetted as the first attribute to be analyzed in this framework.

References

[Ellison et al. 1999] Ellison, R., Fisher, D., Linger, R., Lipson, F., Longstaff, T., and Mead, N., *Survivable Network Systems: An Emerging Discipline*, CMU/SEI-97-TR-013, November 1997, revised May 1999.

[Hevner et al. 2001] Hevner, A., Linger, R., Sobel, A., and Walton, G., *Specifying Large-Scale Adaptive Systems with Flow-Service-Quality Objects*, OOPSLA 2001 (to appear).

[Hevner et al. 2002] Hevner, A., Linger, R., Sobel, A., and Walton, G., *The Flow-Service-Quality Framework: Unified Engineering for Large-Scale Adaptive Systems*, Hawaii International Conference on System Sciences, Kona, HI 2002 (to appear).

[Mead et al. 2000] Mead, N., Ellison, R., Linger, R., Longstaff, T., and McHugh, J., *Survivable Network Analysis Method*, CMU/SEI-2000-TR-013, September 2000.