Survivability Measure

J. K. Millen

June 29, 2000

1 Background

In the paper "Local Reconfiguration Policies" [3], a system was viewed as a collection of components configured to provide a set of user services. Electronic mail, for example, in a local area network, requires a workstation, the cable and associated interface devices, a gateway to the internet service, and so on. Components are not simply hardware devices, but functional combinations of hardware and software.

To study fault tolerance and reconfiguration, we focussed attention on the fact that different sets of components can support the same service. Then, if some components fail, they can be replaced by others in a different configuration. Reconfiguration to restore services as a result of component failure was viewed as a kind of "flow" analogous to information flow. We applied Meadows' theorem on datset aggregates [2] to characterize the maximum safe flow policy.

A service is characterized by the set of alternative configurations that can support it. A service configuration assigns components to support a service. A service configuration is not fully defined by the set of components it employs – two different configurations can use the same set of components. (An example is the use of "I" and "V" to support Roman numerals "IV" or "VI".)

At a given time, a system is in some *state* where a set of services are being supported simultaneously by the set of currently available components. Different subsets of these components are configured to support the various services in way that respects the ability or inability of a component to be shared by more than one service.

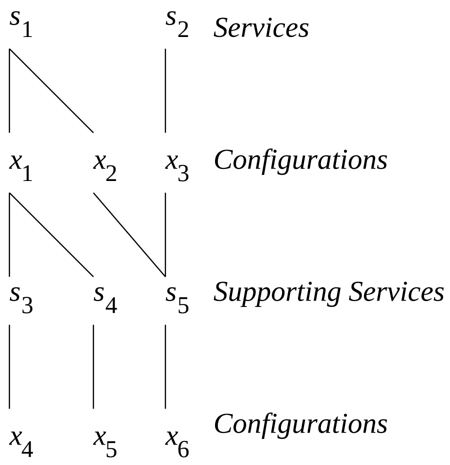

Figure 1: Service Hierarchy

Services were given a *survivability* ordering: one service is no more survivable than another if every service set that supports the first, also supports the second.

1.1 A Hierarchical View

The earlier view of a system was flat. A component was thought of as atomic, and different from a service. Taking a deeper view, now, we look at components through a microscope, and we see that a component is (sometimes) a *lower-level service* with its own components and configurations. A workstation has a keyboard and a display and a CPU, and the CPU depends on software as well as hardware, such as a mail application and an operating system. The hardware is also built on cards, the cards on chips and other components, etc. This hierarchical view is suggested in Figure 1, which illustrates how the service-configuration-component architecture is repeated at lower levels.

If every service has lower-level component services, the hierarchy could be bottomless. However, the lower parts of the hierarchy cease to be interesting when services are isolated from one another in the sense that they have no common components. Whether or not the hierarchy is infinitely deep, we can treat certain services as *atomic* for purposes of analysis.

The service hierarchy has an AND-OR structure that is reminiscent of AND-OR fault trees traditionally used in dependability analysis (see, for example, [1]). A service is supported by an OR of configurations, any one of which may be selected, while each configuration requires and AND of the services supporting it. This relationship may be propagated down the hierarchy to arrive at an AND-OR combination of atomic services. The emphasis of this model, however, is on studying the dependency and survivability properties of all the services, not just the user interface services at the top and the atomic ones at an arbitrarily defined bottom.

In a hierarchical setting, the definition and nature of survivability must be re-examined. The rewards for doing so include new opportunities such as developing the concept of *dependency* between services and defining a *measure* of survivability.

2 Systems

The formal definition of a system has changed to reflect the fact that components are services. The notion of a basis mapping in [3] has been recast as a collection of *configurations,* where each configuration is the part of the basis mapping applying to a given service.

Definition 2.1 (System) *A* **system** X *is a set of* **configurations***, together with a partition* $\overline{X}$ *on* X*. The classes in* $\overline{X}$ *are called* **services***, and if* $x \in X$*,* $\overline{x} \in \overline{X}$ *is the service containing* x*. Also, for* $x \in X$*, there is a set* $\underline{x} \subseteq \overline{X}$ *of services that* **support** x*. The support function is* **irreflexive** *in the sense that* $\overline{x} \notin \underline{x}$*.*

In effect, a service is identified with its set of configurations, where each configuration associates that service to a possible set of supporting services.

Configurations of (in) the same service are **irredundant** *with respect to support sets in the sense that, if* $\overline{y} = \overline{x}$ *and* $\underline{y} \subseteq \underline{x}$*, then* $y = x$*.*

The irredundancy condition says that two distinct configurations of the same service cannot have the same or nested support sets. Thus, if two configurations have the same support set, they must belong to different services (like "IV" vs. "VI" in the earlier example). We do not allow two configurations of the same service to have nested support sets because one of them would not be minimal.

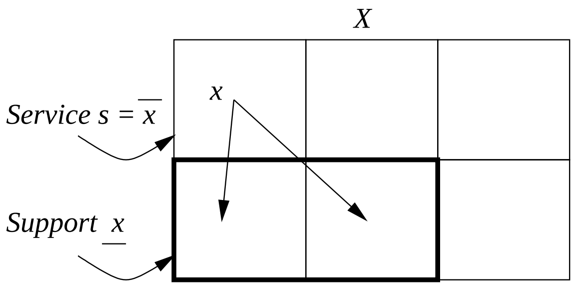

Figure 2: System Model

The irredundancy condition implies that if a service $\overline{x}$ is **atomic**, in the sense that it has an empty support, $\underline{x} = \emptyset$, then it can have only one configuration, so $\overline{x} = \{x\}$. There can be any number of atomic services, however, since different configurations (for different services) can have the same support sets.

Any service set u containing a support set, that is, such that $\underline{x} \subseteq u$, may also be said to **support** not only x but also the service $\overline{x}$. However, this terminology is used only when $\overline{x}$ is not atomic.

When components can be services with their own components, there is the possibility of cycles in the support relation. We know that $\overline{x} \notin \underline{x}$, but could there be longer cycles? We have tacitly discouraged this notion by referring to our model as a "hierarchy," but the formal restriction on this comes below when we introduce the "covering" relation.

The survivability of two services can be compared using this definition of support. One service is (equally or) less survivable than another if every service set that supports the first service will also support the other one.

Definition 2.2 (Survivability Pre-ordering) *For nonatomic services s and t, $s \sqsubseteq t$ if (u supports s implies u supports t.)*

This is a "flat" notion of survivability, which only enables us to compare services that are at the same level in the hierarchy. Part of our plan is to define a measure of survivability that applies to all services, enabling any pair to be compared.

The survivability relation is a pre-ordering because, as remarked earlier, distinct services can be supported by the same set of components.

The relation can be expressed in terms of configurations. First, we extend the overbar and underbar notations to sets of configurations in the natural way:

Notation. If $p \subseteq X$, we write

- $\overline{p} = \{\overline{x} | x \in p\}$, and
- $\underline{p} = \{\underline{x} | x \in p\}$.

In particular, if s is a service, $\underline{s}$ is the set of service sets supporting (some configuration of) s.

Note that "bar" operations are monotone: if $p \subseteq q$ then $\overline{p} \subseteq \overline{q}$ and $\underline{p} \subseteq \underline{q}$.

Theorem 2.1 (Basis Comparison) *$s \sqsubseteq t$ iff for all $u \in \underline{s}$ there exists $v \in \underline{t}$ such that $v \subseteq u$.*

Proof. Suppose that $s \sqsubseteq t$ and $u \in \underline{s}$. Then $u = \underline{x}$ for some $x \in s = \overline{x}$. In particular, u supports s. So u supports t. There exists $y \in t$ such that $\underline{y} \subseteq u$. Let $v = \underline{y}$. This proves one direction.

Now assume that for all $u \in \underline{s}$ there exists $v \in \underline{t}$ such that $v \subseteq u$, and suppose u supports s. There exists $x \in s = \overline{x}$ such that $\underline{x} \subseteq u$. But $\underline{x} \in \underline{s}$ so there exists $v \in \underline{t}$ such that $v \subseteq \underline{x} \subseteq u$. So u supports t, as desired. ∎

3 Hierarchical State

The operational state of a system at a given time is characterized by a set of configurations that are currently *active*. In a state, each supported service has at least one active configuration. Not all the services in the system need be supported.

There is a fundamental completeness property for a state, relating lower-level and higher-level services: if a service is supported in the state, the services supporting its configuration are supported in the state as well.

Definition 3.1 (State) *A set of configurations $p \subseteq X$ is a* **state** *if, for every $x \in p$, $\underline{x} \subseteq \overline{p}$.*

Note that X itself is a state, and that the union of states is a state. Furthermore, every atomic service is a state.

3.1 Realizability

States may or may not be *realizable*, because they may require some components to be shared by more than one configuration. Whether that is permissible or not depends on the details of the services and the components. To handle this concern abstractly, we merely assume that there is a set of realizable states associated with the system. The set of realizable states is $\mathcal{R}$. (Thus, the full formal definition of a system would include the configuration set X, the service partition and support mapping, and $\mathcal{R}$.)

In [3], there was a "Spare Axiom" stating that if a state was realizable, one could reduce its services and add more components, and the resulting state would also be realizable. Here, we state a similar axiom about realizable states.

Note that while X is a state, it is not necessarily (or even probably) realizable.

Axiom 3.1 (Contraction) *If f is a realizable state and g is a state such that $g \subseteq f$ then g is realizable.*

Contracting a state by reducing the set of configurations in it can only eliminate some sharing, and cannot destroy its realizability. We still have to check whether the smaller set is a state, since arbitrary deletion of configurations does not always preserve the completeness property of a state.

Also as in [3], there is an axiom saying that a state is realizable if it requires no sharing of services by configurations.

Axiom 3.2 (Disjoint Realizability) *If a state p is pairwise disjoint then p is realizable.*

3.2 Ordering

Definition 3.2 (Configuration Covering) *A configuration x* **covers** *a configuration y if $\overline{y} \in \underline{x}$. Let "$\geq$" be the reflexive, transitive closure of the covering relation.*

That is, $x \geq y$ if $x = y$ or x covers y or x covers $z \geq y$ for some z. We will also refer to $\geq$ as the covering relation.

The covering relation on configurations induces a covering relation on services. In particular, $\overline{x}$ covers $\overline{y}$ if x covers y.

Definition 3.3 (Service Covering) *A service s* **covers** *a service t if $t \in \bigcup \underline{s}$.*

The transitive covering relation $\geq$ is extended to services similarly.

The covering relation is assumed to be noetherian in the sense that there are no infinite sequences $x_1 > x_2 > ....$ This implies, in particular that no covering cycles, and hence no support cycles, exist. It also means that $\geq$ is a partial ordering.

A cycle of covering links would imply mutual dependencies between configurations and between the services they support. This is not impossible, but it complicates analysis. We can avoid this in practice by assuming that mutually dependent services are packaged together in one service.

Axiom 3.3 (Noetherian) *Every sequence $x_1, x_2, ...$ such that x_i covers x_{i+1} is finite.*

Definition 3.4 (Cone) *Give a set of configurations p, and $x \in p$, we define the* **cone** *of x in p to be $p^x = \{y \in p | x \geq y\}$.*

Note that cones are partially ordered in the same way as their generators.

Theorem 3.1 (Cone Covering) *If $x \in p$ and $x \geq y$ then $p^x \supseteq p^y$.* ∎

The cone of a configuration in a state is a state.

Theorem 3.2 (State Cone) *If p is a state and $x \in p$, then p^x is a state.*

Proof. Because the covering relation is noetherian, we can prove this inductively for all $x \in p$.

If x covers nothing, $\underline{x} = \emptyset$ (so $\overline{x}$ is atomic), $p^x = \{x\}$ and this set is trivially a state.

Now the induction step: Suppose, for all y, if x covers y then p^y is a state. We claim that p^x is a state. Let $z \in p^x$.

Claim: $\underline{z} \subseteq \overline{p^x}$. Since $z \in p^x, z \in p$ and $x \geq z$. Either $x = z$ or, for some y (possibly $y = z$), x covers y and $y \geq z$. If $x = z$ then the claim is true trivially. So assume x covers y for some $y \geq z$.

Then $z \in p^y$, and since p^y is a state we have $\underline{z} \subseteq \overline{p^y} \subseteq \overline{p^x}$, as required. ∎

The cone of a configuration in a realizable state is realizable by the contraction axiom.

3.3 Dependencies

A service depends on a component service if every state offering that service contains that component. More generally, a service depends on a set of services if at least one of those services must be present.

Definition 3.5 (Cut Set) *$u \subseteq \overline{X}$ is a* **cut set** *for s if for every state p, if $s \in \overline{p}$ then $u \cap \overline{p} \neq \emptyset$.*

Definition 3.6 (Dependency) *s* **depends** *on t if $\{t\}$ is a cut set for s.*

Proposition 3.3 (State Dependency) *s depends on t iff for all states p, if $s \in \overline{p}$ then $t \in \overline{p}$.*

Note that if s is atomic, $s = \{x\}$ is a state and s depends only on itself.

Lemma 3.4 (Covering Dependency) *If s depends on t, then either $s = t$ or, for every $x \in s$, there exists $r \in \underline{x}$ such that s covers r and r depends on t.*

Proof. Suppose $s \neq t$. Then s is not atomic. Choose any $x \in s$. We know that $\underline{x} \neq \emptyset$ since s is not atomic. For any $r \in \underline{x}$, s covers r.

Suppose, in contradiction to the lemma, that no $r \in \underline{x}$ depends on t. Then, for each $r \in \underline{x}$, there exists a state p_r such that $r \in \overline{p}$ but $t \notin \overline{p}$.

Then $q = \bigcup_{r \in \underline{x}} p_r \cup \{x\}$ is a state. (We have to check that $\underline{x} \subseteq \overline{q}$, but this is true because each $r \in \overline{p_r}$.)

This contradicts the assumption that s depends on t, since $s = \overline{x} \in \overline{q}$, but $t \notin \overline{q}$. ∎

This shows that dependency is a partial ordering.

Theorem 3.5 (Dependency Partial Ordering) *Dependency is a partial ordering.*

Proof. Reflexivity and transitivity are trivial. Antisymmetry follows from the Covering Dependency lemma, since covering yields a partial ordering. ∎

A service can depend on a component two levels down without a dependency on anything in between. See Figure 3.

4 Survivability Measure

We seek some measure of survivability that would allow an arbitrary pair of services to be compared. In fact, we would like it to be defined on all components, including atomic services. It could be a numerical measure, or, more generally, and by analogy with multilevel security, it could have values in a lattice. What properties should a survivability measure σ into a lattice satisfy?

Clearly, the ordering of survivability lattice values should respect the existing survivability ordering, so (using $\leq$ for the lattice partial ordering, with $\vee$ and $\wedge$ for join and meet):

S1: $s \sqsubseteq t$ implies $\sigma(s) \leq \sigma(t)$, for s, t nonatomic.

Furthermore, if a service depends on a component, it cannot be more survivable, so

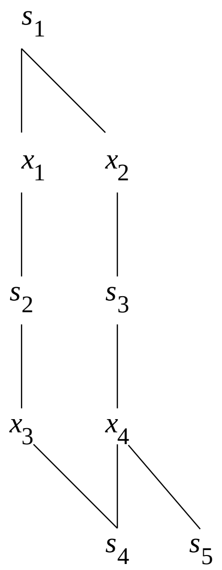

Figure 3: Indirect Dependency

S2: if s depends on t, then $\sigma(s) \leq \sigma(t)$.

Consider now the special case of dependency where a service is has only one configuration supported by exactly one component, i.e., $s = \overline{x} = \{x\}$ and $\underline{x} = \{t\}$. The relationship of the services s and t is so close that they ought to have exactly the same survivability:

S3: if $\underline{s} = \{\{t\}\}$ then $\sigma(s) = \sigma(t)$.

We will define a survivability measure as any function on services into a lattice satisfying S1-S3.

Definition 4.1 *$\sigma : \overline{X} \to L$ is a* **survivability measure** *if L is a lattice and σ satisfies S1-S3.*

4.1 Max-Min Survivability

In this section we suggest a particular survivability measure. Strictly speaking, this is not a single measure, but rather a schema for defining a measure once we are given the lattice and the survivability of each of the atomic components. These can be assigned independently since none of S1-S3 apply to them alone.

Definition 4.2 (Max-Min Survivability) *σ is* **max-min** *if for all non-atomic services s,*

$$\sigma(s) = \bigvee_{x \in s} \bigwedge_{r \in \underline{x}} \sigma(r)$$

With this definition, we can prove the desired properties S1-S3.

Theorem 4.1 (Max-Min S1) *Suppose σ is max-min, $s \sqsubseteq t$ and s is not atomic. Then $\sigma(s) \leq \sigma(t)$.*

Proof. By the Basis Comparison Theorem, for all $u \in \underline{s}$ there exists $v_u \in \underline{t}$ such that $v_u \subseteq u$. This means that, since s is not atomic, we may write

$$\begin{aligned}
\sigma(s) &= \bigvee_{x \in s} \bigwedge_{r \in \underline{x}} \sigma(r) \\
&= \bigvee_{u \in \underline{s}} \bigwedge_{r \in u} \sigma(r) \\
&\leq \bigvee_{u \in \underline{s}} \bigwedge_{r \in v_u} \sigma(r) \\
&= \bigvee_{\{v_u | u \in \underline{s}\}} \bigwedge_{r \in v_u} \sigma(r) \\
&\leq \bigvee_{v \in \underline{t}} \bigwedge_{r \in v} \sigma(r) \\
&= \sigma(t). \blacksquare
\end{aligned}$$

Theorem 4.2 (Max-Min S2) *Suppose σ is max-min and s depends on t. Then $\sigma(s) \leq \sigma(t)$.*

Proof This is proved by induction on the covering relation. The result is trivial if $s = t$, so assume $s \neq t$. By the max-min formula for $\sigma(s)$, it suffices to show that, for all $x \in s$, there exists $r \in \underline{x}$ such that $\sigma(r) \leq \sigma(t)$. But, by the Covering Dependency lemma, there exists $r \in \underline{x}$ such that r depends on t, and s covers r, so the conclusion follows from the induction hypothesis. $\blacksquare$

Theorem 4.3 (Max-Min S3) *If σ is max-min and $\underline{s} = \{\{t\}\}$ then $\sigma(s) = \sigma(t)$.*

Proof. Immediate. ∎

Corollary 4.4 (Max-Min Survivability) *Any max-min function on $\overline{X}$ is a survivability measure.*

Because the properties S1-S3 are ordering properties, they are preserved if the survivability function is remapped by any isotone function. A function f from a lattice to a lattice is *isotone* (sometimes called "monotone") if $a \leq b$ implies $f(a) \leq f(b)$. An isotone function is not necessarily a lattice homomorphism, which preserves meets and joins.

Theorem 4.5 (Isotone Survivability) *If σ is a survivability measure, so is $f \circ \sigma$, for any isotone lattice function f.*

4.2 Probabilistic Example

A non-max-min example of a survivability measure can be constructed by using probability (of mission availability, for example) as a measure, with atomic survivabilities assigned arbitrarily, and assumed independent. The partial ordering on probabilities is the usual total ordering of reals in [0,1].

Let $\alpha(s)$ be the event that s is available, and let $\sigma(s)$ be the probability of this event, $\Pr[\alpha(s)]$. If s is not atomic, $\alpha(s)$ is the event that all services in some configuration of s are available. This leads us to the following:

Definition 4.3 (Event Map) *α is an* **event map** *if α maps services to measurable subsets of a probability space, in such a way that, for non-atomic s, $\alpha(s) = \bigcup_{x \in s} \bigcap_{r \in \underline{x}} \alpha(r)$.*

It follows immediately that:

Proposition 4.6 (Event Max-Min) *an event map is max-min.*

While an event map itself could be used as a survivability measure, we can go one more step and take the probabilities of the events.

Theorem 4.7 (Probabilistic Survivability) *If $\sigma(s) = Pr[\alpha(s)]$, where α is an event map, then σ satisfies S1-S3.*

Proof. The probability measure itself is isotone, so we may apply the Isotone Survivability Theorem. ∎

This survivability measure is not max-min, since the probability of a union or intersection event is not the join (Max) or meet (Min) of the probabilities of the component events. For example, if s has one configuration consisting of atomic services s_1, s_2, we would have $\sigma(s) = \sigma(s_1)\sigma(s_2)$, rather than $\text{Min}(\sigma(s_1), \sigma(s_2))$.

While applying isotone functions gives us a larger class of survivability measures than max-min measures alone, there are survivability measures that cannot be described this way.

Proposition 4.8 (Non-Isotone Survivability) *There is a survivability measure (satisfying S1-S3) that is not the composition of an isotone function with a max-min function.*

Proof.(This example is due to Bruno Dutertre.) Consider $s = \{x, x'\}$ where $\underline{x} = t$ and $\underline{x'} = t'$ Suppose $\sigma(s) = 0$ but $\sigma(t) = \sigma(t') = 1$, with the usual ordering $0 \leq 1$. The properties S1-S3 apply vacuously, so σ is a survivability measure.

For any max-min function μ, $\mu(s) = \mu(t) \vee \mu(t')$. Hence, if σ is expressible as $f \circ \mu$ for isotone f, we would have $\sigma(t) \leq \sigma(s) = 0$, impossible if $\sigma(t) = 1$. ∎

5 Conclusions and Future Work

We have extended the survivability model in the Reconfiguration Policy paper from a flat service-component concept to an arbitrarily deep hierarchy reflecting the engineering depth of real systems. Dependency of one service on another is defined, with the expected properties. Some general properties of a lattice-valued survivability measure have been noted, and a simple class of such measures suggested.

Many questions remain unanswered. First, we do not have a complete characterization of survivability measures. It would also be natural to examine the consequences of the kind of structuring tactics that is common in implemented systems, such as layering and mutual dependency of encapsulated modules.

References

[1] M. Boyd and D. Iverson, "Digraphs and fault trees: a tale of two combinatorial modeling methods," *1993 Proc. Annual Reliability and Maintainability Symposium*, 1993.

[2] C. Meadows, "Extending the Brewer-Nash Model to a Multilevel Context," *1990 IEEE Computer Society Symposium on Research in Security and Privacy,* pp. 95-102.

[3] J. Millen, "Local reconfiguration policies," *1999 IEEE Security and Privacy Symposium*, IEEE Computer Society, 1999.