ISR Institute for Software Research
University of California, Irvine

# Temporally Expressive Scenarios in ScenarioML

**Thomas A. Alspaugh**
University of California, Irvine
alspaugh@ics.uci.edu

May 2005

**ISR Technical Report # UCI-ISR-05-6**

**Institute for Software Research**
ICS2 210
University of California, Irvine
Irvine, CA 92697-3425
**www.isr.uci.edu**

# Temporally Expressive Scenarios in ScenarioML

Thomas A. Alspaugh
Institute for Software Research
University of California, Irvine
Irvine, CA, USA 92697-3425
alspaugh@ics.uci.edu

ISR Technical Report # UCI-ISR-05-06
May 2005

## Abstract

*Sequential, non-overlapping events are the norm in traditionally-expressed scenarios and use cases, but the world is much more fluid. Events have duration and may overlap, be separated in time, begin or end together, or have various other specific temporal relations. The ordering of the events may be completely known or partially uncertain, resulting in any of a large (but finite) number of relations for any two events. These relations, which can be formally stated and manipulated, are separable in form and meaning from the events themselves, which in requirements are most often expressed in prose. The temporal relations and partial ordering of events can be a significant part of what is specified, and must be inferred by a reader if not explicitly expressed. This paper presents a scenario language, ScenarioML, which expresses requirements scenarios using a broad and effective selection of event relations and structures. ScenarioML scenarios range from concrete scenarios to parameterized schemata that represent large families of scenarios related in a variety of temporal and structural ways. The language is designed for automated analysis and operations on temporal event relations, as well as other aspects of scenarios. An example from aircraft navigation is presented.*

# Temporally Expressive Scenarios in ScenarioML

Thomas A. Alspaugh
Institute for Software Research
University of California, Irvine
Irvine, CA, USA 92697-3425
alspaugh@ics.uci.edu

ISR Technical Report # UCI-ISR-05-06
May 2005

## Abstract

*Sequential, non-overlapping events are the norm in traditionally-expressed scenarios and use cases, but the world is much more fluid. Events have duration and may overlap, be separated in time, begin or end together, or have various other specific temporal relations. The ordering of the events may be completely known or partially uncertain, resulting in any of a large (but finite) number of relations for any two events. These relations, which can be formally stated and manipulated, are separable in form and meaning from the events themselves, which in requirements are most often expressed in prose. The temporal relations and partial ordering of events can be a significant part of what is specified, and must be inferred by a reader if not explicitly expressed. This paper presents a scenario language, ScenarioML, which expresses requirements scenarios using a broad and effective selection of event relations and structures. ScenarioML scenarios range from concrete scenarios to parameterized schemata that represent large families of scenarios related in a variety of temporal and structural ways. The language is designed for automated analysis and operations on temporal event relations, as well as other aspects of scenarios. An example from aircraft navigation is presented.*

## 1 Introduction

Scenarios are widely used in a number of ways during the development process, and by a variety of participants [BFJ92, Ale04]. Stakeholders use them to communicate what is wanted, and developers use them to confirm their understanding [BFJ92, MMM98, Rob04]. Both may use them as the primary form in which requirements are recorded [Coc00, Hau04, JCJ92], or as a preliminary form from which specialists produce more refined forms such as goals and requirements [LW98]. Developers may use them as guide and scaffolding in a process by which other artifacts such as goals are developed [RSBA98]. They are used to simulate and explore a system's use [BFJ92] or a design's utility [Sut03], and to present a test or validation [BBL04].

Scenarios range in form from prose narratives to structured models of event sequencing. Prose scenarios can express or imply subtle and complex temporal relations among the events they describe, but more-structured scenario forms are often focused on other aspects of narrative. Use cases organize the numerous alternatives and exceptional cases that can occur during purposeful transactions to achieve a goal. Message Sequence Charts and Sequence Charts clearly present the sequences of communications between actors and system components. Activity Diagrams and flowchart forms highlight the decisions that are made over the course of a transaction. Each of these forms is useful in its own way and its own context.

What constitutes an event in a scenario is variously described in the literature: something that happens, an action, an interaction, a step, or in some usages only an instantaneous change from one state to another. Our view is that any of these may constitute an event, but that an event is fundamentally something occurring over a time interval, not instantaneously. While a number of commonly-used formalisms (such as Message Sequence Charts and Sequence Diagrams) assume non-overlapping events for simplicity (often expressed as time points or restricted to be state changes), in the real world closer examination shows that no event is truly indivisible. Rather, a real event that

appears to be instantaneous at one level of detail, abstraction, or concepts, is revealed at a finer-grained level to be composed of smaller sub-events or to be better described by concepts that distinguish its time span and evolution over that time.

It is notable that (to paraphrase Rousseau) events occur freely in the world, but are usually described in linear succession, or perhaps in concurrent groups of linear successions; events have duration, but are usually described as if instantaneous. This is advantageous for specifying or designing a solution whose components already operate on this model, because it is a highly effective simplification and can avoid thorny challenges that concurrency can bring. But it is not effective for accurately and fluently describing situations whose events are more subtly related to begin with.

This paper describes a form, ScenarioML, whose purpose is the expression of a broad variety of event relations and orderings. The events it describes are assumed to have duration, and the relations of interest between them are qualitative. ScenarioML is oriented toward temporally complex and subtle narratives, such as are often found in descriptions of domains and group interactions, and during early requirements activities of all kinds. Its temporal relations are based on Allen's interval algebra. Interval algebra is a formalism that exhaustively expresses all possible qualitative temporal relations between definite or indefinite intervals, and (with certain caveats) supports temporal inference on them [All83]. A ScenarioML scenario states what is essentially an arbitrarily subtle partial order of potentially-overlapping events. ScenarioML is designed to express complex temporal situations, but also to express simple relations simply.

Though this paper does not focus on them, ScenarioML also provides a panoply of operators for expressing sets of partial orderings: iteration, alternation, permutation, interruption, and exception; episodes for reuse of events and event orderings; parameterization of scenarios and episodes; refinement operators to hide or incorporate additional levels of detail; and markup to link instances of defined terms with their definitions.

As its name suggests, ScenarioML is expressed in XML (Extensible Markup Language), which takes advantage of a large and growing range of publicly available tools and packages and makes the language more effective as a medium of interchange among researchers, practitioners, and tools for analysis and operations. Specialized tool support is under development, with prototypes completed for some tool functions. An HTML presentation format eases reading and understanding of ScenarioML scenarios, and appears in Figure 7.

The remainder of this paper is organized as follows. Section 2 gives an example that motivates the discussion to follow. Section 3 discusses Allen's interval algebra, and Section 4 presents ScenarioML, focusing on the temporal aspects which are defined in terms of Allen's interval algebra. Section 5 discusses analyses and operations on ScenarioML scenarios. Section 6 contrasts the related work, and Section 7 presents lessons learned and future work.

## 2 A motivating example

As a motivating example, we will consider a description of a typical flight of a small plane from Palomar Airport to Riverside Airport, both in the U.S. state of California. The description is based on a recent flight in which the author sat in the co-pilot's seat of a small plane, collecting data on navigation and pilot cognitive workload.

> While waiting for clearance to take off, the pilot begins entering the waypoints for the route from Palomar to Riverside into the plane's GPS (Global Positioning System) device. The pilot taxis to the end of the runway, the airport tower clears the plane for takeoff before the entire route is entered, and the pilot takes off. Once the plane climbs high enough to enter the TIS (Traffic Information System) radar service area, the TIS receives data about the location and path of nearby planes from radar stations on the ground, and warns the pilot if another plane approaches. Shortly after this altitude is reached, the pilot resumes entering the remaining waypoints. The TIS allows the pilot to safely devote attention to the GPS knobs and display without having to look outside the airplane at all times to scan for approaching traffic. Each waypoint must be entered before the GPS can guide the pilot to it, but otherwise the waypoints may be entered at any time before that, and can be changed along the way to a different route if necessary.
>
> The waypoints are identified by brief names such as KCRQ and PDZ, and correspond to radio navigation beacons on the ground or to the intersections of the standard air routes between beacons. Today's flight begins at Palomar Airport (KCRQ) and continues to Oceanside (OCN) and the intersection of routes V23 and V363 (KRAUZ). It turns at the intersection of V363 and V8, unnamed on the standard charts but frequently referred to as POM12 because it is 12 NM (nautical miles) south of the Pomona beacon. From there, the route continues to Paradise (PDZ) and finally to the Riverside airport (KRAL). This is the standard flight plan from Palomar to Riverside.
>
> At two times during the flight, other planes come within approximately 2 NM and 1000 feet of altitude. The TIS plays an audible warning "Traffic, traffic" through the pilot's headphones and highlights the symbol representing that

plane on its display screen as a yellow circle. The pilot adjusts the scale of the display to show the other plane's relative position more clearly, considers the location of that plane and all nearby planes, and decides how best to maneuver to avoid the other plane. After the plane has been avoided, the pilot navigates back to the route and continues the flight.

Frequently (but not on this flight) the air traffic controllers re-route planes on this route, vectoring them 20° to the right to cross the corner of the now-closed El Toro Marine Corps Air Station and thus avoid traffic on the landing approach to Santa Ana Airport (Orange County) runway 19R. In this case, the pilot skips the POM12 waypoint and turns back onto the V8 route to PDZ when his vector crosses it.

This scenario displays several characteristics common in real-world scenarios:

- Several things happen at once.
- The interesting events have durations.
- Point events (such as arriving at the TIS horizon altitude) are usually significant not in themselves, but because they begin or end interval events.
- There are temporal constraints (usually connected to causation) between the events in a strand of the scenario.
- There are sometimes additional temporal constraints among events in different strands of the scenario (for example, between entering a waypoint and flying to it).
- Some events happen unpredictably.
- Some things might not happen at all (such as a TIS incident), or might happen several times.

If we were to represent this scenario using any of the traditional forms of a scenario description, some things will be awkward or difficult to represent, especially the relationships among non-sequential events and the specific kinds of unpredictability. Much temporal (and causal) information is likely to be lost.

## 3 Allen's Interval Algebra

Allen's interval algebra expresses qualitative relations between temporal intervals, and provides a basis for reasoning from a set of given relations on specific intervals to other relations that they imply [All83]. Allen defined thirteen *basic relations* between fixed time intervals, shown in Figure 1. These relations are pairwise disjoint and mutually exhaustive: two fixed intervals can be related by no more than one of the thirteen, and there are no two fixed intervals that are not related by one of the thirteen. This makes them an excellent basis for reasoning about intervals.

(The names listed for the relations are those used by Krokhin *et al.* [KJJ03], and the symbols are adapted from the same source but expressed as single letters.)

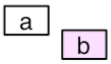
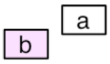
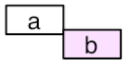
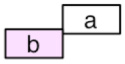
| Relation | | Converse | |
|---|---|---|---|
| **Graphically** | **Name** | **Name** | **Graphically** |
|  | precedes (p) | preceded by (P) |  |
|  | meets (m) | met by (M) |  |
|  | overlaps (o) | overlapped by (O) |  |
|  | finished by (F) | finishes (f) |  |
|  | contains (D) | during (d) |  |
|  | starts (s) | started by (S) |  |
|  | equals (e) | | |

**Figure 1. Allen's basic relations**

Each basic relation's converse is also one of the basic relations ("equals", e, is its own converse). For each distinct pair of converses, one relation's symbol is lower case and the other relation's is upper case. As an example, if a(p)b (a precedes b) for two intervals a and b, then the converse relation b(P)a (b is preceded by a) also holds.

Allen combined these basic relations to describe indefinite intervals whose position in time is incompletely known (referred to simply as "intervals" in the sequel). The relation between two intervals is expressed as the union of all possible basic relations that could hold between them. As an illustration, a GPS waypoint must have been entered before navigation to that waypoint can be done. Two possible basic relations can hold: the "Enter waypoint *w*" interval event could *precede* the "Fly leg to waypoint *w*" interval event, or it could *meet* it, but no other basic relation is consistent with the requirement. We would then say the temporal relation between the events is *(pm)*, the union of *p* and *m*. Of course, for the specific fixed events that are part of a real flight, only one of the two basic relations actually holds.

There are $2^{13}$ such unions of relations between indefinite intervals, and they exhaustively cover all the possible relations between two intervals (plus the null relation representing"no possible relation"). These unions are referred to simply as "relations" in the sequel. The converse of a relation is the union of the converses of its basic relations. The *intersection* of two relations is the set-theoretical intersection of the basic relations of the two relations.

Reasoning about more than two intervals is based on composition of relations. If *x, y,* and *z* are intervals related by *x(r)y* and *y(s)z,* then the relation between *x* and *z* is the *composition* of *r* and *s*, denoted $r \circ s$ , and we write $x(r \circ s)z$. The composition of two basic relations $b \circ b'$ can be calculated using the definitions of the basic relations. Composition of two general relations $r = (b_1 \cdots b_n)$ and $s = (b'_1 \cdots b'_m)$ is the union of the pairwise compositions of their component basic relations:

$$(b_1 \cdots b_n) \circ (b'_1 \cdots b'_m) = \bigcup \{b_i \circ b'_j \mid 1 \le i \le n,\ 1 \le j \le m\}$$

Allen and others published tables of composition of basic relations [All83, KJJ03], and we have published a Java command for calculating compositions [Als05]. Figure 2 shows the composition resulting from a(p)b and b(p)c, and we see that the composition results in a single possibility a(p)c. In general, however, the composition will be more complex. Figure 3 shows the composition of a(o)b and b(o)c, resulting in three possible basic relations between a and c. The composition is their union, a(pmo)c.

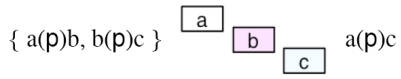



**Figure 2. Composition of Allen relation p.p**

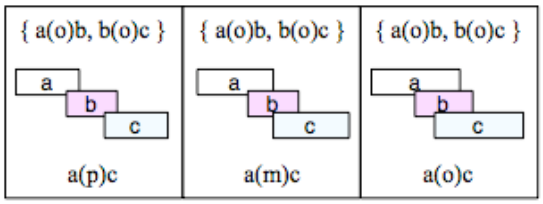



**Figure 3. Composition of Allen relation o.o**

The transitive closure of composition on a set of relations on intervals propagates the effect of each relation to all the other relations among intervals. Allen presented an algorithm for approximating this propagation. In essence, it attempts to infer the strongest temporal relation that can be shown to hold between each pair of intervals mentioned in a set of relations on intervals. Exact solution of inference for Allen's interval algebra has been shown to be NP-hard [VKB89].

For some sets of relations on intervals, such as { *x*(pm)*y*, *y*(pm)*x* } (*x* is both before and after *y*), there is no assignment of fixed intervals to the interval variables that satisfies the constraints. We say that the relations are not *satisfiable* in this case. The problem of determining whether an arbitrary set of relations on intervals is satisfiable has been shown to be NP-complete [VBK89]. However, determining satisfiability within certain subalgebras has been proven tractable; a *subalgebra* of Allen's interval algebra is a subset of the $2^{13}$ relations that is closed under intersection, converse, and composition. Eighteen maximal tractable subalgebras have been identified, in which satisfiability may be determined in polynomial time [KJJ03].

## 4 ScenarioML temporal relations

ScenarioML events are recursively structured, with simple events as the leaves of the structure tree and temporal groupings of progressively larger scope up to the root. A *simple event* is represented by the text that describes it:

<simpleEvent> The plane flies high enough to be in the TIS radar service area. </simpleEvent>

Simple events may be grouped together in twelve kinds of *temporal groupings*, summarized in Table 1. Each of the twelve kinds is characterized by the Allen relation that holds between adjacent events in the grouping. A temporal grouping of events is represented by the type of temporal grouping and the events contained in the group. The events may be simple events, or they may be other temporal groupings. An example of how ScenarioML expresses this appears in Figure 4, which gives a fragment of a ScenarioML representation of the small plane scenario from Section 2. The fragment shows the beginning of a Catenation temporal grouping and several levels of its subevents.

```
<catenation>
 <simpleEvent>The plane climbs high enough to be in
  the TIS radar service area.</simpleEvent>
 <comprehension>
  <simpleEvent> The plane flies in the TIS radar
   service area. </simpleEvent>
  <iteration iterationsCounted="*">
   <succession>
    <simpleEvent>Another plane comes within
     approximately 2 NM and 1000 feet in
     altitude.</simpleEvent>
    <succession>
     <simpleEvent>The TIS (Traffic Information
      System) identifies the plane and announces
      "Traffic, Traffic" through the pilot's
      headphones. </simpleEvent>
     <simpleEvent>The pilot punches the traffic
      button ("TRFC") on the instrument to
      display where the other plane is, relative
      to the pilot's. </simpleEvent>
     <simpleEvent>The TIS highlights the display
      symbol identifying the other plane.
      </simpleEvent>
```

**Figure 4. Small plane scenario in ScenarioML (excerpt)**

| For grouping (Allen relation) | Each event... | ...its predecessor in the group... |
|---|---|---|
| Separation (p) | begins after | ends |
| Catenation (m) | begins when | ends |
| Succession (pm) | begins after or when | ends |
| Supersession (oFD) | begins during | — |
| Comprehension (D) | occurs during | — |
| Progression (pmoFD) | begins after | begins |
| Nonregression (pmoFDseS) | begins no earlier than | begins |
| Indeterminate (pmoFDseSdfOMP) | occurs at any time | — |
| Concurrent (oFDseSdfO) | overlaps with | — |
| Connate (seS) | begins when | begins |
| Conterminate (Fef) | ends when | ends |
| Coeval (e) | begins (ends) when | begins (ends) |

**Table 1. ScenarioML temporal groupings**

The table gives the Allen relation for each grouping and a description in words that illustrates it. For example, in a Separation grouping, whose adjacent events are related by (p), each event begins after its predecessor in the group ends. In a Concurrent grouping, whose adjacent events are related by (OFDseSdfo), each event overlaps with its predecessor in the group. The temporal groupings are also illustrated graphically in Figure 6.

The names of the groupings were chosen to represent the relation between the grouping's events, as far as possible:

- A Separation's events are separated by an interval.
- The events of a Catenation occur in a connected succession.
- Events of a Succession succeed each other.
- Each event of a Supersession occurs partially over the preceding one.
- Each event of a Comprehension includes the event following it.
- Each event of a Progression is at least a little beyond the previous one.
- Each event of a Nonregression has started no later than its predecessor.
- Indeterminate events have potentially any relation.
- Concurrent events must overlap at least in part with the events listed adjacent to them.
- Connate events are "born" together.
- Conterminate events share a termination.
- Coeval events last over the same time span.

The names of the groupings are either nouns, all ending in "-ion", or adjectives not ending in "-ion". Nouns are used for the grouping types in which the listing sequence of the subevents affects the temporal relations among them. In these groupings, subevents that are listed earlier also occur earlier in time. Adjectives are used for the grouping types in which the listing sequence does not matter. Each of these groupings is its own converse, so the order of its subevents is immaterial.

The Allen relations for the twelve groupings were chosen for several reasons:

- Of the 8191 non-null Allen relations ($2^{13}$ minus "no temporal relation"), these are among the *easiest to state and understand*, and thus likely to be the most used in practice, and most likely to be used to group events correctly.
- These relations have been *sufficient* to describe the events and situations we have encountered in our validation work (Comprehension, Connate, Conterminate, and Coeval were added for this reason).
- Each of these relations' *composition with itself* is also one of the twelve groupings.
- Each of the relations' composition with itself is a *fixed point* of the composition operator.
- The twelve groupings are a *tractable subclass* of Allen's interval algebra.

As with Allen relations in general, we can use composition to infer the temporal relation between non-adjacent events in a grouping; thus, simple composition properties for the groupings are valuable. If *x, y*, and *z* are consecutively-listed events in grouping *G,* then by the definition of groupings, $x(G)y$ and $y(G)z$. The relation between *x* and *z* is then $x(G \circ G)z$. The relation between *x* and a fourth or later-listed event in the grouping would be the result of additional composition with *G*. Since the result of each grouping's composition with itself is a fixed point of composition, the relation between any two non-adjacent events in a grouping will be simply $(G \circ G)$, the same no matter how far separated the events are within the group. Table 2 gives the relation between non-adjacent events in each grouping.

| Grouping | Relation between non-adjacent events | Relation between any two events |
|---|---|---|
| Separation (p) | Separation (p) | Separation (p) |
| Catenation (m) | Separation (p) | Succession (pm) |
| Succession (pm) | Separation (p) | Succession (pm) |
| Supersession (oFD) | Progression (pmoFD) | Progression (pmoFD) |
| Comprehension (D) | Comprehension (D) | Comprehension (D) |
| Progression (pmoFD) | Progression (pmoFD) | Progression (pmoFD) |
| Nonregression (pmoFDseS) | Nonregression (pmoFDseS) | Nonregression (pmoFDseS) |
| Indeterminate (pmoFDseSdfOMP) | Indeterminate (pmoFDseSdfOMP) | Indeterminate (pmoFDseSdfOMP) |
| Concurrent (oFDseSdfO) | Indeterminate (pmoFDseSdfOMP) | Indeterminate (pmoFDseSdfOMP) |
| Connate (seS) | Connate (seS) | Connate (seS) |
| Conterminate (Fef) | Conterminate (Fef) | Conterminate (Fef) |
| Coeval (e) | Coeval (e) | Coeval (e) |

**Table 2. ScenarioML grouping composition**

For four of the groupings (catenation, succession, supersession, and concurrent), the relation between adjacent and non-adjacent events is not the same. The table also gives the relation between any two events in the grouping, which is simply the union of the adjacent and non-adjacent relations. For the remaining groupings, any two events in the grouping have a single relationship, which we consider ideal.

The twelve groupings are a subclass of the **H** (or Ord-Horn) subalgebra of Allen's interval algebra, and so form a tractable subclass for inference [KJJ03]. Inference of relations between indirectly related events is thus practical.

We mentioned earlier that it is possible to construct a set of Allen relations on events that is unsatisfiable, that is, for which there is no choice of fixed times for the events that would satisfy all the relations. It is not possible to make an unsatisfiable ScenarioML scenario with the language constructs described so far, but *event references* introduce that possibility. An event reference is a tag that refers to an event defined elsewhere, so that the event may be referred to in a grouping that does not contain the event's definition. For clarity of expression, ScenarioML also introduces event relations, which unlike groupings do not define compound events, but instead add an additional temporal constraint on events defined elsewhere. Figure 5 shows a relation from the ScenarioML form of the small plane scenario. The scenario fragment shown in the figure defines a simple event named "enterKRAL" and a relation that constrains it to occur before the event named "flyToKRAL", defined elsewhere in another grouping. This relation expresses the fact that in the scenario, each waypoint must be entered before it can be flown to.

```
    <simpleEvent name="enterKRAL">
     The pilot enters the Riverside airport waypoint
     (KRAL).</simpleEvent>
   </succession>
</nonregression>
<relation temporality="succession">
  <eventReference ref="enterKRAL"/>
  <eventReference ref="flyToKRAL"/>
</relation>
```

**Figure 5. Event reference and relation**

## 5 Operations on scenarios

A goal of the development of ScenarioML is that it not only to be expressive, but also support analyses and operations on the form of the scenario. ScenarioML separates the expression of each event from the expression of the temporal relations between the events. Each simple event is expressed in prose, but the temporal relations are expressed ultimately in interval algebra, and so there is the opportunity for software tool support to automate tedious

work and expand the scope of work that can be attempted. In this section we list and briefly discuss some of the analyses and operations that ScenarioML supports.

The most direct operation that interval algebra supports is *inferring the strongest relation* that can be shown to hold between two events. The interval algebra foundation of ScenarioML and the fact that it is a tractable subclass makes possible manual or automated calculation of that relation for any arbitrary pair of events in a scenario.

A related analysis is the question of whether a scenario is *satisfiable* or not. With the use of event references and temporal relations imposed from outside the tree structure of event groupings, the possibility of unknowingly creating an unsatisfiable scenario is a very real one.

More specific is the question of whether a scenario is *satisfied by specific event timings.* A scenario's meaning seems most fundamentally to be derived from the varieties of real-world events that can be said to be an instance of the scenario. Interval algebra supports the temporal aspect of this analysis.

A general class of operations are the *transformations that preserve meaning*, of which we have identified some and have grounds for expecting many more. Some event groupings can be split and the parts grouped at a higher level, for example a catenation can be divided into two catenations that then become the subevents of a new catenation one level up. This transformation and others analogous to it preserve the meaning of the scenario while altering the form in which it is expressed. Not all grouping can be split and recombined in this way, but some can be if certain additional relations are imposed on its events through references to them.

In the opposite direction, some temporal groupings can be combined while preserving meaning. The question is particularly interesting for groupings of different kinds; no general approach has been identified to date, but there appear to be possibilities.

The use of temporal groups together with additional relations on events through references to them opens the possibility of *inverting* groups and relations. It is possible in some cases to interchange event definition and event reference while exchanging grouping and relation, resulting in an equivalent scenario in a much different form. The operation is complex enough, and indeed the process of verifying that the operation is possible is difficult enough, that tool support is necessary to make this a practical possibility.

## 6 Related work

SMaRT (Scenario Management and Requirements Tool) is a software tool for creating, editing, and analyzing scenarios, developed at North Carolina State University [WAA03]. It supports scenarios with sequential events, hierarchically structured in linear sequence, alternation, and iteration, with episodes for reusing the events of a scenario in other scenarios. It implements and extends the scenario management work of Alspaugh *et al.* [AAB99]. SMaRT does not have a public file format that can be used to combine its functionality with that of other tools.

SDML (Scenario Definition Markup Language) [DPI02, DPI03] is an XML scenario language that expresses the use case forms of Cockburn and Rumbaugh *et al.* [Coc97, RJB99]. A basic SDML scenario is a linear sequence of interactions, with possible branch points listing alternative variant subsequences. Other scenarios can be referenced unconditionally by *inclusion* or conditionally by *extension*. More complex structure is obtained by iterating a scenario and by composing two scenarios in parallel or in mutual exclusion [DPI04]. ScenarioML includes equivalents of SDML structuring elements except for mutual exclusion. SDML events are expressed as actors and actions, each from a list defined as part of the use case, as in Alspaugh *et al.*'s Integrated Scenario Management Strategy and the SMaRT system [AAB99, WAA03]. A graphical editor has been implemented for SDML scenarios [DPI04]. The SDML schema does not appear to be public at this writing, so that SDML cannot be used to combine the functionality of scenario tools.

*Interval scripts* [PMB97, PB03] are low level interactive scripts specifying the temporal relations between time intervals using Allen's interval algebra. They are intended for describing and managing interactivity in immersive environments. Interval scripts use the full Allen's interval algebra rather than a subalgebra as ScenarioML does, and express relations between pairs of events. Inference is approximated by projecting each relation onto combinations of *past, now,* and *future* and computing a conservative approximation to inference in this algebra, for which strict inference is still NP-hard. Although interval scripts have the same temporal basis as ScenarioML scenarios, there are few other similarities.

## 7 Lessons learned and future work

We have seen that scenarios can be extended with more general temporal relations in a way that gives additional expressiveness while still remaining relatively easy to use. Having this additional expressive power has led us to write scenarios describing situations that would not have seemed appropriate for scenarios before, one example being the combination of tasks that the pilot of a small plane must address. The additional expressive power has also led us to produce scenarios that are more detailed and complete than would have been practical in the past.

The existence of an XML-based language for expressing scenarios has led to unusual uses of them that would otherwise be impractical or not thought of, notably a collaboration with virtual character researchers, a research project on specific aspects of pilot cognitive workload, and investigation of novel approaches to testing. As is often the case, the availability of a new or extended tool leads to the discovery of new uses for it.

The availability of ScenarioML as a common language for expressing scenarios has made tool support for work with scenarios much more of a practical possibility. The existence of a large body of publicly-available packages for Java and XML allows development of scenario tools on a shorter time scale and with fewer developers, and allows quick prototyping of experimental scenario.

Perhaps most interesting is that the people that have used ScenarioML to date have been intrigued by it. There is reason to believe that ScenarioML can be useful and effective as a way of expressing and working with scenarios, but beyond those utilitarian aspects it seems clear that users enjoy seeing what they can do with ScenarioML.

There are several directions of future work for ScenarioML.

A fundamental test of a scenario form is the extent to which it helps to define scenarios, check scenarios, and work with concurrent events within a single scenario and across two or more scenarios. Preliminary results for ScenarioML are encouraging, but more investigation is needed to support comparison with other scenario forms and a strong argument in favor of ScenarioML.

Temporal logics of one form or another are the most common formal approach for stating temporal relations among events, although they are rarely used in conjunction with scenarios. Temporal logic has disadvantages in practice: its users must have substantial training, tool support is essential, and it is somewhat cumbersome for expressing arbitrary relations in software requirements. ScenarioML offers a possible alternative approach which may be easier to use and advantageous in some or many requirements contexts. We plan to investigate the extent to which one might be used instead of the other, and conduct case studies comparing them.

We have seen that ScenarioML can be effective in representing scenarios, and in recording temporal relations that otherwise would be overlooked. Certain patterns of expression seem to recur, but it is not clear whether these are inherent to ScenarioML or a consequence of the way it has been presented to users. It is plausible that some patterns of expression will be more effective, and (for example) better support certain uses of scenarios or be better supported by software tools, but to date there is not enough information to do more than hypothesize. The fact that ScenarioML has a form that can be analyzed by a tool opens the possibility of using the accumulating collections of scenarios as data to study patterns of use.

A particularly intriguing possibility is the matching of scenarios against streams of events, to identify occurrences of specific scenarios at particular points in the stream. This is a possible basis for a more extensive scenario semantics, built upon event types and subsumption among them. A related question is matching one scenario against another, and comparing two scenarios for similarity. ScenarioML scenarios have much more varied structure and semantics than linear-sequence scenarios, so comparison is much more challenging. Work along these lines is underway in connection with autonomous character research here at UCI.

Tool support is essential to make a fairly complex notation like ScenarioML useful. Work on a graphical editor for ScenarioML is in its preliminary stages. As part of that work, the issue of meaning-preserving transformations on ScenarioML scenarios is significant as a source for more effective and powerful tool support in an editor and in general. Such transformations can be used to improve the form in which a scenario is expressed, without changing the meaning the scenario expresses. This leads to the question of what forms are more advantageous, and in what contexts, and connections with effective patterns of expression.

A related area of research is the presentation of the large amount of information in a ScenarioML scenario in a form that is manageable and understandable. The XML form of a ScenarioML scenario becomes progressively more difficult to understand as the scope of the scenario increases, as do notations in general. We have investigated effective presentation forms for scenarios, one of which is shown in Figure 7; there, relations that are primarily se-

quential are expressed *down* the page, while relations that are primarily concurrent are expressed *across* the page. This form is a useful one but more research is needed.

ScenarioML has a number of constructs not directly connected to temporal relations, and these have not been as fully explored as its temporal groupings and relations. Scenario management is still a substantial unsolved problem, and one approach (seen in the Use Case community among others) is to make use of language constructs to address scenario organization, management, and re-use. This has been an ongoing area of research and the availability of ScenarioML as a basis for work in this area is a considerable benefit.

Specification-based testing is a promising area in which major unsolved problems remain. The possibility of using scenarios as the specification from which tests are derived has some intriguing aspects: scenarios are closer to user-expressed goals than the forms from which specification-based tests have been derived in the past, and the scale of scenarios offers potential benefits in addressing testing beyond the unit level. ScenarioML forms a basis for investigating these possibilities.

## 8 Acknowledgements

The author thanks his colleagues in the Rosetea research group and the Informatics Department of UC Irvine, and the anonymous reviewers of an earlier version.

## 9 References

[Ale04] I. F. Alexander. Scenarios in System Development. In I. F. Alexander and N. Maiden, editors, *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*, pp. 3-24. John Wiley & Sons, 2004.

[All83] J. F. Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM* 26(11):832-843, Nov. 1983.

[AAB99] T. A. Alspaugh, A. I. Antón, T. Barnes, and B. W. Mott. An Integrated Scenario Management Strategy. In *Fourth IEEE International Symposium on Requirements Engineering (RE'99)*, pp. 142-149. 1999.

[Als05] T. A. Alspaugh. Software Support for Calculations in Allen's Interval Algebra. Institute for Software Research Technical Report UCI-ISR-05-02, University of California, Irvine, February 2005.

[BBL04] F. Basanieri, A. Bertolino, G. Lombardi, G. Nucera, E. Marchetti, and A. Ribolini, "Cow_suite: A UML-Based Tool for Test-Suite Planning and Derivation," *ERCIM News* (58) pp. 30-32, July 2004.

[BFJ92] K. Benner, M.S. Feather, W.L. Johnson, and L. Zorman. Utilizing Scenarios in the Software Development Process. In *IFIP Working Group 8.1 Working Conference on Information Systems Development Processes*, pp. 694-751. 1992.

[Coc97] A. Cockburn. Using Goal-Based Use Cases. *Journal of Object-Oriented Programming*, 10(7):56-62. 1997.

[Coc00] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley Longman Publishing Co., Inc., 2000.

[DPI02] G. Della Penna, B. Intrigila, A. R. Laurenzi, and S. Orefice. An XML Definition Language for Software System Specification. In *6th World Multi Conference on Systemics, Cybernetics and Informatics*, pp. 179-190, 2002.

[DPI03] G. Della Penna, B. Intrigila, A. R. Laurenzi, and S. Orefice. An XML Definition Language to Support Scenario-Based Requirements Engineering. *International Journal of Software Engineering and Knowledge Engineering*, 13(3):237–256, June 2003.

[DPI04] G. Della Penna, B. Intrigila, A. R. Laurenzi, and S. Orefice. A Methodology for Scenario Development. In *16th International Conference on Software Engineering and Knowledge Engineering*, pp. 7-12, 2004.

[Hau04] P. Haumer. Use Case-Based Software Development. In I. F. Alexander and N. Maiden, editors, *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*, pp. 237–264. John Wiley & Sons, 2004.

[KJJ03] A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *Journal of the ACM* 50(5):591-640. Sep. 2003.

[JCJ92] I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press, 1992.

[LW98] A. van Lamsweerde and L. Willemet. Inferring Declarative Requirements Specifications from Operational Scenarios. *IEEE Transactions on Software Engineering* 24(12):1089-1114. Dec. 1998.

[MMM98] N. A. M. Maiden, S. Minocha, K. Manning, and M. Ryan. CREWS-SAVRE: Systematic Scenario Generation and Use. In *Proceedings: 3rd International Conference on Requirements Engineering*, pp. 148-155. 1998.

[PB03] C. Pinhanez and A. Bobick. Interval Scripts: A Programming Paradigm for Interactive Environments and Agents. *Pervasive and Ubiquitous Computing*, 7(1):1-21. 2003

[PMB97] C. S. Pinhanez, K. Mase, and A. Bobick. Interval scripts: a design paradigm for story-based interactive systems. In *CHI '97: SIGCHI Conference on Human Factors in Computing Systems*, pp. 287-294, 1997.

[Rob04] S. Robertson. Scenarios in Requirements Discovery. In I. F. Alexander and N. Maiden, editors, *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*, pp. 3-24. John Wiley & Sons, 2004.

[RJB99] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison Wesley Longman, 1999

[RSBA98] C. Rolland, C. Souveyet, and C. Ben Achour. Guiding Goal Modeling Using Scenarios. *IEEE Transactions on Software Engineering*, 24(12):1055–1071, Dec. 1998.

[Sut03] A. Sutcliffe. Scenario-Based Requirements Engineering. In *11th IEEE International Conference on Requirements Engineering (RE '03),* pp. 320-329, 2003.

[WAA03] W. Stufflebeam, A. I. Antón, and T. A. Alspaugh. SMaRT — Scenario Management and Requirements Tool. In *11th IEEE International Conference on Requirements Engineering (RE '03)*, pp. 351. 2003.

[VKB89] M. Vilain, H. Kautz, Henry and P. van Beek, Constraint Propagation Algorithms for Temporal Reasoning: A revised report. In D. S. Weld and J. de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems*, pp. 373-381. 1989.

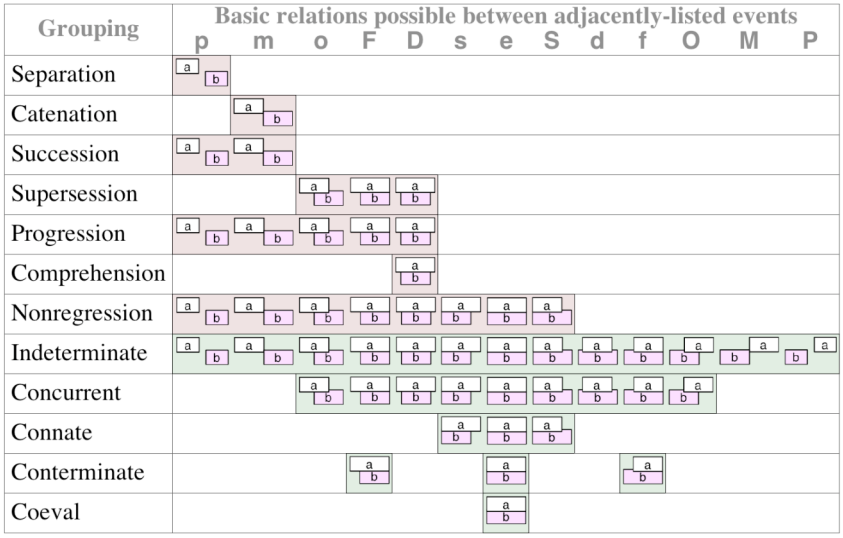



**Figure 6. ScenarioML temporal groupings**

*NONREGRESSION*

1. *SUCCESSION*
   1. The pilot enters the Palomar airpoint waypoint (KCRQ) into the GPS device. [*enterKCRQ*]
   2. The pilot enters the Oceanside VOR waypoint (OCN). [*enterOCN*]
   3. The pilot enters the KRAUZ intersection waypoint. [*enterKRAUZ*]
   4. The pilot enters the POM12 waypoint. [*enterPOM12*]
   5. The pilot enters the Paradise VOR waypoint (PDZ). [*enterPDZ*]
   6. The pilot enters the Riverside airport waypoint (KRAL). [*enterKRAL*]
2. *COMPREHENSION*

I. *SUCCESSION*

1. The pilot takes off from Palomar Airport (KCRQ). [*takeoffKCRQ*]
2. The pilot flies to the Oceanside VOR waypoint "OCN". [*flyToOCN*]
3. *ALTERNATION* (At El Toro)

   A. *SUCCESSION* (Following the flight plan)
      1. The pilot flies along the V23 ("Victor 23") IFR route to the KRAUZ intersection waypoint. [*flyToKRAUZ*]
      2. The pilot flies along V363 to the POM12 intersection of V363 and V8. [*flyToPOM12*]

   B. *SUCCESSION* (Flight plan amended by ATC)
      1. ATC radios instructing pilot to turn right 20 degrees and intercept the V8 to Paradise.
      2. The pilot turns right 20 degrees.
      3. When the GPS shows the plane has almost reached the original route again, the pilot turns right to intercept the V8 route.

4. The pilot flies along V8 to the Paradise VOR waypoint "PDZ". [*flyToPDZ*]
5. The pilot flies to the Riverside airport waypoint "KRAL", then follows the tower's instruction for the approach and landing. [*flyToKRAL*]

II. *CATENATION*

1. The plane climbs high enough to be in the TIS radar service area.
2. *COMPREHENSION*

I. The plane flies high enough to be in the TIS radar service area.

II. *ITERATION* * *times*

1. *SUCCESSION*
   1. Another plane comes within approximately 2 NM and 1000 feet in altitude.
   2. *SUCCESSION*
      1. The TIS (Traffic Information System) identifies the plane and announces "Traffic, Traffic" through the pilot's headphones.
      2. The pilot punches the traffic button ("TRFC") on the instrument to display where the other plane is, relative to the pilot's.
      3. The TIS highlights the display symbol identifying the other plane.
      4. The pilot adjusts the scale of the display to 2 NM to show the other plane's location most clearly.
   3. *SUCCESSION*
      1. The pilot analyzes the situation to decide how best to avoid the other plane.
      2. The pilot maneuvers to avoid the other plane.
   4. The pilot resumes normal navigation.

3. The plane descends in altitude below the TIS radar service area.

*RELATION 1. SUCCESSION:*
- *EVENT REFERENCE* [enterKCRQ]
- *EVENT REFERENCE* [takeoffKCRQ]

*RELATION 2. SUCCESSION:*
- *EVENT REFERENCE* [enterOCN]
- *EVENT REFERENCE* [flyToOCN]

*RELATION 3. SUCCESSION:*
- *EVENT REFERENCE* [enterKRAUZ]
- *EVENT REFERENCE* [flyToKRAUZ]

**Figure 7. A ScenarioML scenario fragment, presented in HTML**