When Things Go Wrong: Predicting Failure in Multi-Agent Systems

Stephen Fickas, Rob Helm and Martin Feather

CIS-TR-91-15

Department of Computer and Information Science
University of Oregon

When Things Go Wrong: Predicting Failure in Multi-Agent Systems

Stephen Fickas and Rob Helm[1]

Martin Feather[2]

1. Our expertise

Our expertise is in the area of AI&SE. Our early research focus was on mapping from formal specifications to efficient implementations using a transformational approach. One system that resulted from this work was called Glitter. Glitter can be viewed as an interactive problem solver for exploring transformational implementations [Fickas, 1985].

More recently, we have become interested in the mapping from requirements to specification, i.e., the process that precedes that addressed by Glitter. In particular, in using Glitter we found that coming up with the initial formal spec that Glitter expects as input was just as hard as mapping the spec to an implementation. Both Fickas, through the Kate group, and Feather, within the ARIES project [Johnson&Harris, 1990], have now focused their research on the process of building formal specifications.

A note for those outside of the software engineering area: one can view our use of a formal specification as equivalent to the use of a behavioral model in other domains. The same concerns with model synthesis, analysis, simulation and explanation crop up in our work. Expanding slightly, one can overlay notions of function-structure-behavior onto our representation of formal requirements (function), specification (structure), and operationality (behavior).

2. Our current interests

A current interest of the Kate and Aries groups is the specification of multi-agent systems. Related to this workshop, we have been studying multi-agent systems that have an intelligent, real time control aspect, e.g., air traffic control, ground transportation systems, nuclear reactors. From these studies we have devised a theory of multi-agent design [Feather, 1987],[Fickas&Helm, 1991]. We will not describe our theory in detail here, but we will lay out its major pieces:

The specification/model that we analyze and manipulate consists of two components[3]:

1. a *generative* part denoting, for each agent, the capabilities of that agent (e.g., pilot - enter or exit air space; controller - assign runway). In our model, this takes the form of a discrete event language that can be viewed, alternatively, as a subset of Gist [London&Feather, 1982] or a superset of Petri Nets.

1. The first two authors are with the Computer Science Department at the University of Oregon. Questions about the paper can be directed to fickas@cs.uoregon.edu.
2. The third author is with the USC Information Sciences Institute.
3. Our motivation for adopting this style of specification derives from [Balzer&Goldman, 1979].

2. a *constraining* part consisting of a set of 'goals', i.e., constraints on behavior. For the purpose of describing composite systems, it is often convenient to express goals in terms of system-wide properties, regardless of how that system is decomposed into agents (e.g., two planes shall never share the same runway). In our model, goals are represented in a style of temporal logic roughly similar to that of the ERAE language [Dubois&Hagelstein, 1989] and the distributed-system logic of [Castro, 1990].

Our design theory identifies a set of processes performed on a specification:

Identification of problematic behaviors. We rely on a scenario/plan generator [Anderson&Fickas, 1989] to find counterexamples, i.e., behaviors that demonstrate a goal is not being met by the current model of agent cooperation. These counterexamples have two roles: 1) they let us know if a goal is achieved in the current model, and if not, 2) they act as a guide on how the goal or model can be changed to bring achievement [Fickas&Helm, 1991], [Downing&Fickas, 1991].

Requirement modification. In our theory, requirements are represented by goals. These goals may be modified in light of problematic behavior. There are two open research problems here. First, can we characterize a finite and useful set of goal modifications (e.g., goal weakening, divide-and-conquer)? Second, given two competing goal transformations, we must define the evaluation criteria that selects among them - without such criteria we would always be free simply to weaken an offending goal into non-existence. As noted in [Herlihy&Wing, 1991], a theory of *the cost* of strengthening and weakening constraints is an open problem.

Assignment of responsibility. This is the alternative "design move" to goal modification. Informally, only those agents responsible for a goal are expected to limit their own behavior to ensure satisfaction of that goal. As with goal modification, there are two questions of interest: 1) can we usefully characterize the process of responsibility assignment, and 2) what criteria is available for choosing among alternative assignments, e.g., systems using all human agents, systems using a mixture of human and software agents, systems using strictly software and hardware agents. Both [Feather, 1987] and [Dubois&Hagelstein, 1989] have taken up the first question. The second question has become a major focus of our current research. In particular, it is one of the questions that we bring to the workshop.

The outcome of design is a specification of the agents that will be involved in the system, and the problem solving protocol that will be used to coordinate their actions to meet system goals. Agents, themselves, will include specifications of the following components: 1) what goals they are responsible for, 2) what abilities they must have to carry out those responsibilities, 2) how they must limit their behavior to meet their goals, and 4) what interface they must have to coordinate with other agents. For the latter, we stop well short of specifying the level of interface detail typically found in the HCI and engineering fields. Thus, while we may specify that two agents must agree on a signal between themselves, we do not worry about the type of signal (e.g., voice, graphical, mechanical, electronic, etc.).

3. Classes of failure in multi-agent architectures

Our theory of multi-agent design has the need for an explicit evaluation component (refer to *assignment of responsibility* in the previous section). Our efforts to date have centered on general notions of overall goal achievement (i.e., have all the system goals been covered by the agents in-

volved), the reliability of agents carrying out their assigned responsibilities *and* avoiding acts that are detrimental to the system (e.g., agents acting as "loose cannons"), and the soundness of the inter-agent problem-solving protocol (i.e., matching problem-solving interaction with the problem at hand). To illustrate the source of these criteria, we list a case sampler of what we term evaluation failures taken from two of the multi-agent domains we have studied, train systems [Shaw, 1961][4] and the Three Mile Island (TMI) plant [Kemeny, et al 1979]. We have broken these cases into three classes: those involving problems with goal coverage, those involving single agent failure (the proverbial weak link), and those involving the broader problems of inter-agent cooperation.

3.1 Role assignment failures

Referring to our design theory introduced in section 2, the responsibility of achieving system-wide goals is assigned to agents. This assignment process can go awry in two basic ways:

Under-assigned roles (who's watching the store). It is difficult to analyze the myriad behavior of complex systems, and how they relate to the goals of the system. Actions necessary in meeting a goal may be left unassigned, i.e., no agents are made responsible for the goal (and hence, the goal achievement act). In early train systems, the conductor of a stalled train was made responsible for warning trains approaching from the rear to slow down, but no one was assigned the responsibility of warning front approaching trains - trains were scheduled so that head-on collisions were impossible. Later scheduling policy allowed interleaved, bi-directional travel, but still no one was assigned to warn front approaching trains. A spate of head-on collisions lead to the introduction of a new agent assigned to warn from the front. At TMI, while there were county agents assigned to carry out evacuation, there was no TMI agent assigned the responsibility of notifying the county of a disaster requiring evacuation. An unexpected agent, the press, eventually carried the message.

Over-assigned roles (too many cooks). A goal can be over-assigned in the sense of having too many agents *redundantly* responsible for it[5]. For example, at TMI there were judged to be too many agents responsible for reporting the same information. The resulting cacophony of reports, sometimes in conflict, led to chaos[6]. Taking the converse, in some train systems over-assigned roles led to no reaction. For instance, both the conductor and the engineer were assigned the responsibility of looking for special orders (typically, orders to wait at a siding). Cases arose where neither looked because each thought the other would check.

3.2 Single agent failures

Given the notion of responsibility from our design theory, we are led to the notion of irresponsible behavior[7]: agents that fail to follow certain roles they have been assigned to achieve some goal. This comes in at least four flavors:

---

4. We have also looked at multi-agent failures in the related domain of air traffic control [Nance, 1986], and have just as readily found ATC cases to fill our evaluation categories. However, train failures are slightly easier to describe and have a longer history, i.e., more chances to go wrong and in more ways.
5. There is a related problem of having too many agents *fractionally* responsible for a goal - see section 3.3.
6. Of course, some forms of hardware safety systems rely on multiple, redundant components, and a voting or averaging scheme to integrate the output of each.
7. There is not necessarily a pejorative sense here - we mean literally the behavior of agents that does not match their assigned responsibilities.

Responsibility overload (underpowered agents). An agent may be assigned 1) a single responsibility that it cannot fulfill, or 2) a set of responsibilities, each of which it can handle in isolation, but not in accumulation. As an example of the former, conductors on a stalled train were responsible for warning approaching trains of the break down. Conductors were expected to *walk* out to a safe stopping distance and set their flags. Unfortunately, trains were sometimes scheduled too close together to give the conductor time to run, let alone walk, the required distance. This was the cause of a number of train collisions. As an example of the second overloading problem, sometimes station operators on a busy line were assigned multiple responsibilities: set the stop signal, clear the stop signal, communicate with adjacent operators on track status. The agents got caught up in other duties and did not set the clear signal in time - trains backed up. The same station operators let other duties interfere with setting the stop signal in time - trains collided[8]. While one would expect that priorities could be specified among responsibilities when an agent becomes overloaded, for instance trading off disaster for inconvenience, it appears to have been impossible to provide such an ordering for every conceivable problem in the train management domain. Without an explicit priority system, overloaded agents are left to infer their own (see "outguessing" below).

Information overload. The TMI accident was blamed on both too much information being presented to operators, and at too low a level to be useful[9]. Operators could not determine what was important and what was of secondary interest. One of our colleagues, David Novick, is studying air traffic control protocols between controller and pilot from a similar view [Novick, 1990].

Outguessing the system (overpowered agents). Intelligent agents may come to believe that they know the structure and function of a system, *and* its current state. This may lead them to infer many strange and wonderful things. On the positive side, correct inferences can sometimes avert disaster, e.g., a station operator manually flagging down a train given a broken stop signal, a TMI operator inferring that a warning gauge is non-functional, and shutting down the reactor. On the negative side, cases abound where the agent's view of 1) the system structure is incorrect, 2) the system function is incorrect, or 3) the current system state is incorrect. Each incorrect view can lead to its own kind of disaster, e.g., not carrying out an *assigned* role because it does not appear to be necessary (or of low priority), carrying out an *unassigned* role because it does seem necessary.

Conflicting interests. In early train systems, an engineer was given the dual (and interfering) responsibilities of maintaining safety and making speedy progress. Further, the railroad big wigs decided to fire engineers who had a weak *progress* record. Engineers began to disobey speed limits, warning signals and even direct orders to stop, leading to trains colliding, derailing, and running off open draw bridges[10]. At TMI, the operators seemed to have a more noble conflict: trying to avoid both over-filling and under-filling the reactor chamber with water. They compromised, disastrously, on the under-filling side.

---

8. Another twist with the same result is an agent playing roles in more than one system, e.g., a train station operator also acting as the Western Union telegraph representative for an area.
9. There was also a critique of the actual control room interface, i.e., the gauges, lights, buzzers, but as noted earlier, this is beyond our scope.
10. There is a social/organizational component to this case that may imply that we are addressing more than we actually are. We have not tackled the psychology of organizations, inter-personal relationships, cognitive engineering, etc. Critiques of multi-agent failures directly tied to such fields are beyond our current evaluation model.

3.3 Multiple-agent (or protocol) failures

A problem solving protocol, in our model, is the *dialog* (nee communication) and *actions* carried out among agents to solve some common problem. We are not the first to study such problem-solving protocols from a formal sense: [Davis&Smith, 1983] discuss a bid-and-award protocol for (dynamically) choosing an agent to work on a problem; [Bond, 1990] describes a multi-agent, negotiation-based protocol. However, we are less interested in devising new protocols, and more interested in the general question of which protocol architectures are best suited to which multi-agent problems. Taking the negative side, when will a given protocol lead to failure for a given problem.

Fragmentation of responsibilities. In contrast to overloading a single agent with too much responsibility, it is possible to divide a responsibility among too many agents, so that overly much coordination among a number of agents is needed to achieve that responsibility. The volume of communication that this necessitates may offer many opportunities for error. For instance, early communication protocol for signalling a train T to stop at a distant station S went through the following steps: 1) the dispatcher contacted an operator at a station *before* S, and verbally give the order to the operator, 2) the operator would stop T, and read the order to the conductor, 3) the conductor would pass the order along to the engineer, and 4) the engineer would stop (?) at S. Unfortunately, there are documented cases of failure of each and every step in this sequence, many leading to a train collision.

Poor risk analysis. There may be mismatches between 1) the reliability of a protocol, and 2) the outcome of miscommunication. In particular, some types of inter-agent dialogs are not well matched to the larger control acts that they service. For example, many of the early multi-agent train-signaling protocols were of a rather risky nature, using the wrong defaults, easily confused control signals, non-failsafe mechanisms, ungraceful-degradation, etc. It seems clear that they were wholly lax or inappropriate for the act they were based on: control of hard-to-stop trains on potential collision courses. Later changes added more elaborate dialogs (e.g., formal confirmation procedures) and safeguards (e.g., automatic derailers).

Hand-off errors (shifting responsibilities). Responsibility can shift from one agent to another. For example, some fragmentation or sharing of responsibility is typically necessary among agents. A standard approach is for agent A to be responsible for goal G until event E, and agent B to be responsible for G from E onwards[11]. For instance, when a train crosses from block A into block B, a responsibility shift should occur. In numerous cases, this shift was not made cleanly, leaving a train in an unmonitored state. In a more bizarre example, a scheduled maintenance of the signals on a block of track called for the replacement of one of the signals. The new (but yet unworking) signal was placed *in front of* the existing (working) signal *in the clear position*. Before the full hand-off could take place, a train was caught in this ambiguous situation, read the wrong signal, and a collision occurred shortly thereafter.

A given protocol may specify hand-offs for emergency situations as well as those occurring during normal operation, e.g., a brakeman will act as back-up and take the conductor's job of flagging a stalled train if the conductor is unable to carry out the flagging responsibility. The prevalent problem here seems to be determining if the conductor has relinquished (or should relinquish) this responsibility. In the worst cases, the brakeman was left to infer this information, and sometimes

11. Instead of an event E, division may be on a time T, e.g. a shift-change.

did not, at least in a timely fashion.

3.4 Cross products

As one might expect, failures are often caused by combinations of the above problems. For example, at TMI the inter-agent communication protocol was flawed, leading to lack of information for the operators. Given the scarcity of information, operators made inferences to fill in what they needed. These inferences were wrong, leading in part to the disaster. Using a train example, an unnamed conductor C of a stalled train had the responsibility of warning approaching trains. However, C reasoned 1) that no trains were scheduled for several hours, and besides, 2) the speed limit on the block was 10 miles an hour. These two assumptions in accumulation, at least as later reported by C, led C to infer that he did not need to fulfill his warning role. Both assumptions were wrong: 1) a "special" was scheduled to arrive in minutes (and did), and 2) the speed limit had just recently been increased. By the time the special saw the stalled train, it was unable to stop before collision.

4. Status

Our ultimate goal is an active design assistant that will help generate formal specifications of multi-agent architectures that are correct by construction. From this perspective, we would like to critique any proposed multi-agent architecture that was based on unfilled roles, or an unsuitable problem solving protocol, or a high propensity for agents to try (or need) to outguess the system, or too many agents involved (either relatively or absolutely), etc. We would like all this in a formal package, one we could integrate with the design theory outlined in section 2. We have some small pieces of this package built, e.g., [Fickas&Helm, 1991] discusses a scenario generator for testing goal coverage, [Downing&Fickas, 1991] discusses a *qualitative* modeling tool for critiquing multi-agent resource-management systems, [Fickas&Nagarajan, 1988] discusses the use of a case-based tool to use past cases of failure, much like those of the last section, to critique specifications in a resource-management domain. However, they are just a small step towards what we see as needed[12], and further, their integration into a single evaluation tool remains an open problem.

5. References

[Balzer&Goldman, 1979] Balzer, R., Goldman, N., Principles of good software specification and their implications for specification languages, *Specification of Reliable Software*, IEEE Press, 1979

[Castro, 1990] Castro, J., Distributed System Specification using a temporal-causal framework (Ph. D. thesis), Imperial College of Science and Technology and Medicine, University of London, Department of Computing, 1990.

[Davis and Smith, 1983] Davis, R. and Smith, R, Negotiation as a metaphor for distributed problem solving, *Artificial Intelligence 20*:63-109, 1983 (reprinted in Bond, A. and Gasser, L., An analysis of problems and research in AI, in Bond, A. and Gasser, L. (eds.) *Readings in Distributed Artificial Intelligence*, San Mateo, CA: Morgan Kauffman, 1988).

12. See, for instance, the compendium of multi-agent failures in SIGSOFT, January 1991.

[Downing&Fickas, 1991] Downing, K., Fickas, S., A Qualitative Modeling Tool for Specification Criticism, *Conceptual Modelling, Databases, and CASE: An Integrated View of Information Systems Development*, Peri Loucopoulos (ed), Ablex, 1991

[Dubois&Hagelstein, 1989] Dubois, E., Hagelstein, J., A Logic of Action for Supporting Goal-Oriented Elaborations of Requirements, In *Proceedings of the Fifth International Workshop on Software Specification and Design*, Pittsburgh, Pennsylvania, May 1989

[Feather, 1987] Feather, M., Language support for the specification and development of composite systems, *ACM Transactions on Programming Languages and Systems*, Volume 9, Number 2, April 1987

[Fickas, 1985] Fickas, S., Automating the transformational development of software, In *IEEE Transactions on Software Engineering*, Vol. 11, No. 11 Nov. 1985

[Fickas&Helm, 1991] Fickas, S., Helm, R., Acting responsibly: reasoning about agents in a multi-agent system, TR-91-02, CS Dept., U of Oregon, Eugene, Or., 97403, February 1991

[Fickas&Nagarajan, 1988] Fickas, S., Nagarajan, P., Being suspicious: critiquing problem specifications, In *Proceedings of the 1988 AAAI Conference*, Minneapolis, 1988

[Herlihy&Wing, 1991] Herlihy, M., Wing, J., Specifying graceful degradation, *IEEE Transactions on Parallel and Distributed Systems*, January 1991

[Johnson&Harris, 1990] Johnson, L., Harris, D., The ARIES Project, *Proceedings of the 5th Annual RADC Knowledge-Based Software Assistant (KBSA) Conference*, Liverpool, NY, pages 121-131, September 1990

[Kemeny, et al 1979] Kemeny, J. et al, The President's Commission on the Accident at Three Mile Island, Pergamon Press, 1979

[Novick, 1990] Novick, D. G., Modeling belief and action in a multi-agent system, *Conference on AI, Simulation and Planning in High-Autonomy Systems*, Tucson, AZ, March, 1990

[Shaw, 1961] Shaw, Robert B. *Down Brakes: a History of Railroad Accidents, Safety Precautions, and Operating Practices in the United States of America,* London, United Kingdom: P. R. Macmillan Limited, 1961.

[Woods&Roth, 1988] Woods, D., Roth, E., Cognitive systems engineering, *Handbook of HCI*, Helander (ed), North-Holland, 1988