Risk management in concurrent engineering in presence of intelligent agents

Taner Bilgiç
Enterprise Integration Laboratory
Department of Mechanical and Industrial Engineering
University of Toronto
Toronto, Ontario, M5S 3G9 Canada
e-mail: taner@ie.utoronto.ca
www.ie.utoronto.ca/EIL/

Draft
November 1996
Revised May 1997

Contents

1 Introduction

In this draft we describe how a specific facet of systems engineering (risks) can be managed for large, concurrent engineering projects. The conceptualization presented in this draft equally applies to to other facets of the systems engineering like cost, quality, schedule etc.

We all have an intuitive understanding of the notion of risk (or "problems" in a looser sense) within the life-cycle of a project[1]. These can be broadly categorized as:

- events that cause a delay in the project,
- events that cause cost overruns,
- events that cause deficiencies in technological performance.

In its most generic form, risks are unexpected events that cause constraint (temporal, resource, performance) violations.

One thing to notice about the definition of risk as defined above is its *causal* nature. Then, identifying risks amount to identifying variables and the causal relationships between them.

Although this level of generality is too abstract to be useful in practice, it gives important clues as to what types of *ontological commitments* one has to make to deal with risks.

Risk management has three stages to it:

1. Risk assessment
2. Risk analysis
3. Risk handling

Risk assessment is the primary function on which risk management is built and it is the stage where the notion of risk is the most vulnerable. This stage is contaminated with the debate on whether probabilities are subjective or objective. The debate is vital to risk management since the concept of risk management may change drastically depending on the adopted approach. With the spread of Bayesian statistics we are learning how to reconcile the two notions of probability as opposed to sticking to one representation.

Risk analysis is where tools are applied to structure the risk information. There are various approaches to risk analysis. Williams (1995) provides a detailed survey of risk management for large projects in which he reviews heavily quantitative analysis tools stemming from the objective perception of probabilities. Only very few qualitative tools are reported in the literature.

Risk handling is a scheme in which one can perform risk analysis continuously for an on going project. The concept is in tune with control:

> Managing that is based on identification and control of those areas and events in the system engineering life cycle that have the potential for causing unwanted change in either the process or the product. (Sage 1992)

There has been matrix management methods as well as rule-based methods proposed in the literature for this purpose. Possible risk handling strategies are:

Avoidance: Avoid the potential failure consequence and/or its probability by planning around them, where possible.

[1] We are thinking of projects that involve services of many engineers to build an electro-mechanical artifact.

Prevention: Continually sensing the condition of a program and developing options and fall back positions to permit alternative lower risk solutions.

Assumption: Acknowledgment of the existence of the risk but a decision to accept the consequences if failure occurs.

Transfer: To transfer risk from your organization to another, e.g., sub-contractor.

1.1 Objective

We propose to develop a technology for managing risks in large engineering projects which require the interaction of many human and/or software agents. This technology should be able to *dynamically* control risk in projects.

1.2 Requirements

The first requirement for developing this technology is a rich knowledge representation about the product, project and the organization.

There are different types of risk in different stages in the design. For example in the contractual stage the risks usually stem from differences in the interpretation of requirements between the customer and the manufacturer. Whereas in the conceptual design stage the risks may arise due to:

- wrong conceptualization due to missing/inadequate requirements,
- delay in approval of submitted designs,
- delay due to miscommunication and lack of coordination.

A risk management technology should be able to *acquire*, *represent* and *reason* on knowledge about different types of risk.

The technology should be flexible enough to enable the user to define her own risk handling strategy.

1.3 Proposal

To meet the requirements posed above for a risk management tool, we propose to use

- TOVE ontologies for rich knowledge representation about the products, project, and the organization.
- Bayesian Networks (Pearl 1988, Jensen 1996) as the main risk analysis tool. By the nature of their construction, Bayesian Networks capture the causal relations between the risk variables.
- Use a rule-based system to capture risk handling strategies.

Figure 1 summarizes this conceptualization.

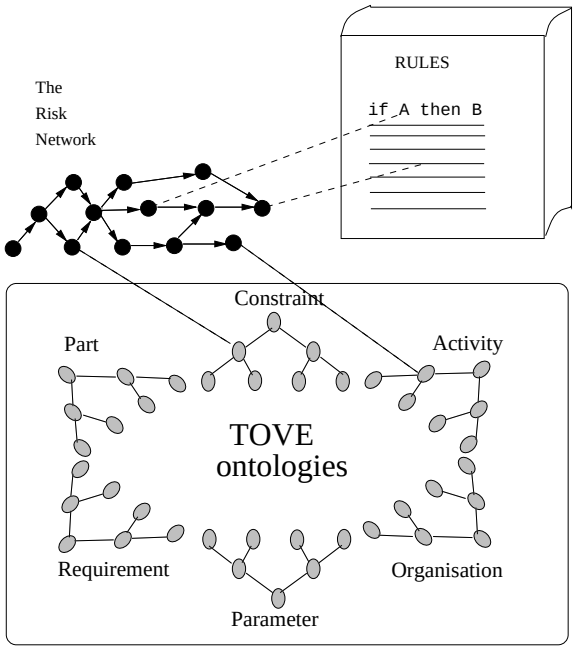

Figure 1: Conceptualization of the risk management technology

1.4 Working principle

The working principle for this conceptualization can be summarized as follows: a Bayesian Network captures the structure and values of risk variables in a dependency model. This model can be created in various ways:

- Human agents define it using a user interface,
- The structure can be extracted from the task structures each task being a node,
- The task structures can be translated to TOVE activity and constraint ontology and the network can be compiled from there.

There are *symptom patterns* embedded with the risk network to trigger events in the rule-based system. These symptom patterns are typically thresholds set on various nodes (or subsets of nodes) in the network. More specifically, these are events for which posterior probabilities are going to be calculated based on observations. When the probabilities of those events exceed a certain threshold, the rule-based system is going to be activated.

The rule-based system contains rules as to what action to take when a certain symptom is detected. These can be as simple as notifying related agents or as sophisticated as imposing constraints on other agents. The risk control strategies are encoded as symptom patterns around the risk network and rules corresponding to them in a rule-base.

1.5 Discussion

The advantage of using rich knowledge representation schemes, with ontological commitments to minimize ambiguity is well understood. The ontological approach is promising in that sense and we adopt it. The research challenge in this case is extending current ontologies and producing specific but highly reusable ontologies for the problem at hand. We envision that TOVE ontologies will be utilized by the Exchange agents and the TAEMS task structures (Decker 1995) will be translated to TOVE ontologies.

Representation of risk structures using Bayesian Networks brings about two main advantages. Firstly, the variables are identified and their temporal and causal relationships are explicitly accounted for. This results in an economic representation of the joint probability in question, which in turn yields efficient inference algorithms[2] (Jensen 1996). In our framework Bayesian Networks are more valuable for their representation rather than efficient inference. Furthermore, we are more interested in the qualitative inference rather than a quantitative one (Wellman 1990). The research challenge for this portion of the proposal is to come up with a directed acyclic graph representation that is rich enough for a sophisticated representation of risks and uniform inference mechanisms that will work with qualitative as well as quantitative input. The definition of risk variables (nodes of the Bayesian Network) based on TOVE ontologies, particularly the activity ontology (Gruninger & Pinto 1995) also poses a challenging problem.

Rule-based approach to encode risk handling methods has been proposed earlier (Niwa 1989), however due to poor representation of risk structures it was not as impressive as claimed. On top of TOVE ontologies and a Bayesian Network representation of risks, a rule-based architecture to encode risk handling methods is more likely to be successful.

As a further extension, this framework will also enable the system to detect patterns in risk management and possibly abstract a notion of "similar" risks. Such an information would be very useful to avoid similar risk situations.

2 Current Status

Within the RaDEO project, Systems Design Management Agent (SDMA) is going to be utilized, first, as a tool to manage risks. However, the architecture is flexible enough to consider other facets of systems engineering like, costs, quality, concurrency, etc.

2.1 Architecture

In this section, we summarize the current state of the development of an SDMA for risk management (hereafter SDMA-Risk). The aim is to summarize the architecture of the SDMA and to facilitate discussion of further developments. The conceptual schema is given in Figure 1. Figure 2, depicts the architecture corresponding to that conceptualization showing dependency of each module/layer on others.

The current implementation of SDMA is in ECLiPSe (ECLiPSe 1995). ECLiPSe is a a flexible Prolog implementation particularly suited for writing constraint programming extensions. SDMA currently uses basic ECLiPSe and the CLP(Q,R) constraint solver (Holzbaur 1995) that comes with it.

[2] Of course, inference in Bayesian Networks is an NP-complete problem (Cooper 1990). Nevertheless, there are various exact and approximate approaches to inference. Usually the exact inference algorithms are exponential in the number of variables. However, we do not expect the number of variables to be too large in representing the risk.

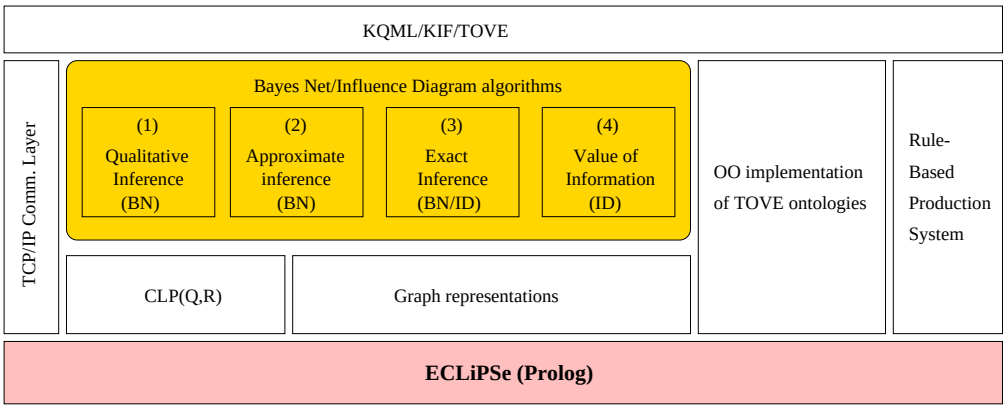

Figure 2: Current Architecture of SDMA

Generic TOVE ontologies on parts, parameters, constraints, requirements, functions, activities, time, organization are available to SDMA in an Object-Oriented implementation with persistent store. It is not yet clear how SDMA is going to utilize this knowledge.

Risk variables are represented as a network explicitly caring for their inter-relations. This gives rise to a Bayesian Network (BN) (Pearl 1988). For risk management purposes it is desirable to add decision and value nodes to such a network and making it a general Influence Diagram (ID) (Howard 1990). BN (and ID) representations are implemented using various graph manipulation utilities in Prolog.

Various algorithms are implemented to operate on the BNs and the IDs and they define basic risk analysis mechanisms of SDMA-Risk. Note that we are not only utilizing BNs to find the joint probability function of the overall risk but also for the side effects of this operation, namely finding the effects of an observation (belief/fact etc.) on each variable (node) of the BN. This is the main risk analysis mechanism that drives SDMA-Risk. There are numerous types of activities on can perform on BNs or IDs depending on what type of information is available.

1. When there are no probability values and only qualitative effects are sought after, we have an efficient[3] algorithm based on (Druzdzel & Henrion 1993).

2. When there are no probability values or only partial values available, we use a probabilistic simulation algorithm to arrive at approximate probabilities for the nodes. This algorithm uses the CLP(R) constraint solver to reduce the search space in random number generation.

3. When there are full probability values for each node of the BN (and in the case where there is a single value node and several decision nodes in the ID) an exact inference algorithm based on non-sequential dynamic programming (Dechter 1996) is available.

4. When the ID grows too large so that inference becomes intractable, one can order the nodes of the BN (within the ID) as per their *value of information*. This implementation is based on (Poh & Horvitz 1996), and extends their framework to subsets of observed variables. This (partial) ordering yields which risk variables need immediate attention without performing inference.

[3] $\mathcal{O}(N)$, where N is the number of nodes in the graph.

A rule-based reasoning system is also implemented in ECLiPSe to encode risk control strategies. This implementation is in the spirit of OPS5, however much more simple and inefficient. In an earlier implementation of SDMA we have used NASA's CLIPS for this task, however we wanted to keep the current SDMA prototype fairly simple and self contained in this implementation. The rules capture certain risk control strategies and when a rule is fired an arbitrary Prolog code can be executed (e.g. sending a KQML message to other agents, creating an action or an issue etc. See Appendix B for an example.).

SDMA-Risk has a KQML/KIF layer to talk to the outside world. Other KQML speaking agents should use a special syntax to call SDMA-Risk predicates (as well as other TOVE related predicates) which is described in Appendix A. However, KIF syntax in this case is trivial since all SDMA related predicates are of the form $pred(x_1, x_2, \ldots, x_n, y)$ with n input and a single output variable. The KQML layer is also knowledgeable about the RADEO dialect and can generate messages in that format to talk to the Exchange agents.

2.2 SDMA as an agent

From the viewpoint of other Exchange agents SDMA(-Risk) is also an agent that has beliefs about the current state (of the risks) of the design process (the BN or the ID), an intention to take action when symptom patterns are encountered (using the rule-based system) and a desire to monitor and keep the risks under control (due to its overall architecture).

Other agents in the Exchange can change the belief store of the SDMA-Risk by

- adding/deleting/modifying nodes from the BN (or ID), and/or
- changing probability information for the BN, and/or
- changing decision, value information for the ID

and then asking for the inference to be performed on the BN/ID once again.

Other agents might change the intentions of SDMA-Risk by changing its rule base and symptom patterns (currently not implemented).

In turn, SDMA-Risk can

- request specific information from the Exchange agents to modify its belief store (typically on a periodic basis).
- ask a specific Exchange agent to perform/stop performing a certain activity depending on the risk control strategy encoded in its rule-base.

2.3 Assumptions about the environment

SDMA-Risk, as an agent, communicates with he rest of the world via a subset of KQML (a preliminary specification of SDMA-Risk KQML is given in Appendix A).

Other agents (human or software) in the environment should know that they can ask SDMA-Risk:

1. to create a new Risk Network,
2. to modify an existing Risk Network,
3. about the status of a Risk Network or a particular node (i.e. risk variable),

4. to store qualitative or quantitative information about the Risk Network,

5. to acknowledge a new observation (evidence) about a particular node,

6. to run an appropriate inference algorithm based on the observation and the state of the network.

It is assumed that a human agent (e.g. Project Manager, Risk Manager) starts the risk management process. This agent identifies the risk variables and the dependencies between them and essentially creates the initial Risk Network[4]. This can be done at the time task structures are being created. The agent has to identify a certain task as a risk variable and define a subtask for it to "register with the SDMA-Risk". The need for human intervention at this point is because of granularity considerations. Not all tasks are equally important from a risk point of view.

SDMA-Risk is capable of handling both qualitative and quantitative risk information.

2.4 SDMA-Risk Version 0.1

The concepts described in this draft are prototyped in SDMA-Risk version 0.1. This version contains only the qualitative propagation algorithm hooked up (cf. Figure 2, number (1) among the BN/ID algorithms.). All the other modules shown in Figure 2 are in place.

This prototype is designed to show the feasibility of the concepts described in this draft. It is not meant to provide full risk management capabilities.

In Appendix B, we describe an example session with the prototype.

3 Summary

SDMA-Risk is designed to manage risk in concurrent engineering environments and in presence of other intelligent agents. The risk management process is initiated by creating a risk network and providing qualitative or quantitative information about that network. Risk handling strategies can be coded in as rules in a rule based system. SDMA-Risk, then takes over and starts monitoring the risk variables that are in its belief network. Whenever, changes occur, and symptoms are detected it fires rules and executes methods (actions) that appears on the right hand side of the fired rule(s).

Currently, only notification is implemented as a possible action, however a whole spectrum of actions can be executed by the SDMA-Risk. It is envisioned that SDMA-Risk communicates severe risk events (determined by thresholds) to the Project Manager with a simple explanation about them. The Project Manager, in turn, takes appropriate action.

The current prototype lacks a graphical user interface for the human agent to initiate the risk management process. However, being able to communicate in KQML it might be possible to use other user interfaces developed within the MADESmart environment for the SDMA-Risk.

References

Cooper, G. F. (1990). The computational complexity of probabilistic reasoning using Bayesian belief networks, *Artificial Intelligence* **42**(2-3): 393–405.

[4] Technically the Risk Network can either be a Bayesian Network where all nodes of the graph are *chance* nodes (i.e. random variables) or an Influence Diagram, where some nodes are chance nodes but there are also decision and value nodes.

Dechter, R. (1996). Bucket elimination: A unifying framework for probabilistic inference, *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, Portland, Oregon, pp. 211–219.

Decker, K. S. (1995). *"Environment Centered Analysis and Design of Coordination Mechanisms*, PhD thesis, Department of Computer Science, University of Massachusetts, Amherst.

Druzdzel, M. J. & Henrion, M. (1993). Efficient reasoning in qualitative probabilistic networks, *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)*, Washington, D.C., pp. 548–553.

ECLiPSe (1995). *ECLiPSe User Manual Version 3.5*, International Computers Limited and ECRC GmbH, Arabellastr. 17 D-81925 Munich Germany.
*http://www.ecrc.de/eclipse/

Gruninger, M. & Pinto, J. A. (1995). A theory of complex actions for enterprise modelling, *Working Notes AAAI Spring Symposium Series 1995: Extending Theories of Action: Formal Theory and Practical Applications*, Stanford.

Holzbaur, C. (1995). OFAI CLP(Q,R), manual, *Technical Report TR-95-09*, Austrian Research Institute for Artificial Intelligence, Vienna. Edition 1.3.3.

Howard, R. A. (1990). From influence to relevance to knowledge, *in* R. M. Oliver & J. Q. Smith (eds), *Influence Diagrams, Belief Nets and Decision Analysis*, John Wiley and Sons, New York, chapter 1, pp. 3–23.

Jensen, F. V. (1996). *An Introduction to Bayesian Networks*, UCL Press, London.

Labrou, Y. & Finin, T. (1997). A proposal for a new KQML specification, *Technical Report TR CS-97-03*, University of Maryland Baltimore County (UMBC), Computer Science and Electrical Engineering Department, Baltimore, Maryland 21250, USA.
*http://www.cs.umbc.edu/kqml/

Niwa, K. (1989). *Knowledge-Based risk management in engineering*, Wiley Series in Engineering and Technology Management, John Wiley and Sons.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan-Kaufman.

Poh, K. L. & Horvitz, E. (1996). A graph-theoretic analysis of information value, *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, Portland, Oregon, pp. 427–435.

Sage, A. P. (1992). *Systems engineering*, John wiley and Sons, New York.

Wellman, M. P. (1990). *Formulation of Tradeoffs in Planning Under Uncertainty*, Pitman and Morgan Kaufmann.

Williams, T. (1995). A classified bibliography of recent research relating to project risk management, *European Journal of Operational Research* **85**: 18–38.

A KQML Specification for SDMA-Risk

In this section we describe the KQML specification for SDMA-Risk. Basic assumptions of agent communication languages, that the message passing is point-to-point and reliable, apply here as well.

A generic SDMA-Risk KQML message has the following keyword inputs:

```
:sender     :receiver    :language   :ontology
:reply-with :in-reply-to :content
```

In all SDMA-Risk KQML messages a `:sender` keyword must be present and should correspond to the symbolic name of the sender agent. The `:receiver` keyword should also be present and must be `sdma` (the symbolic name of SDMA-Risk).

`:language` and `:ontology` are strings that name the language and the ontology used to specify the `:content` of the message. Their defaults are `sdma_risk` and `tove`, respectively.

If the KQML message received has a `reply-with` information, SDMA-Risk will include it in `in-reply-to` of its reply.

The `:content` contains the primary communication to be achieved by the message. It contains an expression which is to be interpreted within the context of `:language` and `:ontology`. The following is a brief description of the `:content` for `:language sdma_risk`.

As suggested in the proposal for a new KQML specification (Labrou & Finin 1997), the `:content` for `:language sdma_risk` has to be double quoted, since `:language sdma_risk` is a subset of `:language prolog`.

SDMS-Risk can handle the following performatives:

achieve The `:content` can be one of the SDMS-Risk services shown in Table 1.

Table 1: SDMS-Risk KQML performative `achieve`

SDMS-Risk command	argument(s)	Description
`bn_initialize/0`		Initialize a Belief Network (BN)
`bn_addArc/1`	from-to	Add a dependency arc to the BN
`bn_deleteArc/1`	from-to	Delete a dependency arc from the BN
`bn_deleteNode/1`	node	Delete a node from the BN
`qpn_addArcSign/2`	from-to,sign	Add arc sign information to the BN
`qpn_deleteArcSign/2`	from-to,sign	Delete arc sign information from the BN
`qpn_addEvidence/2`	node,sign	Add qualitative evidence to the BN
`qpn_propagate/0`		Ask for a qualitative inference on the BN
`qpn_abolish/0`		Abolish the BN and all qualitative info.

The arguments of SDMS-Risk commands are *arcs* (represented as a *from-to* node pair, where both *from* and *to* are node names), *nodes* (represented by its name which is a string, preferably a prolog atom), and *signs* (one of '+','-','?', or '0' corresponding to positive, negative, unknown, and zero effect).

ask-if The `:content` can be one of the SDMS-Risk services shown in Table 2.

error The `:content` of the `error` message contains a string that describes the error. Error message is returned when SDMS-Risk believes that the incoming KQML message is malformed.

Table 2: SDMS-Risk KQML performative `ask-if`

SDMS-Risk command	argument(s)	Description
`bn_active/0`	-	Is there an active BN?
`qpn_active/0`	-	Is there an active QPN?

sorry The `:content` of the `sorry` message contains a string that describes the nature of the error. Sorry message is returned when SDMS-Risk interprets the incoming KQML message but cannot execute it.

B An example session with SDMA-Risk

This section describes an example session with the current prototype of SDMA-Risk (v0.1). The description is for a Sun workstation running Solaris or Sun OS.

The first step is to install ECLiPSe as described in its Installation Instructions. It is assumed that the shell environment variable `$ECLIPSEDIR` contains the ECLiPSe distribution[5] and ECLiPSe binaries are in the search path.

SDMA-Risk comes as a (GNU) zipped tar file. When (GNU) unzipped and untarred it creates a self contained directory for itself. At that directory type[6]:

```
% eclipse -b sdma-risk.script

ordsets.pl compiled traceable 29008 bytes in 0.10 seconds
graphs.pl  compiled traceable 43496 bytes in 0.10 seconds
lists.pl   compiled traceable 14112 bytes in 0.05 seconds
sorts.pl   compiled traceable 8680 bytes in 0.03 seconds
bn_rep.pl  compiled traceable 7960 bytes in 0.32 seconds
qpn.pl     compiled traceable 28472 bytes in 0.05 seconds
kqml2pl.pl compiled traceable 19080 bytes in 0.08 seconds
sdma_kqml.pl compiled traceable 16104 bytes in 0.03 seconds
scattered.pl compiled traceable 14328 bytes in 0.03 seconds
prod.pl    compiled traceable 17840 bytes in 0.12 seconds
forward.rules compiled traceable 1360 bytes in 0.00 seconds
prod.rules compiled traceable 2120 bytes in 0.02 seconds
server.pl  compiled traceable 6512 bytes in 0.02 seconds
test.pl    compiled traceable 1352 bytes in 0.00 seconds
Starting  SDMA-Risk Server
Server at <hostname> 7888
```

The SDMA-Risk is now up and running. It will use port 7888 on the host machine.

Then, open another (X) window (on the same host or any other host). We are going to use this window to open up a telnet session to the `<hostname>` and mimic another agent. From the client window:

```
% telnet <hostname> 7888
```

[5] Note that ECLiPSe has to be installed with the Megalog (database) option for full functionality of the SDMA.
[6] Ignore the warning messages if there are any.

```
Trying <IP address>...
Connected to <hostname>
Escape character is '^]'.
```

We are now ready to send a KQML message to the SDMA-Risk. Unfortunately, this way, we have to reopen the telnet session for each KQML message to be sent. The first message is for initializing a BN.

```
(achieve :sender tester :receiver sdma :content "bn_initialize" :language sdma_risk)^@
```

The character at the end of the KQML message, `^@` is ASCII-0 (zero) and is used to denote the end of the KQML message in this implementation. It is obtained by pressing `<ctrl>-@` on X-Terminal windows to (t)csh.

The resulting message returned to the client is:

```
(tell :receiver tester :content (true) :language KIF  :sender <hostname>/7888)
```

which denotes that the BN is now initialized.

Opening up telnet sessions for each of the following messages, we create a BN for SDMA-Risk:

```
(achieve :sender tester :receiver sdma :content "bn_addArc(a-b)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "bn_addArc(a-c)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "bn_addArc(a-d)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "bn_addArc(c-f)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "bn_addArc(d-c)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "bn_addArc(e-f)"
         :language sdma_risk)^@
```

Then we give the qualitative information:

```
(achieve :sender tester :receiver sdma :content "qpn_addArcSign(b-a,+)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "qpn_addArcSign(a-c,+)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "qpn_addArcSign(a-d,+)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "qpn_addArcSign(d-c,+)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "qpn_addArcSign(c-f,+)"
         :language sdma_risk)^@
(achieve :sender tester :receiver sdma :content "qpn_addArcSign(e-f,+)"
         :language sdma_risk)^@
```

We also give an observation to the SDMA-Risk:

```
(achieve :sender tester :receiver sdma :content "qpn_addEvidence(b,+)"
         :language sdma_risk)^@
```

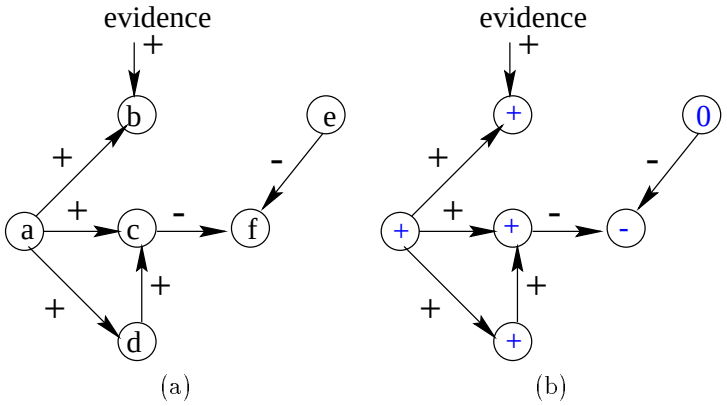

Figure 3: Example QPN for the session. (a) before the inference, (b) after the inference.

In essence, we have created the BN shown in Figure 3.a. There are six nodes in the BN and qualitative influences are shown on the arcs. The + on arc $a - d$ implies that observing a makes the occurrence of d more likely.

We are now in a position to ask SDMA-Risk to perform an inference based on the available information:

```
(achieve :sender tester :receiver sdma :content "qpn_propagate"
         :language sdma_risk)^@
```

This executes the QPN propagation algorithm to find the effects of observing a positive influence on node b on all other nodes in the network. The result of the algorithm is shown in Figure 3.b. Nodes a, b, c and d are positively effected from this observation, node f is negatively influenced, whereas node e is not affected at all.

After performing the inference SDMA-Risk does not send out these results but consults to its rule-base to see if there are any symptom patterns defined for the nodes with increased likelihoods. Assuming that each node in the network corresponds to a risk variable, we have a rule in the rule-base stating:

If there is an increase in the probability of occurrence of a node report it to the appropriate person in the project.

This rule is defined in the `forward.rules` file as

```
(qpn_symptom(Node),+) -: (symptom(Node), +, sdma_notify(Node)).
```

which states that if `qpn_symptom` is detected for a node, then assert variable `symptom` for that node and execute `sdma_notify` for that node.

Having encountered this rule, SDMA-Risk produces the following series of KQML messages:

```
(tell :receiver tester :content (Risk symptom detected for node b)
      :language KIF  :sender <hostname>/7888)
(tell :receiver tester :content (Risk symptom detected for node a)
      :language KIF  :sender <hostname>/7888)
(tell :receiver tester :content (Risk symptom detected for node c)
```

```
      :language KIF  :sender <hostname>/7888)
(tell :receiver tester :content (Risk symptom detected for node d)
      :language KIF  :sender <hostname>/7888)
(eos :receiver tester :language KIF  :sender <hostname>/7888)
```

The last KQML message denotes the end of a series of replies.

Each node (risk variable) may have an owner in the organization. This owner can be represented in the organization ontology in TOVE as an entity who has appropriate role and authorization in the project. The action, then would be using TOVE organization ontology to deduce who will be notified. In fact, the action taken by the SDMA-Risk (`sdma_notify(Node)` in the above example) can be an arbitrary prolog predicate, including all TOVE related `kb_*` predicates, and sending out KQML messages to other agents in the Exchange. This way, one can have a very flexible architecture for Systems Design Management.