Relationship-Driven Policy Engineering for Autonomic Organisations

Kevin Feeney, Karl Quinn, Dave Lewis, Declan O'Sullivan, Vincent Wade
Knowledge and Data Engineering Group, Trinity College Dublin
{Kevin.Feeney|Karl.Quinn|Dave.Lewis|Declan.Osullivan|Vincent.Wade}@cs.tcd.ie

Abstract

Autonomic systems are needed to self-manage the increasing complexity of pervasive communications access and the ubiquitous computing services they offer to humans. Policy based management is the means by which humans will impose their will upon how autonomic systems govern themselves. Ultimately, these systems mediate information and services between people, in a rich tapestry of human associations formed though personal, commercial, and civic relationships. Therefore, if policies are to accurately reflect the wishes of their human authors, the process of authoring or engineering them must closely reflect the intricacies and fluidity of human relationships. We see this as the route to providing intuitive use of policy systems and thus to autonomic systems in empowering their users. We have previously presented a scheme for community-based management that addressed the dynamic organizational aspects of human relations in policy authoring. In this paper we review developments with this approach and describe its integration with a trust management system designed to flexibly regulate access to resources according to the trust relationships between participants. This approach promises to enable autonomic organisations, with structures that automatically adapt to the needs of the users.

1. Introduction

Self-managing, or autonomic, systems aim to provide a level of self-configuration, self-optimisation, self-healing and self-protection that frees the system administrator from having to understand the changing complexities of the distributed IT systems and networks [kephart]. Human interaction with the self-managed system are restricted to the expression of policies or goals expressed at a level of abstraction that is related to outcomes relevant to the human users rather than system level features. The success of such self-managed systems is thus predicated on the human user being able to accurately express the required goals. This is already a challenging task in traditional highly-centralised and hierarchical organizations, where knowledge of the required goals and the authority to impose them is restricted to a small group.

We motivate our work by an analysis of pervasive computing and the goal of such systems self-managing, i.e. of the requirement for autonomic pervasive computing. Pervasive computing is characterized by high levels of transient user mobility between environments that offer services operated by a variety of organizations and individual. Consider, for instance, the daily transitions experienced by a university academic leaving their domestic smart space in the morning, maintaining connectively to email and instant services while commuting on bus then train to work. On campus the academic uses her own office, lecture theatres, seminar room, before meeting a colleague in a local café for lunch and proceeding to the offices of a potential industrial collaborator and then attending a seminar at another institute. At each location she will be offered a variety of services, some available globally via cellular access, but many only available locally and via a variable chain of local wireless access networks, local application service operators and value added service providers. Even this simple example reveals the changing ad hoc network of service and human associations that people must negotiate every day and which pervasive computing must seamless mediate in order to succeed. This may be performed through various adaptive techniques for the composition and presentation of services. Autonomic pervasive computing requires that this adaptation be governed by the policies of the human actors involved in using and operating these services. Thus the policy authoring process, which generates the policies applicable to the use of specific services by specific individuals and groups and particular settings, needs to be conducted based on the ad hoc associations between people, i.e. the uncertain, evolving human relationship they form. This reflects trends in organizational management that has seen the rise of

outsourcing and the emergence of organizational forms where authority over particular resources is delegated to autonomous bodies. Typically, however, the delegating organization needs to exercise fine-grained control over the limits of the mandate and be capable of exercising ultimate control over their crucial resources [thompson]. The model of autonomous groups co-operating as part of larger organizations is increasingly being employed in SME value chains, collaborative projects between corporations, virtual organizations and the practice of corporate divisions being run as independent businesses. Organizations thus are becoming increasingly decentralized and fluid in their decision making and in the distribution of authority to manage resources, thus greatly impeding the ability of those organizations to accurately model their structure for the purpose of accurate policy development. Policy engineering for autonomic pervasive computing merely represents a further extension of this situation, emphasizing the need to capture and track evolving human relationship in order to accurately generate policies.

We will also explore trust in depth in section three from a viewpoint of providing a trust system that mirrors the human model of trust.

In section 2 we present community based policy management and policy-based personalisation of trust in section 3. Section 4 presents our research towards integrated relationship-driven policy engineering. We finally present our conclusions and future work in section 5.

2. Community-Based Policy Management

In [feeney] we presented a community-based model for the management of policies in organizations with a decentralized and evolving structure. A framework based on this model has been implemented to help policy-driven self-managed systems mediate with organizations in defining management policies in the context of continual organizational change and assisting in the resolution of organizational conflicts in the management of resources that arise as a result. The community-based abstraction is employed to overcome the problems with rigidity to change exhibited by existing role-based policy schemes. The community-based framework is designed to reflect the way that humans have traditionally interacted with organizational policy as described by Djiker and Barrett [barrett]. The community abstraction groups together all of the stakeholders in policy decisions. Policies applied at general levels of the organization allow the behaviour of more specific groups to be bounded by general rules. The fluid evolutionary approach to building organizational structure reflects a progressive grounding approach, where policy sets develop over time. This community abstraction was initially developed through observation of the operations of loose Internet based communities [feeney]. This type of organization represents an extreme in terms of its organizational form in conventional terms, however the flexibility, dynamism and complex distribution of authority of these communities are increasingly represented in traditional forms of organization and will, we believe, become the norm in pervasive computing operations. Such Internet communities therefore are the target of our case studies and experimental work.

This model builds upon previous work in the field of Policy Based Management, but simplifies the specification of the organization by using a single grouping construct, the community, to model the organization. It introduces a concept of delegation whereby authority and rights are delegated to dynamically build a model of the organization. Communities retain all of the advantages of specifying policies for roles or domains: policy decisions are made on the basis of the communities that a person belongs to, rather than their identity and policy can be composed through conjunction of the communities to which that an individual belongs.

However, the community abstraction goes further than the standard role-model in several ways. In particular, communities provide a means of incorporating group context and decision-making; they integrate administration into the model and allow policy rules to be interpreted with respect to the overall structure of authority in the organization. They facilitate the self-modelling of groupings within the community, autonomous control of shared resources and allow for privacy of different parts of the structure. A hierarchy of communities provides a means for modelling the organization in a much more succinct and intuitive way than by using roles with a variety of different relationships between them.

A primary strength of this framework is that it does not require a detailed and costly process of requirements engineering to create a model of the organization. We can create the most basic structure and allow the detailed divisions of responsibility and the organizational groupings to evolve in an organic manner. Thus, we can introduce a PBM system by merely modelling the entire organization as a single community, with authority over the full set of resources managed by the system. As the needs arise we can create sub-communities and delegate to them responsibility for specific resources. The analysis of

policy conflicts can be used to signal structural problems in our model and in the underlying real-world community, thus providing constant feedback to refine the model.
By implementing a community policy framework, based on the Ponder framework, we have demonstrated the feasibility of implementing a community structure in a general policy language. However, there are particular features of the community model, notably the hierarchical enforcement of policies, which are not supported by available PBM systems. Therefore a custom policy language has been used in an implementation of the framework in a open source Internet community portal where it is used to managed the definition of fine-grained access control policies for portal managing CVS repositories, bug lists and feature requests.

3. Policy-based Personalization of Trust

Our approach to modelling trust mirrors the human model in definition, representation, calculation, and outcome. In [quinn04, quinn05] we describe a trust management application that provides a trust based selection process that can assess a range of similar services and select the appropriate service specific to the user's trust requirements. In [golbeck03], Friend-Of-A-Friend (FOAF) [foaf] is extended to provide trust values ranging from one to ten can indicate how much trust or distrust exists between entities but the trust value itself is quite 'shallow'. In our approach trust values use a similar range format but the values themselves have a rich semantic 'depth' on which they are computed.
Our research approach to modelling trust can be viewed as having four levels;

(i) Meta-model,
(ii) Upper ontology,
(iii) Domain specific model,
(iv) Personalisation element empowered through the use of policies.

The meta-model governs the relationships that can be formed between trust concepts to create our upper ontology and the specialised models. Figure 4.1 illustrates the initial three trust relationships, which are "derivedFrom", "informedBy", and "affectedBy". The strongest relationship is "derivedFrom" and implies a measured or evidence based bond between concrete concepts only. The second strongest relationship is "informedBy" and it can be based partially on evidence and can be formed between any conceptual or concrete concept. Finally, the weakest relationship is "affectedBy" and it provides a less tangible relationship between conceptual concepts and both conceptual and concrete concepts. These relationships are also directed. We created the relationship system in an effort to ensure that only abstract (conceptual) relied heavily on well defined (concrete) concepts.
Concrete concepts are derived from evidence based on user interactions with an entity, in this case a service. The properties that make up a concrete concept are therefore well defined. Conceptual concepts are more opinion based and are supported with no real, or tangible, evidence and a conceptual concept's properties are generally not as well defined as a concrete concept.

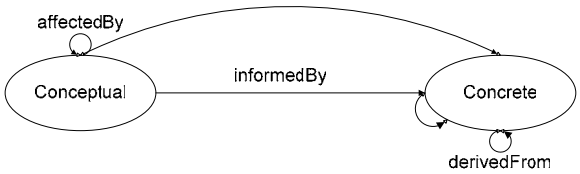


Figure 3.1

The OWL [owl] based upper ontology, figure 3.2, is a generic and reusable model that is the basis for the creation of domain specific models. The upper ontology is made up of trust concepts, trust relationships, and collections of classes for logical grouping. The trust concepts are split into two groups; concrete and conceptual. The concrete trust concepts are "Credibility or Reliability" [Golbeck03], "Reputation" [Golbeck04], "Competence and Honesty" [Grandison00]. The conceptual trust concepts are "Belief" [McKnight96], "Confidence or Faith" [Shadbolt02].

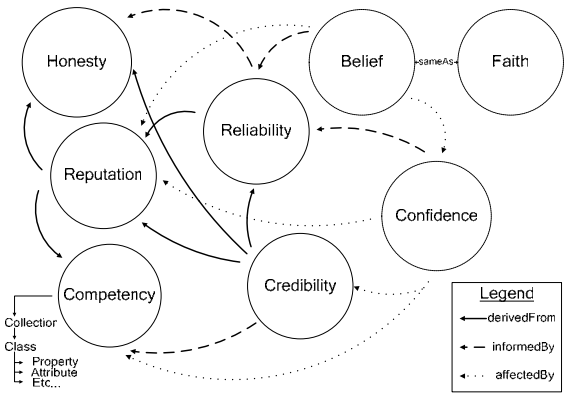


Figure 3.2

In our initial work we created a domain specific model that described the properties and attributes that make up the trust concepts specific to services. These included the attributes that could be perceived as making a service reliable (downtime, mean time between failure, etc). In [quinn05], we argue that the ubiquity and dynamism of services in pervasive computing environments provides users with a myriad of service choices, which implies a greater uncertainty about the trustworthiness of available services due to this greater choice.
Personalisation is empowered through the specification of policies, which enables the creation of individual trust calculation models on a per user basis.
Our trust management application utilises three main components:

1. The upper ontology.
2. The semantic annotation of services with trust data enabled through the upper ontology.
3. A set of trust calculations can be utilized to compute a value for trust.

It is the final component that renders a trust value that forms the basis for trust recommendations. Once a recommendation has been made the application can reconcile it against the users trust policies. The application can then decide whether or not the service corresponding to the recommendation will be selected or not.

3.1 Determination of Risk

Risk is closely associated with trust as there is an element of risk involved with trust, which may range from no risk at all to some degree of great risk. It can be stated that lower risk transactions have a smaller necessity for trust and high risk transactions would have a greater necessity for trust.
Risk analysis is the systematic use of available information to determine how often specified events may occur and the magnitude of their consequences [AS/NZS 4360:1999]. Risk evaluation is the comparison of risk analysis results with the acceptance criteria for risk and other decision criteria [NS 5814].
In the real world risk can be mitigated through the use of information regarding previous actions (reputation) as well as current contextual information. For example, financial institutions may grant loans based on an applicants previous repayment record as well as on their current income and assets. Therefore, risk can be used in the process of determining trust.
In terms of web service composition the risk associated with the constituent services can be of great significance. If two out of three constituent services are considered to have a low risk (of service completion) associated with them and the third constituent service has a larger high risk (of service completion) associated with it then the composite services total risk cannot be viewed as low risk. In essence the risk associated with a composite service is only as strong as its weakest constituent services risk level. There is also the risk of using a service(s) associated with a particular service provider.
Risk management aims to identify, analyse, evaluate, resolve, monitor and communicate risk. When preparing to orchestrate web services it may be pertinent to carry out a risk assessment. The challenges in assessing risk in mobile ad-hoc environments are quite similar to trust. The trust and risk information that we hold about ourselves can be viewed as reliable. However, when dealing with an individual service or service provider it may not be sufficient to accept their stated risk levels (i.e. a percentage). Therefore, just as with trust, a decentralized approach for determining risk in mobile, ad-hoc networks is required. The web of trust approach can propagate risk information in the same way as it passes authentication information. For example, it may be necessary to ascertain a service/providers reputation in order to (partly) base risk calculations on.

3.2 Community-based Risk Approach

We suggest a distributed approach to risk evaluation based on either a straight forward aggregate or weighted aggregate of the recommended trust values for a given action as provided for by individuals within a community. For example, the action of accessing a file owned by the community would result in a three step process as outlined, from a high level view, in the steps below;

1. Community members, or individuals, would be asked to individually specify a level of risk that is associated with access to the community owned file.
2. The individual also provides a trust threshold value, respective of the specified risk.
3. The community threshold trust value would then be calculated as either;
    a. The aggregation and average of the individual trust threshold values.
    b. Community policies specify the trust value threshold based on aggregated and averaged individual risk indictors.

The ability to indicate, or calculate, trust threshold values based on steps outlined above accurately

models the real world situation. One could envisage a community as having a low risk regard for the action of turning off a light, but the same community may place a high regard on the action of deleting community information.
The community can optionally use a weighted aggregate system to allow the management of the community to take into account an individual's status within that community. A highly weighted individual could be perceived an influential person, which is reflected in the final aggregate recommendation.

3.3 Group Trust Approach

In [quinn05], the method for determining an individual's trust in a service is presented. The methodology for determining trust for community entities is similar. As per figure 3.3, the subject (an individual) can calculate a trust value for a given object (in this case a service & provider) from many sources of evidence or opinion, which are provided for by participating collaborators in the community.

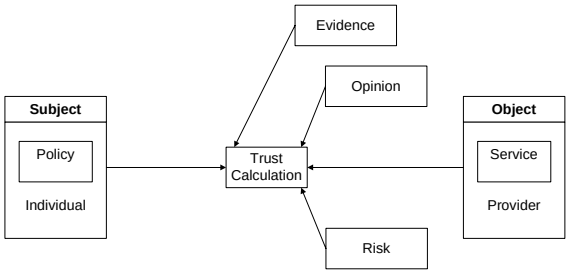


Figure 3.5

In order to determine how much trust a community has in an object we utilize the individually calculated trust values in conjunction with the individuals' inter-community status (weighting). The status of an individual within a community is analogous to a website on the Internet. It is then possible to create an algorithm that calculates community trust values on an authority and hub basis, much like Google's PageRank [brin98]. A highly trusted individual in the community would have a greater say in the determination of the overall communities trust values. Individuals calculate trust values for other individuals within a community just as they calculate trust values for other objects.
This can be extended to the community-to-community scale whereby communities will calculate trust values for other communities. This has the advantage of reducing the overhead for calculating a given individuals trust value across disparate communities. The overhead is reduced as one can infer that individual X has a minimum trust value Y simply because she is part of community Z, where Z insists of a minimum trust value of Y for membership.

4. Towards Integrated Relationship-Driven Policy Engineering

By incorporating trust calculations into our community based policy management infrastructure, we can create a sophisticated and fine grained management model, where the authority to modify policy rules, and hence control managed resources, is regulated according to the trust relationships between the participants. This model lends itself to dynamic, flexible environments, where management responsibility is shared amongst the users. To illustrate the operation of this management model, we examine its application to a pervasive computing environment.
The particular environment is a large research institution, containing a number of pervasive computing spaces. In line with the overall aims of pervasive computing, the system designers aim to allow the users of the space to manage the resources themselves. The institution hosts a constantly changing group of researchers, who are occupied in investigating diverse aspects of pervasive computing and have a wide range of expertise in various technical fields. Thus, it is not possible for the designers to specify in fine detail the configuration of the resources in the spaces, the access rights of the users, or even to identify a set of common roles that users can fit into.
Due to these limitations, the system designers adopt a community based policy management approach, where the user community manages the resources through the dynamic formation of sub-communities and the delegation of resources to them.
Within the community management model, the resources of a community are modelled as a tree, with each node on the resource tree being associated with a particular "authority tree".

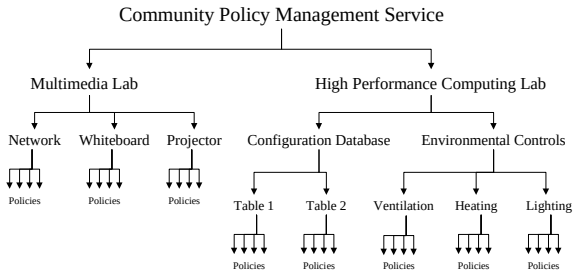


Figure 4.1 Resource Tree

Figure 4.1 shows a section of the resource tree for the pervasive computing environment, and figures 4.2 shows the authority tree associated with each node on the resource tree. Distribution of rights is achieved by passing a *resource authority* to a sub-community. Communities operate internally by establishing decision making rules for the resource authorities that they possess.

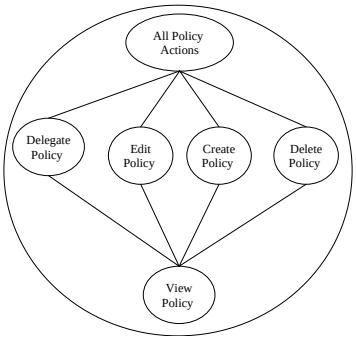


Figure 4.2 CPMS Authority Tree

The designers create a minimal structure for the user-community, shown in figure 4.3, where an *administrators* group is established as a sub-community of the overall users community and all rights to alter the policy store for the pervasive computing spaces are delegated to this community.

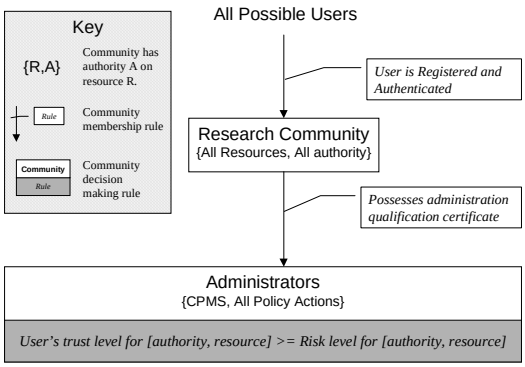


Figure 4.3 Community Structure

In the absence of detailed information about the competencies of the user, the membership qualification rule for the administrators community is defined as being *all users who have completed a course in administration of the system*. Once this basic structure has been put in place, the community based policy management system takes over the management of all of the resources in the spaces and the fine details over exactly how the access rights and responsibilities are distributed evolve according to the choices that the users make.

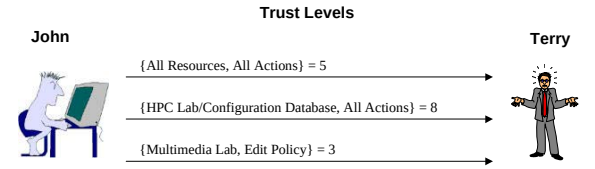
The initial community structure is obviously too simple to be practical as a management model. Although all of the administrators are technically qualified to write policy rules to manage the environment, this is an extremely coarse metric for assessing competence. Some administrators will only be interested in the management of subsets of the managed resources, while some management actions on resources will effect other users of the environment and thus need to be restricted to users who are particularly competent and trusted in that area. Therefore, the administrators need to construct a more detailed sub-division of the access rights to management actions on the managed resources, by restricting the set of resources that different users can manage and the set of management actions that they can take.

This problem can be solved through the community management infrastructure by creating sub-communities and delegating management responsibility for certain resources to them and by setting different decision making methods for different actions. By using trust calculations as a factor in community membership rules and as a decision making criterion, much of the overhead in terms of designing the community's organizational structure and authorization procedures can be made invisible to the users. Their access rights to management actions evolve according to the levels of trust that other users have in them. The community management infrastructure becomes an autonomic system with little need for users to consciously interact with it or even be aware of its existence.

Trust & Risk

From the point of view of our trust model, each node on the authority tree is a service and each user is a service provider. Each user can specify a trust relationship towards another user at any level of granularity, whether that be for all actions on all resources, or to take a specific action on a specific resource. As outline in section 3.2 each user can also specify a trust threshold for any action on the resource-authority tree. This allows each user to effectively specify the risk that they believe is associated with particular actions. The various trust relationships from user a to user b are mapped as values to nodes on the resource tree. Typically, users will start out with a coarse general trust relationship with other users and develop it as they gain experience of the user working in different contexts, by creating more specific trust

relationships with them. For example, we may discover that a certain user is very competent in administering databases but less so with other resources. Thus we might specify a greater trust relationship towards them for actions on databases and lesser trust in all other cases, as illustrated in figure 4.4.



John has relatively high trust in Terry's ability to manage the HPC Lab Configuration Database and relatively low trust in his ability to change policy rules in the multimedia lab and medium trust otherwise

Figure 4.4 Trust Relationships for Individuals' Resource Authorities

In our case study, the decision making rule for the administrators community is simply any *user who is sufficiently trusted by the community to carry out the action*. The management system evaluates this by calculating a community *risk score* for each action and a community *trust score* for each user in taking this action. The community risk score is a trust threshold calculated from the various risks that the users have associated with actions, while the trust score is calculated from the trust relationships towards that user from other members of the community. A user is permitted to carry out the action only if their calculated community trust score is greater than the community trust threshold (i.e. the risk) associated with that action. The structure of the community and the distribution of access rights evolve dynamically as the members of the community gain more knowledge about their colleagues and populate the resource trees with trust scores. The progressive accumulation of knowledge by the community members feeds directly back into the management system and is the basis for all access control decisions. Users who become less trusted by the community, will instantly find this reflected in their access rights.

However, while the incorporation of trust relationships into the community management infrastructure does allow us to dispense with some of the complexities of community structure, there are still many situations where the creation of specialized sub-communities is required. For example, in our example we might find that many of the administrators lack competency in the management of high performance computing resources and moreover lack the understanding to evaluate the risks of various actions in the area. In this case, the creation of a sub-community with specific responsibility for this resource makes sense. By making the membership rule for this specialized sub-community dependent on the competency of the user in the area, as judged by the larger community, we incorporate trust calculations and leverage the knowledge of the community. The new HPC administrators community is then responsible for defining trust relationships and risk scores internally to allocate access rights amongst themselves (figure 4.5)

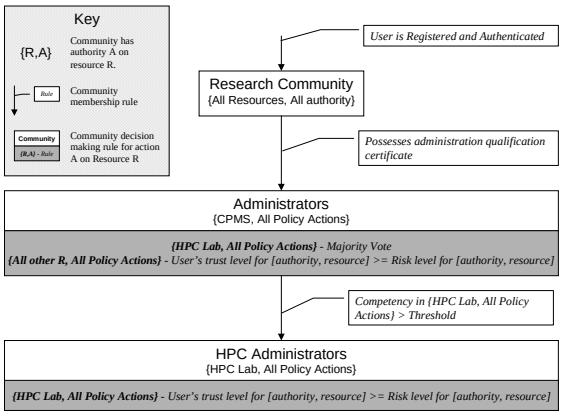


Figure 4.5 Evolving Community Structure

The parent community retains the resource authority that it has given away as a failsafe in case they need to override their decision, but they set an expensive decision making method for invoking the authority. The administrators, or any of their sub-communities can similarly decide to spin off management responsibilities to sub-communities whose membership is made up of those who are judged particularly competent in particular areas.

5 Conclusions and Further Work

By using trust and risk scores as factors in community membership policy rules and community decision making, we can regulate access to management actions (expressed as changes to the policy set for particular resources) according to the relationships within the community. The management infrastructure and community structure are largely invisible to users who merely need to specify their trust relationships to others and their assessments of the risks associated with particular actions. Even these trust and risk values could potentially be implicitly calculated by the management system. This could be achieved either by drawing on existing semantic information, or by observing the behavioural patterns of each user and inferring trust relationships based upon them, which would amount to an entirely autonomic management

infrastructure operating without any conscious user input whatsoever.

In addition to the autonomic distribution of authority, the CBPM also manages the integrity of the policy rule set, largely without any user interaction. When policy conflicts are detected, they are automatically resolved when one of the communities that authored the conflicting rules is an ancestor of the other. When resolution is not possible, a decision making event is invoked in the nearest common ancestor of the conflicting community and possible solutions are suggested to the community for approval. This type of conflict should only occur in the event of a structural problem in the community's organisation or incomplete specification of delegated authority. Thus, the management infrastructure only need interact with the user when a configuration error is encountered and over time we can expect that the users will react to the system's prompts and progressively refine the community's structure, distribution of authority and policy set to eliminate these irresolvable conflicts. The management system will tend towards robustness.

The combination of the CBPM infrastructure and the semantic information embodied in trust relationships enables an autonomic organisation. An organisation where the distribution of authority and responsibility among the members is a function of trust relationships. With the enormous complexity of many modern organisations and the emergence of new horizontal and decentralised forms such as SME value chains and virtual organisations, the ability of organisations to manage their resources securely in a dynamic and unstructured environment is critical. Leveraging the knowledge of the members negates much of the need to carry out the complex, difficult and expensive task of centrally planning and engineering the system. The CBPM system allows the trust relationships to dynamically shape the organisation's structure.

The architecture of the system is illustrated in figure 5.1. The CBPM system requests decisions from the relationship management system and approves management actions when a confirmation is received back. The gathering of users' trust and risk evaluations and the translation of individual trust scores into community trust scores remain works in progress (indicated by diagonal shading on the diagram), but the rest of the system is supported by the existing implementation with little or no modifications. The CBPM treats the relationship manager merely as another decision making module and is not affected by its dependence on trust rather than some other decision making method.

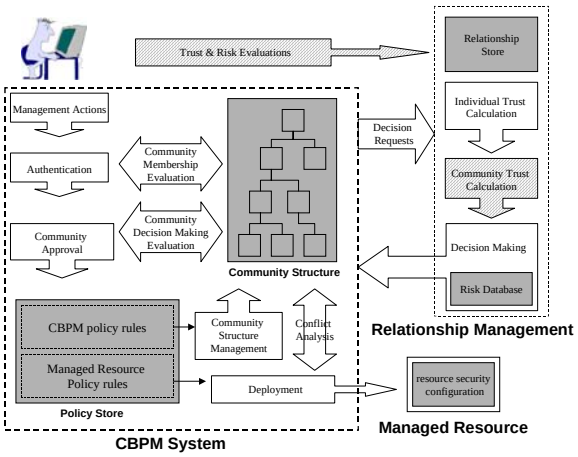


Figure 5.1 CBPM-Trust Architecture

Within the trust sub-system, to date we have implemented the Meta Model, Upper Ontology, a Domain Specific Model (services), which have all been utilised by the *deepTrust* application [quinn05]. The *deepTrust* application was designed and developed in conjunction with Ericsson Research Group.

Our immediate future work aims to create a domain specific model for Trust Based Access Control (TBAC). Expanding the personalisation of all aspects of the model of trust, empowered through the use of policies, is the next key challenge for our research. Personalisation enables the dynamic creation of subject specific upper ontologies, allows subjects and objects to specify policy requirements, and provides a more tailored approach for the definition and utilisation of trust.

Acknowledgements

This work is partially supported by the Irish Higher Education Authority under the M-Zones programme. Kevin Feeney is co-funded by an Ussher fellowship at Trinity College Dublin. Karl Quinn's work receives a scholarship from IRCSET, the *deepTrust* application was designed and developed in conjunction with Ericsson Research Group.

References

[AS/NZS 4360:1999] Australian standard 4360, Second Edition, 1999.

[barrett] Barrett, R., "People and Policies: Transforming the Human-Computer Partnership", 5th

IEEE International Workshop on Policies and Distributed Systems and Networks, IEEE, 2004

[brin98] Brin, S., Page, L., 'The Anatomy of a Large-Scale Hypertextual Web Search Engine', Proc. of the 7th Intl. WWW Conference, April 14-18, 1998.

[feeney] Feeney, K., Lewis, D., Wade, V. "Policy-based Management for Internet Communities", in proc of 5th IEEE International Workshop on Policies and Distributed Systems and Networks, IEEE, 2004

[kephart] Kephart, J., Chess, D., "The Vision of Autonomic Computing", IEEE Computer, Jan 2003, pp 41-50.

[golbeck03] Golbeck, J., Hendler, J., Parsia, B. 'Trust Networks on the Semantic Web', 12th International Web Conference (WWW03), Budapest, Hungary, May 2003.

[golbeck04] Golbeck, J., Hendler, J., 'Inferring Reputation on the Semantic Web', 13th International Web Conference (WWW2004), New York, NY, USA, May 2004.

[grandison00] Grandison, T., Sloman, M., 'A Survey of Trust in Internet Applications', IEEE Communications Surveys, 3, pp. 2-16, Fourth Quarter 2000.

[foaf] RDFWeb: FOAF: 'the friend of a friend vocabulary', available at: http://rdfweb.org/foaf/.

[owl] OWL Web Ontology Language Reference, available at: http://www.w3.org/TR/owl-ref/.

[mcknight96] McKnight, H.D., Chervany, N.L., 'The Meanings of Trust; Technical Report 94-04, Carlson School of Management, University of Minnesota', 1996.

[NS 5814] NS Norsk standard NS 5814.

[quinn04] Quinn, K., O'Sullivan, D., Lewis, Wade, V.P., 'Composition of Trustworthy Web Services', Information Technology & Telecommunications Conference (IT&T 2004), Limerick, Ireland, October 2004.

[quinn05] Quinn, K., O'Sullivan, D., Lewis, D., Brennan, R., Wade, V.P., 'deepTrust Management Application for Discovery, Selection, and Composition of Trustworthy Services.' Proceedings of IDIP/IEEE 9th International Symposium on Integrated Network Management (IM 2005), Nice, France, May 2005.

[shadbolt02] Shadbolt, N., 'A Matter of Trust', IEEE Intelligent Systems, pp. 2-3 January/February 2002.

[thompson]. Thompson P. and McHugh, D., Work Organisations, Palgrave, New York, 2002.