Presented at IE13, Athens, Greece, 16-19th July 2013

Crowd Intelligence in Intelligent Environments: a Journey from Complexity to Collectivity

Idham Ananta
Department of Computer Science and Electronics
Universitas Gadjah Mada
Yogyakarta, Indonesia
idham@ugm.ac.id

Vic Callaghan, Matthew Ball, Michael Gardner
School of Computer Science and Electronics Engineering
University of Essex
Colchester, UK

Jeannette Chin
Department of Computing and Technology
Anglia Ruskin University
Cambridge, UK

Abstract: ***One form of* complexity in intelligent environments arises from their heterogeneous nature. The growing variety of environments and countless stereotypes of users operating Intelligent Environments will, theoretically, increase the complexity and resources needed to utilise them. However we argue that utilizing Crowd Intelligence techniques in Intelligent Environments offer several advantages for dealing with these complexities. A novel architecture called a Crowd Based Heterogeneous Ambient Environment Framework (CHAMBER) is introduced, which proposes the use of hierarchical clustering to discover similarities between the infrastructures of individual Intelligent Environments, which are then offered for use by other users wishing to construct their own intelligent environments.**

Keywords: Intelligent Environment, Crowd Intelligence, Collective Intelligence

I. INTRODUCTION

In the twenty years since Mark Weiser [1] published his vision about pervasive and ubiquitous computing, there has been a steadily increasing interest in this area of research. The number of publications in the area has been growing constantly as new kinds of intelligent environment infrastructures and applications are introduced. Researchers offer various points of view, including cross-disciplinary discussions, and combine them with appropriate emerging technology; this continuous variance in perspectives has led to the area of Intelligent Environments becoming very heterogeneous.

Many researchers aim to bridge this heterogeneous gap by investigating features like mobility, adaptability, and security. However, in most cases, expanding the scale of heterogeneity means increasing its complexity which, in turn, leads to more computing overheads and increases the complexity further. This work offers a collective approach to overcome the complexity problems by gathering crowd sourced experiences and information to prompt the user with appropriate options.

II. THE COMPLEXITY PROBLEMS

A. Related Work

We believe that complexity results from the multi-dimensional heterogeneity of intelligent environments, involving numerous physical devices, and individual preferences and needs of the user. This section describes the range of research undertaken previously, spanning problems of device diversity, mobility, portability, and adaptability to the complexity of differing environments and the preferences of the people involved. As a consequence we propose a model for describing the complexity that includes a taxonomy to categorize the different aspects involved.

One of the earliest research projects involving heterogeneous devices was undertaken by Wang and Garlan [2]. The problem they tried to solve was how to enable users roaming through environments with differing sets of devices, to request generic system services that are independent of specific hardware . In solving this problem they introduced the notion of Task-Driven Computing, using several applications-independent primitives that bound the actual services available in the environment to a generic goal (the task). All of its dynamic configurations were handled by a proprietary protocol. Jiang et al worked on a similar idea, and introduced task computing to fill the gap between task and services and proposed the use of a task hierarchy that supported reusability and shielded the user from low-level complexity using abstract representations. These two approaches, while able to offer some degree of portability, had adopted a tightly coupled system, which made them more difficult to interoperate with different environments.

Masuoka [3], introduced a more open approach in the notion of STEER (Semantic Task Execution Editor), by adopting a Service Oriented Architecture. Masuoka make use of SSD (Semantic Service Description) in their system, which was computed with the help of inference, execution, and a discovery engine to deliver semantic services requested by the user. Jiang et al took a different approach using context dependent task modelling to work across different platforms, based on the fact that each single environment had a different configuration, and that each should share a common understanding about the context. Both approaches promised a higher degree of openness, but still engendered the issue of increasing computing demand, as the level of heterogeneity grows, to an extreme level.

The approaches described above used tasks as a higher level abstraction and utilized them as a means of interaction between people and the intelligent environments surrounding them. There was some work, which involved the notion of 'programming' especially for the end user. The MIT Alfred project [4] sought to allow users to compose a program via teaching-by-example, using the concepts of 'goals' and 'plans'. Their system proposed to make use of a macro programming approach that could be generated by the verbal or physical interaction. Hague [5] proposed a tangible media metaphor to represent programming logic in which programming was undertaken by turning appropriate faces of a cube. Humble [6] proposed a jigsaw puzzle metaphor as graphical programming representation to build applications.

The PiP framework [7][8], has several key concepts, which are relevant to our research. It introduced Programming-by-Example as a new way to program/customize simple tasks in digital homes ; this involved the end-user performing their desired activity using devices in the environment which the system encoded as a "program" that was executed later when the same condition(s) reoccur. For example, an end-user might always reduce the volume of an mp3 player when the phone rings, and so PiP system allowed the user to automate this by utilizing a programming-by-example technique (the user simply demonstrated the system behaviour they required). In this way digital home customization could be performed intuitively by non-technical end-users with very little training required.

In addition to reducing complexity, using an end-user driven approach, enabled users to unleash their creativity and promoted a sense of ownership and trust as all preferences are their own work (not machine generated preferences); however, these approaches also require a significant amount of cognitive work from the user, which may not be suitable for some people[9][14]. From this viewpoint, Ball [9] proposed the concept of adjustable autonomy utilizing human and agent teamwork to manage and program an Intelligent Environment.

This project is sponsored by Directorate General of Higher Education Republic Indonesia

The ATRACO project [10] introduced the metaphor of Ambient Ecologies to describe smart environments enriched with connected sets of devices and services, and the interaction between them, the environment itself, and its users. The interesting aspect of Ambient Ecologies is their scalability and adaptability: an Ambient Ecology could be composed of a number of artefacts that make up composable building blocks, which not only offer a higher degree of scalability, but also the ability to adapt to changes. Their approach was based on well-known software engineering principles which used the SOA model combined with intelligent agents and ontologies.

ATRACO used a local ontology to support what was termed an application sphere (a self-contained but connected environment). A combination of a Fuzzy task agent with ontology management allowed application sphere to adapt to local changes. However, while it represented a commendable advance on earlier ideas, it required relatively high computational resources on the client/local side.

B. Model for Multilevel Complexity

This study of Complexity is motivated by an increasing number of intelligent environments which tend to have a different set of devices and services, along with diverse occupants who frequently have different preferences and needs. Figure 1 shows our model of types of Intelligent Environment used to differentiate the variety of users, environments and preferences. The work described in this paper is situated as a single user and single environment space where the operations of the system are primarily focused on how to fulfil user preferences (expressed as rules/sets of activities via end-user programming, as previously described) within the resources available in the environment. However, when more than one user is involved in a single environment the complexity level rises because the system must adapt to and accommodate additional user preferences using the same available resources. Furthermore, for the case of a single user who moves from one environment to another, the issue of mobility is raised as another facet of complexity. On top of all this, the system must also account for the complex changes in user preferences occurring over time; these could be attributed to a large variety of reasons, such as changes in the environmental state, changes in the user's experience and confidence in the system, or the personal mood and feelings of the user at any given point in time [14].

To capture these issues we created a model of multi-level complexity, as shown in Figure 1. We identified that there are three main variables which define the complexity:

1) *Physical World. – This consists of a set of devices and services. Environments will change, when new devices are introduced, new configurations are set up, state variable changes (like temperature, humidity, brightness level etc.), or if a new user enters the environment.*

2) Preferences - Are a set of rules defined by user. A user can change his/her preferences based on changes in the environment, or his needs.

3) Users - As social creatures, it is common for environments to have more than one person. More people bring more preferences, resulting in the need to represent differing and sometimes changing group preferences.

III. Crowd Intelligence

A. *Motivation*

Crowd Intelligence or as it is more commonly referred to as, Collective Intelligence, is a means to harness group knowledge. For example, searching particular keywords from the millions of pages on the Internet might return unlimited and irrelevant possibilities. However by using crowd intelligence techniques, such as the Google PageRank scheme, the knowledge or behaviour of other users can be harnessed. Google PageRank uses outside links to a particular web page as reference to define how important the page is. This mechanism could be seen as akin to web pages voting for each other to determine which one is the most important and more relevant. This is a prime example of how Crowd (or Collective) Intelligence operates. It works by harvesting knowledge from the 'crowd' of human users who have used their own intelligence to reason about information and services, which is then captured in a way that can be used by the wider web community (and the companies providing these services). In that respect, it can be seen as an alternative to artificial intelligence for reasoning about networked information or a way of shifting the balance from artificial to natural intelligence. These techniques have played a key role in the success of companies such as Google, Amazon, Netflix etc who use similarities between customers as part of their recommendation mechanism.

We argue that Crowd Intelligence can offer similar advantages to the Intelligent Environment World for the following reasons:

1) *Reduction of Computational Load* - AI is known to be computationally intensive employing complex knowledge representations (e.g. Ontology) and reasoning engines. This scheme replaces those mechanisms with a simpler mechanism of recording people's behaviours. The scheme is intrinsically distributed (the reasoning tis undertaken by numerous human clients).

2) *Heterogeneous Capabilities* - Crowd intelligence does not require any strict hierarchy or formalism to make it work. It does, however, require that hosts openly share information in a common representation such as XML meta-data. While there is a concern about privacy and security, the meta-data provided, along with 'behaviour' logs collected, can be anonymised.

3) *Human Centric* – This scheme alters the balance of human versus artificial agent reasoning in favour of the people. Of course this does not mean that such reasoning is better but it does open up an interesting line of research that could investigate how that balance might be optimized, or even made adjustable.

4) *Searching and Ranking* - A typical page rank system on a search engine works by assessing how relevant the documents are, based on a list of keywords. A widely adopted method for doing this is to collect documents (some call it crawling), and index them, thereby enabling faster and more effective searching. There are a variety of techniques used to facilitate successful document ranking , for example: frequency based metrics, page-rank algorithm metrics, and neural networks. Frequency based metrics compose their page rank based on how frequent particular words show up in the documents. Page-rank algorithms, however, compose it by looking at how many other pages link to the page in question, as with Google PageRank (previously described). Neural networks learn to associate searches with results based on the links people click on after they get set of results.

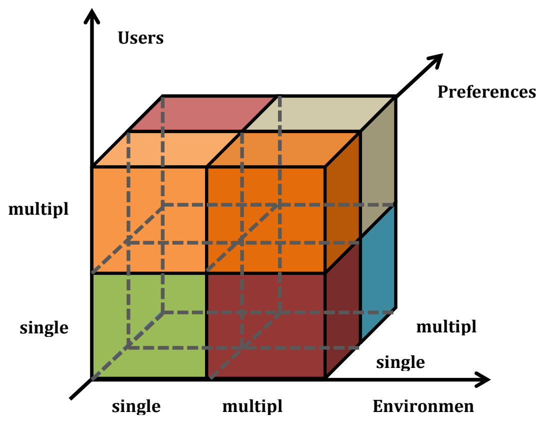


Figure 1. Idham-Callaghan-Chin Complexity Model

In our intelligent environments framework, searching and ranking is used to create a list of rules that are relevant to a particular situation. However there are important differences such as IE resources will not, in general, contain forwarding links to each other, therefore ranking metrics based on linking are less important in intelligent environments and are excluded from our scheme. However, since IE devices will share (exchange) rules/preferences between environments, we utilize a neural network approach to learn from these crowd choices. In our approach, we achieve this through the use of a multilayer perceptron (MLP) network consisting of sets of nodes and connections between them.

B. *Clustering and Collaborative Filtering*

Clustering is a mechanism to identify groups with similar patterns or characteristics. It is a type of unsupervised learning which is able to detect similarities in data. Our framework uses this technique to discover groups of intelligent environments that have a similar set of devices and services.

The purpose of our clustering mechanism is to discover similar environments, so it can formulate recommendations or find the appropriate rules from environments exhibiting similar patterns of devices and services. Any rules that work in one particular environment should also work in an identical environment. In our proof of concept system, we search for identical configurations but clearly, in practice there may always be small differences which would need to be addressed by employing appropriate agent technologies, such as fuzzy logic [16].

Hierarchical clustering aims to organise data into a hierarchical group structures, based on the observed similarities between the data [15]. In the context of intelligent environments, it could be used to find groups of similar devices or services within/across environments at different scales. An intelligent-school for example, might have intelligent-classrooms as well as intelligent-staffrooms, each with their own rules based on different devices and services contained in each. However there might also be some rules that are applicable to the scope of the entire school. So using hierarchical clustering, it is possible to detect two small groups, classrooms and staff room, and also a bigger group, which is the school itself. This approach helps detect the scope of particular rules.

Collaborative filtering is used to search a large set of items, to find a smaller set, qualified by some variables. In a movie recommendation application, for example, collaborative filtering is used to search a large group of customers, and create smaller groups of them who have similar tastes in movies. It produces a ranked list of recommendations based on aspects of their personal preferences.

In Intelligent Environments, this technique could be useful to narrow down sets of user preferences and device rules, based on specific relevant variables such as user-rating, energy saving potential, the number of environments using them, etc.

This technique is not only useful for finding workable solutions for similar environments, but also has potential for recommending rules or actions for specific situations based on either the similarity of the rules (e.g. the same category, such as temperature management) or the specific rules chosen by a similar user (e.g. based on age, gender, etc)

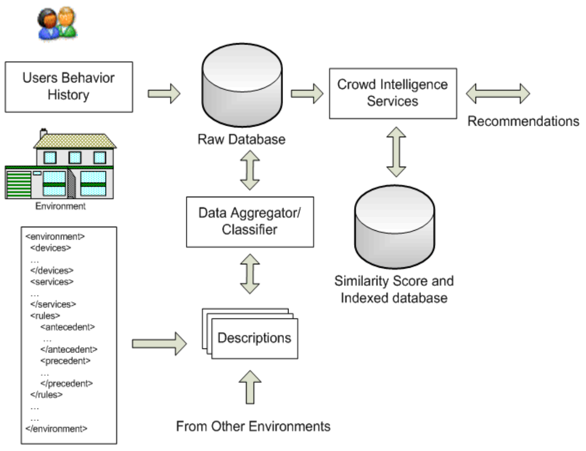
In the proposed framework, each intelligent environment advertises its set of devices and services as meta-configurations, creating a pool of configuration collections, each of which can be individually ranked. To find a set of environments that have similar configurations, a comparison mechanism is required which, in our framework, is a similarity score. We have investigated two ways for doing this: Euclidean distance, and Pearson correlations, which will be discussed later in this paper.

IV. CHAMBER

In this section we describe our Crowd Based Heterogeneous Ambient Environment Framework (CHAMBER). We start with the architecture, describing the implementation and go on to detail the techniques used in this framework.

A. *Architecture*

At a high-level, CHAMBER adopts a simple client-server architecture shown in Figure 2. A server provides



```
<environment>
  <devices>
  ...
  </devices>
  <services>
  ...
  </services>
  <rules>
    <antecedent>
    ...
    </antecedent>
    <precedent>
    ...
    </precedent>
  </rules>
  ...
  ...
</environment>
```

Figure 2. CHAMBER Architecture

computational resources, intelligence tools and a database to store persistent information. The client in this framework is the physical environment, consisting of its own processor devices, sensors, and users. The client is required to provide some essential information such as the environment's configuration, the rules adopted in the environment, and the user behaviour in accessible interfaces.

There are two types of services provided by CHAMBER. The first one is a synchronous service; this stateless service takes the form of client queries to the server. It carries local configuration and information, and waits for the results from the server. The second service is an asynchronous service. This service runs in the background and operates over a longer period of time. One of the examples is a data aggregation service, which will continuously crawl to every environment accessible to update the current CHAMBER knowledge-base.

Recent work by Ball & Callaghan [11] [12], developed the Adjustable Autonomy Intelligent Environment (AAIE) model; this model allows rules (descriptions of desired automated actions) to be created at varying levels of agent autonomy, including either fully-autonomous (using agents to create rules on behalf of users) and non-autonomous (users creating rules via end-user programming). An important aspect of the AAIE model is a confidence-based mechanism used by the agent in creating/learning rules. The more evidence the agent collects to support a learnt rule (e.g. by observing and matching user actions in the environment), the higher the confidence level grows for a particular rule. Confidence decreases can also occur for a rule, however, if it was to trigger an undesirable action that is corrected by the user. This confidence-based mechanism provides a basic ranking system for the rules generated within a single intelligent environment, by ensuring that the more successful rules have a higher level of confidence.

In the CHAMBER framework, such data will form the 'MetaData' made available to the CHAMBER 'Aggregator', as shown in Figure 2. Other MetaData will revolve around the intelligent environment configuration, such as the types of devices available, and any groups of devices that are commonly used together; what's more, the MetaData will include information about the user of the environment, such as any observable traits they may have, e.g., they may be very conscientious with energy saving and keep device usage to a minimum, or they may always require high levels of lighting. By using crawling techniques, the Aggregator gathers the MetaData from all intelligent environments on the CHAMBER network to populate the central database.

The journey from complexity to collectivity starts from collecting descriptions from multiple environments. The description, regardless what kind of configuration in each environment, should include configurations, rules and attributes, plus user information. These three variables come from our complexity model and play a key role as a core way to structure the knowledge from heterogeneous environments. While there might be several ways to parse unstandardized descriptions, we opted to create a more standardized approach for performance reason. Each implementation of CHAMBER in a new environment would use a wrapper to present their environment configuration and activity into a standardized form, in this case an XML page.

A list of devices and services, middleware used, and any other environment configuration related information are presented in the description page. Several rules applied in the environment and their attributes, such as confidence levels, duration, and energy saving levels are listed in the documents. User's profile and its activities (list of rules used in a particular environment) are also described in the document. While this scheme opens other issues such as privacy and security, they are considered out of the scope of this paper.

B. Clustering and Ranking Environments and Rules to Organize Crowd Intelligence

The CHAMBER server will periodically mine the information from the descriptions. It uses the stored set of environment configurations to discover groups of similar environments. This is undertaken by parsing the descriptions and creating similarity vectors of the environment. The same approach is used in the rules. It parses a rule, looks at its similarity (from its antecedent and precedent) to create the rule's similarity vector. The environments' similarity vector is useful for discovering groups of similar environments and identifying sets of rules that, hypothetically, might work in each other's environment. Later on, if some local environment requires workable rules for its environment it will recommend rules from the most similar or perhaps identical environment. The environment's similarity vectors will add weight to rule's rank since it means many environments use them. It can be seen as every environment collectively "votes" for rules, so the rules which have the most votes should have higher ranked in the recommendation. A sample of the computation involved is presented in Table 1.

The description of the rules provided by each environment includes a description of the devices and services used, enabling it to perform some degree of polymorphism. This means that, though a particular rule, such as a lighting systems rule was applicable to a classroom, it might be possible for it to work in a bedroom, as long as the rules include set of devices/services, which are available in both environments.

TABLE 1.RULES RECOMMENDATION COMPUTATION TABLE

Env. ID	**Crowd Intelligence Computation**						
	(SS)	***Rule1 CL***	***SSxCL***	***Rule2 CL***	***SSxCL***	***Rule 3 CL***	***SSxCL***
001	1.0	3.0	3.0	2.0	2.0	4.0	4
002	0.80	3.5	3.76	4.0	3.2		
003	0.60	2.0	4.0	2.5	1.5	2.5	1.5
Sum SSxCL			10.76		6.7		5.5
Sum SS			2.4		2.4		1.8
SSxCL/SS			4.48		2.79		3.05

EnvID = environment ID, SS= Similarity Score, CL=confidence Level/Rating. This computation sample shows that the recommendations rank is: 1. Rule 1(4.48), 2. Rule3(3.05), and 3. Rule 2(2,79)

C. Involving user preferences

The CHAMBER framework can be run anonymously, which is adequate for some routine activities such as managing temperature, ambient light and energy conservation. However, for more subjective rituals, such as studying, reading, sleeping, etc. a user's personal preference might differ significantly to the average person, and thus need a more personalized recommendation.

Adding user preferences into the system requires additional information about the user: user's attributes (such as age, gender, personality, income, etc.) and user actions (searching, rating, using, etc.). Some recommendations can be offered based on similar attributes, or similar actions. In Chamber we achieve this using the 'similarity vector' approach we described earlier together with , collaborative filtering. This technique tries to recognize the voting patterns for particular rules. The recommendation system will search for other users with similar patterns, and recommend those rules. Thus, this approach offers a human centric agent vision since it bases recommendations from the other user's choices (rules) as against rules generated by a machine (AI).

V. CONCLUSION

We have described the complexity of heterogeneous intelligent environments and have modelled it using a multidimensional complexity representation. We have argued that in the future, with growing numbers of interconnected intelligent environments on the network, there will be a huge pool of online environments and devices raising the possibility of harvesting this information to the benefit of the collective users. In particular we suggest that this vast pool of environments will produce an increasing likelihood that a user of one particular environment will be able to find other environments or devices with similarities between physical configurations, rules/preferences applied, and user preferences which could be applicable to their environments.

Thus, in this paper we have investigated how the collective reasoning of the similarities from numerous intelligent environment users can be harnessed by individuals to customise their pervasive computing environments. We have described the concept and provided an overview of related work. In addition, we have presented a novel architecture that builds on our earlier work in adjustable autonomy and end-user programming enabling those schemes to be given crowd sourced recommendations that replace intelligent reasoning in existing schemes. The motivation for this work is to both reduce the computational overheads (by replacing artificial intelligence with natural intelligent) and to improve the quality of decisions by making them more human centric in origin.

ACKNOWLEDGEMENTS

This project is part of two years collaboration research between University of Essex and Universitas Gadjah Mada, supervised by project chairman Prof. Dr. Jazi Eko Istiyanto, from Department of Computer Science and Electronics Universitas Gadjah Mada.

REFERENCES

[1] Weiser, M. (1991). Computer in the 21st Century.

[2] Wang, Z., & Garlan, D. (2000). *Task-Driven Computing.* CMU-CS-00-154, Carnegie Mellon University, School of Computer Science, Pittsburgh.

[3] Masuoka, R., Parsia, B., & Labrou, Y. (2003). Task Computing - the Semantic Web meets Pervasive Computing. *International Semantic Web Conference.*

[4] Gajos, K., Fox, H., & Shrobe, H., "End user empowerment in human centered pervasive computing", in *Proceedings of Pervasive 2002,* (2002), 1-7.

[5] Hague, R. (2003). Towards Pervasive End-User Programming. *Proceedings of Ubiquitous Computing*, (pp. 169-170).

[6] Humble, J., Crabtree, A., Hemmings, T., Åkesson, K.P., Koleva, B., Rodden, T., Hansson, P., "'Playing with the Bits', User-Configuration of Ubiquitous Domestic Environments", Proceedings of UbiComp 2003, Springer-Verlag (2003), 256-263.

[7] Chin, J., Callaghan, V., Clarke, G., "An End User Tool for Customising Personal Spaces in Ubiquitous Computing Environments", *Lecture Notes in Computer Science: Ubiquitous Intelligence and Computing*, Springer-Verlag (2006), 1080-1089.

[8] Chin, J., Callaghan, V., Clarke, G., "Soft-appliances: A vision for user created networked appliances in digital homes", *Journal of Ambient Intelligence and Smart Environments 1,* (2009), 69–75.

[9] Ball, M., Callaghan, V., Gardner, M., & Trossen, T. (2009). Exploring Adjustable Autonomy and Addressing USer Concern in Intelligent Environments. *Intelligent Environments 2009.* Netherlands: IOS Press

[10] Goumopoulos, C., & Kameas, A. (2009). A Service Oriented Architecture Combining Agents and Ontologies Towards Pervasive Adaptation. In V. Callaghan (Ed.), *Intelligent Environment 2009* (pp. 228-236). Barcelona: IOS Press.

[11] Hagras, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A., Duman, H., "Creating an Ambient-Intelligence Environment Using Embedded Agents", *IEEE Intelligent Systems, Vol.19, No.6,* (2004), 12-20.

[12] Hagras, H., Colley, M., Callaghan, V., Clark, G., Duman, H. and Holmes, A., "A fuzzy incremental synchronous learning technique for embedded-agents learning and control in intelligent inhabited environments," in Proc. IEEE Int. Conf. Fuzzy Syst., HI, 2002, pp. 139–145.

[13] Jiang, F., Li, J., & Zhu, Z. (2008). A user-centric Task Computing Architecture for Pervasive Computing. *Third International Conference on Pervasivce Computing and Applications ICPCA,* (pp. 491-496). Alexandria.

[14] Ball, M., Callaghan, V., "Managing Control, Convenience and Autonomy: A Study of Agent Autonomy in Intelligent Environments", *in Special Issue on Agent-Based Approaches to Ambient Intelligence in the AISE book series*, IOS Press (2012).

[15] Johnson, Stephen C. "Hierarchical clustering schemes." Psychometrika 32.3 (1967): 241-254.

[16] Lynch, C.; Hagras, H.; Callaghan, V., "Using Uncertainty Bounds in the Design of an Embedded Real-Time Type-2 Neuro-Fuzzy Speed Controller for Marine Diesel Engines," *Fuzzy Systems, 2006 IEEE International Conference on* , vol., no., pp.1446,1453, 0-0 0