The design of a multi-agent system for flaw-response modelling

J P McKenna, A Gachagan, S D J McArthur and N I McLean

Paper presented at NDT 2006, the 45th Annual British Conference on NDT, Stratford-upon-Avon, UK, September 2006.

Theoretical flaw-response modelling programs are used to simulate ultrasonic flaw detection and provide a cheap and fast alternative to test-piece trials during the Inspection Qualification (IQ) process. The selection of the most appropriate model for a given test scenario requires a high level of technical expertise, necessitating the use of a qualified NDT engineer. Moreover, due to the heterogeneous nature of these models, integration with other NDT software tools can be difficult and requires the user to be responsible for data management within the system.

The multi-agent system (MAS) utilises intelligent software agents capable of autonomous behaviour and decision-making skills in order to create a single software environment to provide support for IQ. The motivation behind this MAS approach, the design and implementation of the agents and the benefits offered by the MAS will be described. A test scenario of a multi-agent IQ will be used to demonstrate the potential advantages offered by this novel systems approach. Specifically, the contribution of these modelling agents to the overall system and their interactions with other tools to automate the IQ process will be addressed.

1. Introduction

Non-destructive testing (NDT) engineers utilise mathematical models that simulate inspection procedures as an essential way of overcoming the cost and time difficulties associated with producing test-piece trials. Validated flaw-response models provide a cheap, fast and robust alternative to support the inspection procedure. Parameters of a specific test, such as defect skew or defect depth can easily be varied to effectively study the consequence on a specified threshold response level without any additional cost or time overhead.

The model allows the user to specify the inspection in question; the user enters the parameters concerning the test component, the type of probe used to simulate the ultrasound, the raster scan of the probe and the defect size, shape and location. The executed simulation then produces results showing the theoretical signal amplitude for that particular simulation.

There are, however, drawbacks to utilising models to provide reliable inspection information. The use of models and, in particular, creating their input files and analysing their results is often restricted to a few experienced individuals who have worked with the models over a long period of time. These experts have in-depth knowledge of the theoretical techniques associated with the models and when presented with several available models are able to select the most appropriate one to describe the current inspection scenario.

Much work has been conducted within the research community to improve the use of models in the inspection procedure by developing intelligent software systems that incorporate them. By automating time-consuming tasks and providing decision support, the intelligent systems assist engineers who utilise mathematical models and aid in the interpretation of their results. One such system is that developed by Robinson *et al*(1) that created an Expert System (ES) to aid theoretical flaw modelling. The ES is capable of interpreting warning flags produced by models operating outside their regime of validity and updating the original simulation to produce a valid modelling scenario.

The primary shortcoming of the majority of existing systems is that they are stand-alone, interpreting a single data source using a particular Artificial Intelligence (AI) technique due to problems with system integration. They work by themselves, unable to communicate or cooperate with other systems resulting in repetitive data re-entry. What would be advantageous is a method of integrating such intelligent systems along with other software tools used in the inspection procedure to form a community of cooperating entities that would free the user from menial/repetitive tasks and also allow less experienced users to utilise the models in an inspection qualification procedure.

An alternative to traditional integration approaches is offered by Multi-Agent Systems (MAS) which provide standardised communications and protocols between individual software modules called 'agents'(2). MAS such as developed by Hossack *et al*(3) provide a flexible and extensible architecture, open to the integration of legacy intelligent systems with new software systems.

Adopting a Multi-Agent approach has enabled a legacy ES incorporating four validated models to be effectively and easily integrated with newly developed models and a Finite Element Modelling (FEM) commercial package to allow timely inspect procedures. Furthermore, this forms part of a larger system, a MAS for NDT, that comprises various software tools such as ray-tracing, coverage mapping, and a least detectable defect tools that continue to be developed from earlier research conducted by McNab *et al*(4).

The paper commences with a summary of the mathematical modelling process and the requirements of a more automated system for test inspection. The design process used in developing the MAS is discussed along with the individual flaw-response agents presently in the system. Finally, a test scenario is presented to illustrate the functionality of the MAS and highlight the benefits of this approach.

2. Mathematical modelling

Using mathematical models to simulate test inspections can be a time-consuming process, which is usually carried out by a suitably experienced NDT engineer. The manual process as demonstrated in

James P McKenna, Anthony Gachagan and Neil I McLean are with the Centre for Ultrasonic Engineering, University of Strathclyde, Glasgow G1 1XW, UK. Tel: +44 (0)141 548 3477; Fax: +44 (0)141 548 2950; E-mail: j.mckenna@eee.strath.ac.uk

Stephen D J McArthur is with the Institute for Energy and Environment, University of Strathclyde, Glasgow, G1 1XW, UK. Tel: +44 (0)141 548 3477; Fax: +44 (0)141 548 2950.

Figure 1 remains relatively unchanged whether using the models in isolation, with the ES or using the FEM package.

The inspection process commences with the NDT engineer designing an inspection that is capable of detecting a defect, having particular properties and characteristics that might cause a failure in the test component. Parameters describing the four main aspects of the inspection must be derived: the probe used to simulate the ultrasound, the raster scan of the probe, the test component under investigation and the properties relating to the embedded defect. Using their in-depth knowledge of the theoretical techniques associated with the models, the engineer must select the most appropriate model that best describes the given test scenario from the available models, to produce accurate simulated response readings.

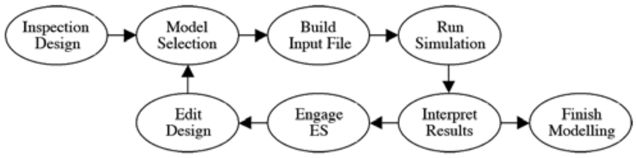

Figure 1. Modelling process

Using these parameters, a suitable input file is built in the format required for the model selected. This requires specific training and experience in the particular model as native keywords are often used and parameter values must appear in the file in the correct order or format to successfully run the simulation.

The model simulation is run using the input file producing an output file containing the simulation results comprising of response values with respect to probe position and highlighting any possible areas where the model may produce inaccurate readings. Again, the engineer is called upon to use their knowledge in interpreting the simulated results and reviewing whether this inspection is capable of detecting the expected flaw in the test scenario. If the results are considered accurate, they may be used in the case of proving the designed inspection is suitable to detect the expected flaw or used in further software tools to provide visualisation for the inspection. However, if decided that the results are inaccurate**,** the test may be redesigned, the model substituted and/or the input file edited before re-running the simulation again. This process continues until an apposite design is reached that is capable of producing an acceptable set of response results.

To accelerate this modelling process the ES was developed to aid the user in the result interpretation stage. Knowledge elicited from engineers was used to develop an intelligent rule-based system that is capable of automatically analysing the results and making changes to the parameters based on 'rules of thumb' used by engineers in the field. Although the addition of the ES has greatly improved the process (allowing less experienced users to carry out test inspections in a greatly reduced time period) inspection design, model selection and data entry still required to be done manually.

Modelling is a user-led process that is time-consuming and repetitive. The user is responsible for all data management within the system; they must ensure that changes to the inspection are noted and accurately re-entered in subsequent model runs. This task can become unwieldy and difficult when considering that there are in excess of 75 parameters involved and over 20 specific keywords for each model.

3. Requirements

Discussions with the engineers currently using the models and the ES, in conjunction with the larger NDTWorkbench[4] software package, state that the tools provided by the package have greatly improved the Inspection Qualification (IQ) process, reducing the time and complexity involved. However, the system lacks integration and is fundamentally a collection of stand-alone software tools grouped together in the same package. There is little or no sharing of data between the programs, relying on the user to continually re-enter the same data again and again. Inspection data and test results should be stored in a manner that makes it accessible to all tools within the system.

To allow for further growth and development of the software package a new open architecture should be employed. This will allow not only new software tools such as new mathematical models to join and interact with the system but allow legacy and commercial systems to be integrated into it too. This will greatly increase the functionality of the system and provide more choice and flexibility in the IQ process.

Continuing the work done by the ES, monotonous tasks such as data gathering and data entry should be automated. The system should be capable of constructing input files, analysing results and editing parameter without the direct intervention of the user. To allow less technically experienced users to utilise the system (and hence release more experienced engineers to do other important tasks) further decision assistance should be supplied such as selecting the most appropriate model during this stage of the process based on the current inspection's specifications.

4. Integration issues

From the engineers' requirement it is apparent that what is needed is a more open architecture that permits the sharing of a group of common data in a flexible, extensible and intelligent system that can cooperate with legacy and future systems. Understandably, this kind of freedom and compatibility in software design raises several significant issues.

The first fundamental issue that must be addressed to allow data sharing is that of inter-system communication as each system may have its own communication etiquette. Secondly, the ability to pass messages between systems is all well and good, but messages must be understood without confusion. To tackle this problem, a common vocabulary that each system subscribes to should be developed to remove any ambiguity in the message content.

At present there is no procedure or protocol to incorporate existing software into the overall system. The biggest hurdle to cross here is the fact that legacy systems exist on different computer platforms and are written in different programming languages making cross-platform communication difficult. Finally, to allow the system to perform more tasks automatically without the direct intervention of the user, the system must have a degree of inbuilt intelligence. Decision-making and task reasoning skills should be a feature of all tools included in the system using knowledge elicited from experts.

5. Multi-Agent System (MAS)

MAS are able to overcome the limitations associated with the current approaches to system integration by offering a common communication language[2]. Each system can be considered as an 'agent' operating within a community of agents, namely the MAS. Through various software programming techniques the agent is provided with inbuilt intelligence so that it is able to manage its own processes and communications with other agents. By communicating with other members of the MAS, each agent can cooperate to provide all the benefits of system integration in a flexible and open manner.

The agent research community has produced a number of standard Agent Communication Languages (ACL) for MAS, in particular Knowledge Query and Manipulation Language (KQML)[5] and the language implemented in this system, Foundation for Intelligent Physical Agents (FIPA) ACL[6]. FIPA is an IEEE Computer Society standards organisation that promotes agent-based technology and the interoperability of its standards with other technologies. FIPA ACL adopts the approach of separating the agent's message type from the message content. Message types are defined to reflect natural human speech and clearly indicate the intended action of the message, for example 'inform' and 'request' relate to the passing of information between agents. The message content contains the information passed and the vocabulary used is left to the developer to define in the ontology, which contains all the terms used in the system's domain. Since the system is applied in the NDT domain the ontology will have terms representing the key concepts used in an inspection such as probe, component and defect.

To simplify the implementation of the MAS a middle-ware application Java Agent Development Framework (JADE)[7] that complies with FIPA specifications and provides an array of graphical tools is employed. JADE is a software framework completely implemented in Java language allowing the MAS to be distributed across different machines running with different operating system platforms.

Allowing the system to be disturbed offers a clear improvement over conventional integration by taking advantage of parallelism to reduce processing overheads, however this also introduces further issues. Agents need a means of locating other agent members of their community and ascertaining what services and resources these agents can provide. In a conventional system this information would need to be hardwired into each member of this system, greatly limiting the flexibility and openness. MAS overcomes this by providing utility agents such as the Directory Facilitator (DF) and the Address Management Service (AMS) agents which provide a 'yellow-pages' service of all the agent names, addresses and services available in the MAS.

Integrating legacy software into a new system typically requires a retrofit approach which could potentially involve hours of re-development work. This issue is handled in MAS by 'wrapping' the agent communication and language functionality around the legacy system and implementing it as task within the new 'legacy' agent[5].

Through a software add-on library to the JADE framework called Jadex (JADE eXtra)[8] the agents are given decision-making and task reasoning skills that will allow them to initiate and carry out tasks autonomously. Jadex provides a reasoning engine which follows the Belief Desire Intention (BDI)[9] model and facilitates easy intelligent agent construction with sound software engineering foundations.

The flexibility and extensibility of MAS have been exploited extensively in the power industry through projects such as ARCHON[10], COMMAS[11] and through research projects such as the PEDA MAS[3]. The novelty in this project lies in the exploitation of the benefits of MAS in an NDT application.

6. MAS development

6.1 Agent identification

One of the first tasks involved in developing the MAS is identifying what agents are needed in the system to accomplish all the tasks the system is required to perform. In the case of the modelling agents and legacy systems this task becomes quite trivial as each software program equates directly to its own agent, for example the PEDGE software model relates directly to the Pedge Agent.

6.2 Task identification

To allow the task reasoning capabilities of the agents to create goals and plans to execute the user's requests, a list of tasks is drawn up for each agent in the system. These tasks map directly to individual tasks the agents are expected to perform, for example gather inspection data, build input file, run simulation, etc.

6.3 Agent interactions

Agents are expected to communicate and collaborate with other agents in order to complete their tasks on behalf of the user. A set of possible agent interactions is drawn up to determine the required message types and possible data exchanges. These agent interactions take the form of diagrams showing clearly the type of messages passed between specific agents in the community and the purpose of the messages.

6.4 Ontology

The next stage in the development of the MAS is defining an appropriate ontology for the given domain, ensuring that it covers all possible message contents established in the Agent Interaction stage. A Java based ontology editor called Protégé[12] was used at this stage to construct the ontology from the main terms and concepts used in the NDT domain and hence the MAS. Each concept is broken down further into its attributes that can then be further defined using data types, values and units (Table 1).

Table 1. Probe ontology concept

Concept	Attribute	Data Type	Value	Units
Probe	CrystalShape	Enumerate	Elliptical	
			Rectangular	
	Size			
	WaveType	Enumerate	Compression	
			Shear	
	ProbeType	Enumerate	Contact	
			Immersion	
	ProbeSystem	Enumerate	Pulse-echo	
			Tandem	
	Frequency	Float		Hertz
	ProbeAngle	Float		Degree

6.5 Implementation

Using the JADE framework and Jadex add-on libraries described earlier, individual agents can be written in the Java programming language.

7. Modelling agents

7.1 Introduction

The following sections discuss the development of a group of agents, which shall be referred to collectively as the *flaw-response modelling agents*, which form a large and important part in the overall NDT MAS. This group comprises agents developed from both legacy software systems and software created specifically for this project.

The legacy systems include four validated models (PEDGE, COREDGE, PKIRCH and CORKIRCH) utilised by British Nuclear Fuels Ltd (BNFL) to simulate ultrasonic testing procedures within the nuclear industry, as well as a knowledge based Expert System to aid theoretical ultrasonic flaw modelling. A commercial software package PZFlex produced by Weidlinger Associates Inc that provides Finite Element Modelling (FEM) functionality will also be incorporated into the system.

London South Bank University (LSBU) has developed further software models to complement the existing ones and a Model Facilitator agent has been created to manage model selection within the MAS. The agents have been implemented using JADE and Jadex using the JAVA object-orientated programming language.

7.2 Legacy flaw-response models

Over several years, BNFL have developed a comprehensive, accurate and flexible theoretical model for the ultrasonic inspection of smooth planar defects in ferritic steel; the principal objective of these models is to provide NDT engineers with simulated flaw-response data for a given test scenario. Two of the models, PEDGE and COREDGE, are based on the Geometrical Theory of Diffraction (GTD), while the remaining two models, PKIRCH and CORKIRCH, are based on the Elastodynamic Kirchhoff Theory. The PEDGE and PKIRCH models are used when a direct-echo response is expected, whereas the COREDGE and CORKIRCH models are used when a corner-echo response is expected (Figure 2). A direct-echo response is one where the ultrasound impinges directly upon the defect before returning to the probe. A corner-echo response involves the ultrasound impinging on the defect before reflecting off the back wall of the component's surface prior to returning to the probe.

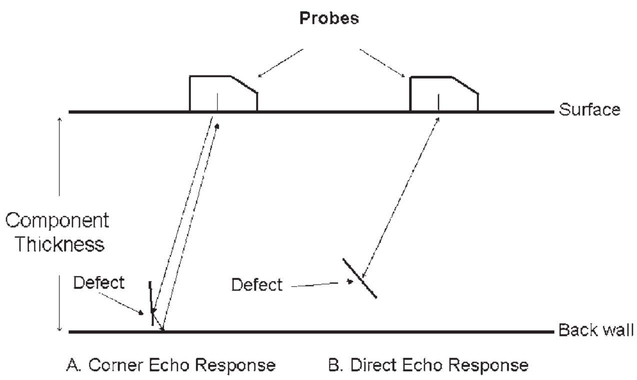

Figure 2. Echo response types

The four models were integrated into the MAS by wrapping agent functionality (communication and reasoning skills) around the legacy system and implementing it as a task within the agent, for example the PEDGE modelling software is used to perform the 'run model' task of the Pedge Agent. To further automate the process, other tasks were added to the agent that provided automatic retrieval of data stored as beliefs within the system to prevent the repetitive input of the same data for each model run or test scenario. Manual tasks performed by the engineer such as construction of model input files, analysis of output files and presentation of results were also developed into agent tasks reducing the prior knowledge needed to make use of the models.

7.3 Expert system

The wrapper technique is again employed to integrate the ES into the MAS using various software functions to provide a translation service between the agent JAVA language and the legacy system native code which is C++ in this case. To maintain the flexibility of the MAS the ES is encapsulated as its own agent, separate from the above models, to allow the user the option of engaging the ES to help with data interpretation or to use the models singularly without assistant. This provides various levels of support to accommodate for different user familiarity whilst maintaining the flexibility and extensibility inherent in a MAS.

7.4 New mathematical modelling agents

An important component of the project is the inclusion of new mathematical agents developed by LSBU. Although these models have been created from scratch for this project they are not being coded as agents therefore they can be treated exactly the same way as the legacy models, employing the wrapper technique to include them in the MAS.

The models have been produced to be stand-alone applications rather than tasks within an agent due to their remote development from the agent development at the University of Strathclyde. The flexibility and scalability of the MAS allow this issue to be easily overcome using the software technique mentioned previously.

7.5 Model facilitator

As previously stated, the user of the current system must have experience using each model and be familiar with their 'regime of validity' in order to apply the most appropriate model for the given test scenario. This task requires a great deal of skill and that is the reason why, to date, this process is usually carried out by an experienced NDT engineer. In the current system, with a few models covering a relatively small test scenario range (the four legacy models), this policy is acceptable and easy to manage.

A MAS is hugely extensible, allowing many modelling agents to exist in one system covering a large testing scenario range, for example nearfield, farfield, varying scanning surface, defects of different shapes and sizes, etc. It would be impossible for an engineer to be familiar with every model and its application. Therefore, if a comprehensive system is to be created, able to cover a large range of test scenarios, a system needs to be developed with intelligence that is able to compare various models and select the most appropriate, that will give the best results given the test specification and data available.

With these requirements in mind, the Model Facilitator (MF) agent has been developed to control all agent interactions with the modelling agents, manage and handle modelling data and most importantly select the most suitable model based on the test data and the 'regimes of validity' of the modelling agents.

The MF is able to perform these tasks due to the intelligence provided by the task reasoning abilities of the Jadex software development platform. When requested to provide modelling services by an external agent, the MF determines the available modelling agents registered with the DF and gathers all relevant test data from the larger systems data-handling agents. It uses this information together with the 'regimes of validity' requested from the available model agents to select the model that demonstrates the best fit between the test data and model validity.

This MF approach to model selection offers additional advantages to the operation of the MAS. The communication overhead within the MAS is reduced by providing one agent (MF) as a point of contact for all modelling requests, the alternative would be for each modelling agent to be messaged individually by the requesting agent. Further, by storing relevant data internally the MF can avoid polling agents to obtain information previously sent in an earlier request. Finally, by adopting a subscription approach new information can be sent to the MF automatically again avoiding the unnecessary polling of agents and hence a further reduction in the communication overhead on the system.

7.6 PZFlex

The open architecture and scalability of the MAS allows not only the inclusion of legacy systems but allows additional functionality to be introduced into the system through commercially available software packages. One such package is PZFlex, a finite element modelling (FEM) software application produced by Weidlinger Associates Inc. PZFlex is a time domain finite element program for solving coupled mechanical-piezoelectric-acoustic equations.

Incorporating this type of package in the MAS not only increases the system's functionality but also automates some of the tasks associated with this type of package that can prevent untrained users accessing it. Manual tasks such as building input files and results presentation can be incorporated into agent tasks removing any learning curve associated with software such as this that requires very specific knowledge in areas such as FEM.

8. Test scenario

To clarify the operation of the MAS a small test scenario was created involving the selection of the most appropriate modelling agent for a giving inspection.

When agents are 'born' their first task is to register with the MAS's utility agents, the DF and the AMS, supplying their name and the services they provide and in return are provided with a unique address within the system.

The scenario starts with the user or an external agent (part of the larger MAS for NDT) requesting modelling services from the MF, as this is the mean by which all agents communicate with the flaw-response modelling agents. To accurately select the most appropriate model for the inspection the MF must establish all the agents present in the system that offer modelling services. The MF can achieve this by communication with the DF through a direct approach (request message) or through the subscription route. With the names and addresses of the modelling agents the MF requests the 'validity' of each model, *ie* whether the model works best in nearfield, or with elliptical defects, etc.

The next step in the process is to gather the data for the inspection from the system's data-handling agents. Through request and inform messages the MF obtains the probe, scan, component and defect parameters. The MF is then able to reason over all the information gathered about the models and the inspection to select the model that will return the most accurate response figures for the simulation.

With the selection made the MF requests the services of the chosen agent, passing it all the information gathered from the Data Brokers. The model agent runs the simulation and returns the results to the MF, which in turn returns them to the original requesting agent. The important point that should be observed in the above scenario is the reduced quantity of user interaction involved in the system. The user simply enters their high-level request ('provide modelling results') and the agents use their reasoning skills to determine the tasks needed to achieve this request.

9. Conclusions

A multi-agents approach was adopted to develop a system to assist with the design and implementation of ultrasonic inspection in the NDT field. The work illustrates how a multi-agent system can be used to effectively integrate existing stand-alone systems and newly developed software models to promote communication and cooperation allowing timely inspection procedures. Furthermore, the system offers long-term extensibility through the creation of an open and flexible architecture that permits the extension of the overall functionality.

References

1. R J Robinson and A McNab, 'A knowledge based expert system to aid theoretical ultrasonic flaw modelling', Review of Progress in Quantitative Non-Destructive Evaluation, AIP Conference Proceedings, Vol 760, pp 1974-1981, 2005.
2. M Wooldridge and N R Jennings, 'Intelligent agents: theory and practice', The Knowledge Engineering Review, Vol 10, Issue 2, pp 115-152, 1995.
3. J A Hossack, J Menal, S D J McArthur and J R McDonald, 'A multi-agent architecture for protection engineering diagnostic assistance', IEEE Transactions on Power Systems, Vol 18, Issue 2, pp 639-647, 2003.
4. D McNab, A McNab, P Leggat, R Robinson and J McDonald, 'A software system for inspection qualification', Review of Progress in Quantitative Non Destructive Evaluation, AIP Conference Proceedings, Vol 700, pp 1603-1610, 2004.
5. M R Genereth and S P Ketchpel, 'Software agents', Communications of the ACM, Vol 37, No 7, pp 48-53, 1994.
6. FIPA ACL specifications [online]. Available: http://www.fipa.org/repository/index.html
7. F Bellifemine, A Poggi and G Rimassa, 'Developing multi-agent systems with a FIPA-compliant agent framework', Software – Practice and Experience, No 31, pp 103-128, 2001.
8. Jadex BDI agent system [online]. Available: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/
9. M E Bratman, Intention, Plans and Practical Reason, Havard University Press, Cambridge, MA, ISBN (paperback): 1575861925, 1987.
10. L Z Varga, N R Jennings and D Cockburn, 'Integrating intelligent systems into a cooperating community for electricity distribution management', Int J Expert Syst with Applicat, Vol 7, No 4, pp 563–579, 1994.
11. E E Mangina, S D J McArthur and J R McDonald, 'COMMAS (COndition Monitoring Multi Agent System)', J Autonomous Agents and Multi-Agent Syst, Vol 4, No 3, pp 279–281, 2001.
12. Protégé Ontology Editor [online]. Available: http://protege.stanford.edu/