CREWS Report 98-37

submitted to

4th International Symposium on Requirements Engineering, RE'1999.

Analysing Socio-Technical System Requirements

by

Alistair Sutcliffe and Shailey Minocha

Centre for HCI Design
School of Informatics
City University
Northampton Square
London EC1V 0HB
United Kingdom

+44 (0)171 477 8412
{a.g.sutcliffe, s.minocha}@city.ac.uk

Analysing Socio-Technical System Requirements[1]

Alistair G. Sutcliffe and Shailey Minocha

Centre for HCI Design,
School of Informatics,
City University,
Northampton Square,
London EC1V 0HB, UK
a.g.sutcliffe@city.ac.uk
Tel: +44-171-477-8411
Fax: +44-171-477-8859

Abstract

Few RE methods address analysis of socio-technical system requirements. This paper describes a method for analysing dependencies between computer systems and users/stakeholders in the operational environment. Domain scenarios describing the system and its context are used to create an environment model based on the i* notation. The dependencies between inbound and outbound events between the system and its usage environment are analysed to elicit requirements to process system input or generate system output. Coupling metrics are applied to these events to assess the degree of dependencies between the system and the users. High level requirements are suggested to deal with different types of organisational design. The method is illustrated with a case study of a service engineer support system.

1 Introduction

Few methods have emerged to analyse socio-technical system requirements, even though many problems in requirements engineering are known to have their origins in complex social problems (Macaulay 1996, Lubars et al. 1993). Ethnographic techniques have been applied to gather data on social issues and requirements do emerge from this process (Sommerville and Sawyer 1997). However, there is little generalisable knowledge, models or analytic methods that can be gleaned from ethnography, so the quality of requirements analysis is dependent on the practitioner's experience.

Some socio-technical models have been proposed for RE, notably the ORDIT project (Harker et al. 1993) which describes systems in terms of agents, task and roles. However, few analytic techniques were reported so the requirements engineer was still dependent on experience for interpreting such models. The Inquiry cycle (Potts et al 1994, Hsi & Potts 1995) uses scenarios to investigate barriers to effective use (called obstacles) that may arise in the social domain. However, the Inquiry cycle does not give detailed techniques for analysing socio-technical system dependencies. Analytic guidance has been given in the stakeholder analysis methods (e.g. Macaulay 1996), which advise modelling requirements according to different user categories or viewpoints. Models of dependencies between people and systems in the i* framework of enterprise models (Yu 1994, Yu & Mylopoulos 1994) enables the impact of different

[1]This research has been funded by the European Commission ESPRIT 21903 'CREWS' (Co-operative Requirements Engineering With Scenarios) long-term research project.

technical solutions to be assessed, as well as giving techniques for trade-off analyses between conflicting goals and non-functional requirements, but it does not provide an analytic method or scenario-based RE.

This paper explores the problem of dependency analysis in socio-technical systems by proposing a method for modelling and analysing event flows between users and the intended system in order to derive high level requirements. This extends the work of (Yu & Mylopoulos 1994) by addressing workflow problems via a coupling analysis derived from concepts in modular software design (DeMarco 1978) and organisational theory (Robbins 1990).

The paper is organised in four sections. The next section introduces the method and this is followed by a more detailed description of the method stages, and coupling analysis. A case study of service engineer support system (SESS) runs through these sections to illustrate the method. The paper concludes with a brief discussion.

2 Method Outline and Models

The CREWS-SAVRE (Scenarios for Acquisition and Validation of Requirements) method compares scenarios describing the domain with requirements specifications and models, focusing on events or information flows between the system and its environment. Analysis questions probe the dependencies between people and computers across a tentative system boundary. The system boundaries will change during a requirements investigation as alternative designs emerge; hence there is a single model of the intended system-environment upon which a boundary is imposed. The relationship between scenarios, use cases and the requirements specification is illustrated in Figure 1.

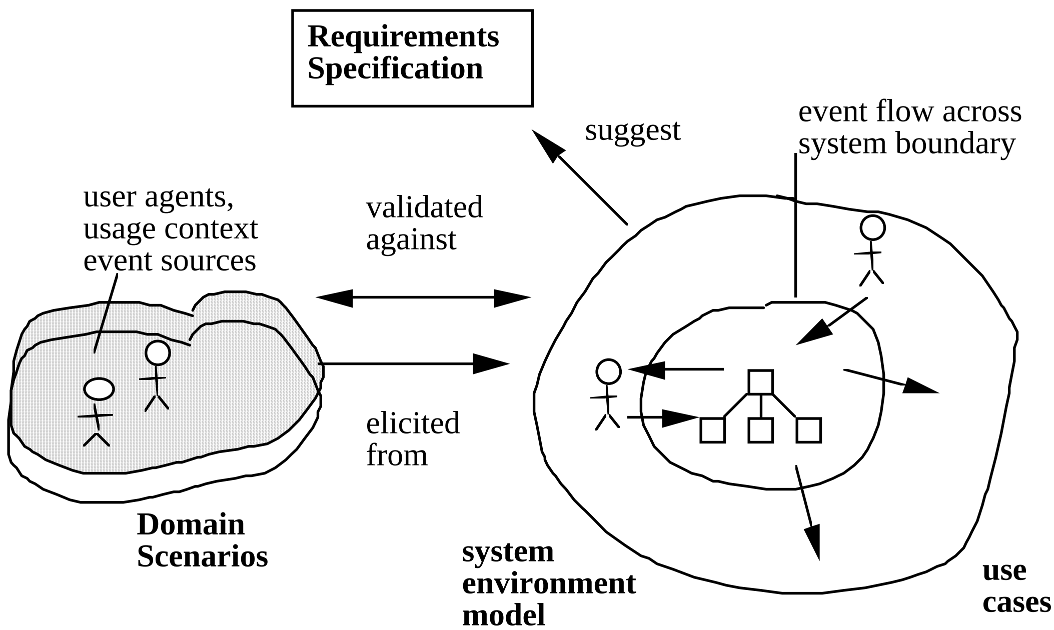

Figure 1 Relationships between Domain Scenarios, Environment Model and Use Cases

The method uses "domain" scenarios that contain descriptions of activities in manual systems, interaction with legacy systems, descriptions of agents, roles, and their organisation settings gathered from real-life examples. These are used to elicit facts for the system environment model.

2.1 CREWS-SAVRE Method outline

The method stages are summarised in Figure 2. The method is iterative, so once the use cases and environment models have been created, all the other stages proceed concurrently.

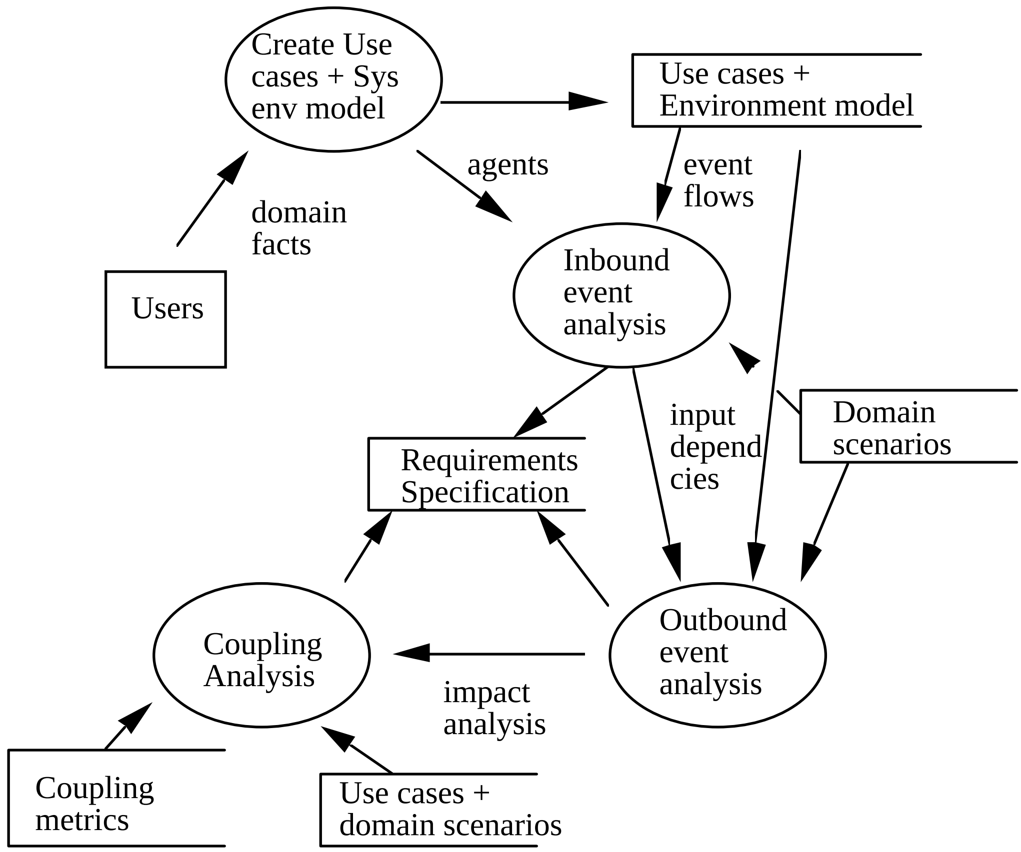

Figure 2 CREWS-SAVRE method stages and associated models

Stage 1. Use Case and System Modelling

The system environment model is created as an overview model. Then use cases elaborate tasks and interactions between agents to provide more detail. Use case modelling follows standard OO procedures (see Jacobson 1992), so it is not described further in this paper. Use cases are expressed as interaction diagrams (UML 1997) to map out the sequential dependencies of event flows between the agents and the system.

Stage 2. Inbound Event Analysis

Each use case is elaborated by comparing it with one or more domain scenarios and tracing events originating either from human agents or from other objects in the system environment. Each event inbound from the system environment indicates a requirement for a system action, as well as an action in the use case. In this manner the event analysis helps to refine the use cases by identifying events and system responses to deal with the events.

Stage 3. Outbound Event Analysis

Outbound validation is more difficult because the impact on a social system has to be judged. By their nature social systems are complex and unpredictable and the change introduced by a computer system frequently produces unanticipated and undesirable side effects. The outbound event flows are traced to their destination and then questions analyse the acceptability of the system output for the user. First, the domain scenario and use case are cross referenced to ensure output is generated when and where it is needed. Steps in the user's task that imply an information need are identified in the scenario; so if a user needs information at a particular point in the scenario a system output function should exist in the requirements specification.

Stage 4. Coupling Analysis

This assesses the dependencies between the social and technical systems. Coupling analysis is based on software engineering concepts and organisational theory (Mintzberg 1979, Robbins 1990) that advises that control based coupling should be avoided. In human organisation control coupling occurs through lines of control, commands, and obligations to carry out activities in response to others. The motivation for avoiding control coupling is twofold. First, it decreases the flexibility of user-system interaction and decreases autonomy. Secondly, too much control imposes the system's goals on the user's working practices, and this may lead to system rejection. The system input and output event flows are classified and counted. The more commands there are, the closer the coupling between the social and technical systems will be. Closely coupled systems are reviewed to either change the design for more automation (computer autonomy) or to increase human control, and design the computer system for an advisory rather than a controlling role.

2.2 System Environment Model

Domain scenarios, captured as narratives or in other media, provide the basis for creating the system environment model. This is an enterprise level model which evolves through different versions as requirements analysis progresses. Rather than invent yet another modelling notation, we have adopted the notion of i* model. This consists of goals, soft goals (non- functional requirements), tasks, agents, and resources (see Yu 1994). The two additions we make are to first, add a socio-technical system boundary, and secondly, relationships to model dependencies between tasks, agents, and objects:

- responsibility: models the association between an agent and an action/task or goal which it has the duty to carry out, e.g. < agent, is-responsible-for, goal | action | task>;
- authority: describes the relationship between two agents in which a dominant agent has authority over the behaviour of a subordinate, or ability of an agent to permit another agent to initiate task or consume some resource; e.g. <agent (x), has-authority-over, agent (y) [task | resource]>;
- accountability: models the relationship by which the achievement of goal or task by an agent is assessed or monitored, e.g. <agent (x), held accountable for, goal,| task, by agent (y)>;
- capability: an agent has the necessary knowledge and abilities for carrying out an action/task, e.g. <agent , has-capability-for, action | task>;

An example of a system environment model is illustrated in Figure 3. This introduces the case study of the SESS (see the use case diagram in Figure 4). Four agents are involved; the controller who receives calls from customers and then allocates the service calls to engineers and schedules their work. The engineers respond to the controllers' commands, drive to the customer's location diagnose the problem with photocopiers and other office equipment and then repair the faults. Engineers interact with customers and the store control agent to obtain spare parts to replenish their local stock. Engineers also have to report their progress, status and location to the controller. The case study focuses on the controller's task and the dependencies between the controller and the engineers in his/her district. The engineer is accountable to the controller for reporting progress on customer calls, and the controller has authority to direct the engineers work.

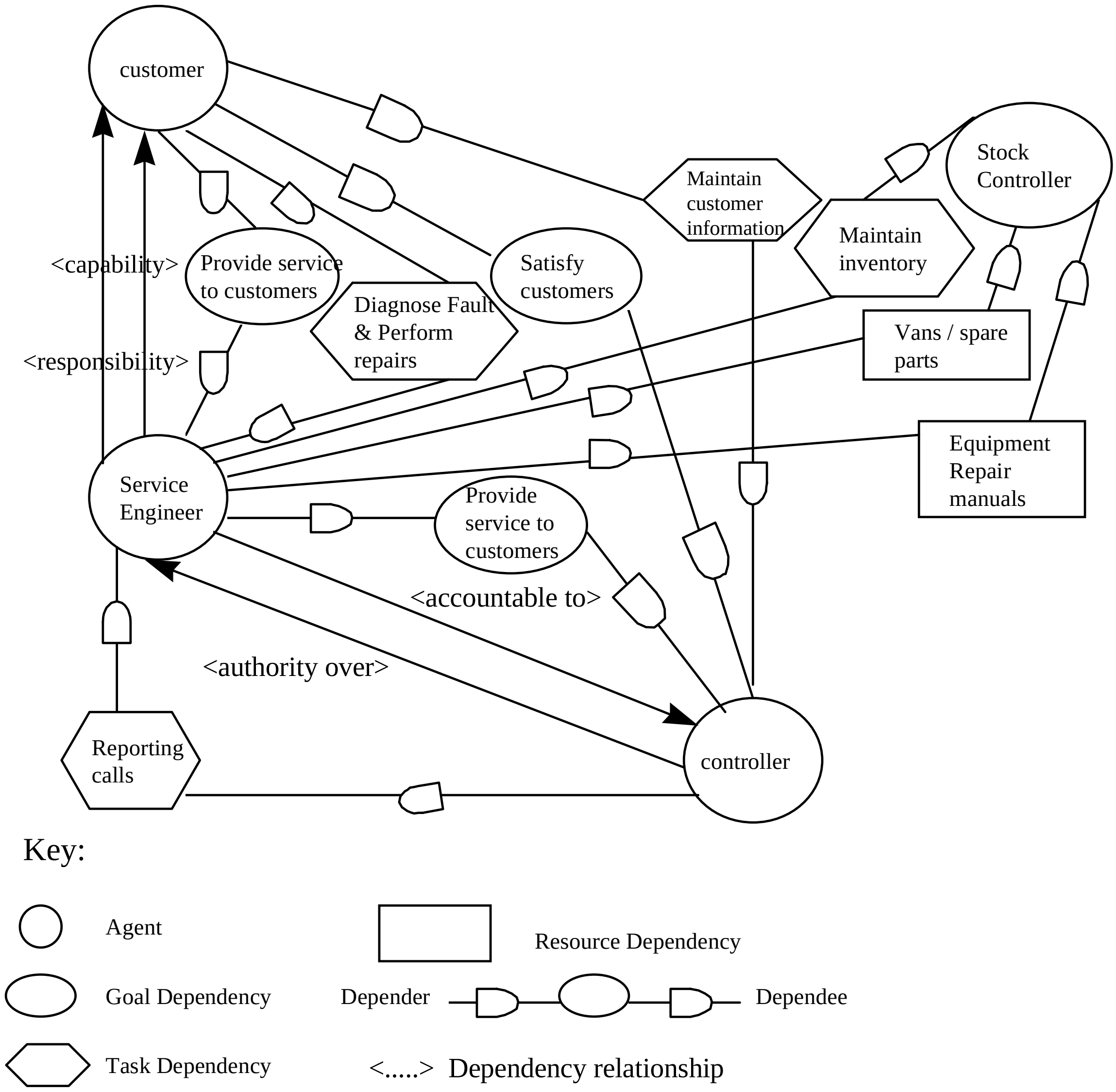

Figure 3. System Environment model for the SESS, following the i* modelling notation

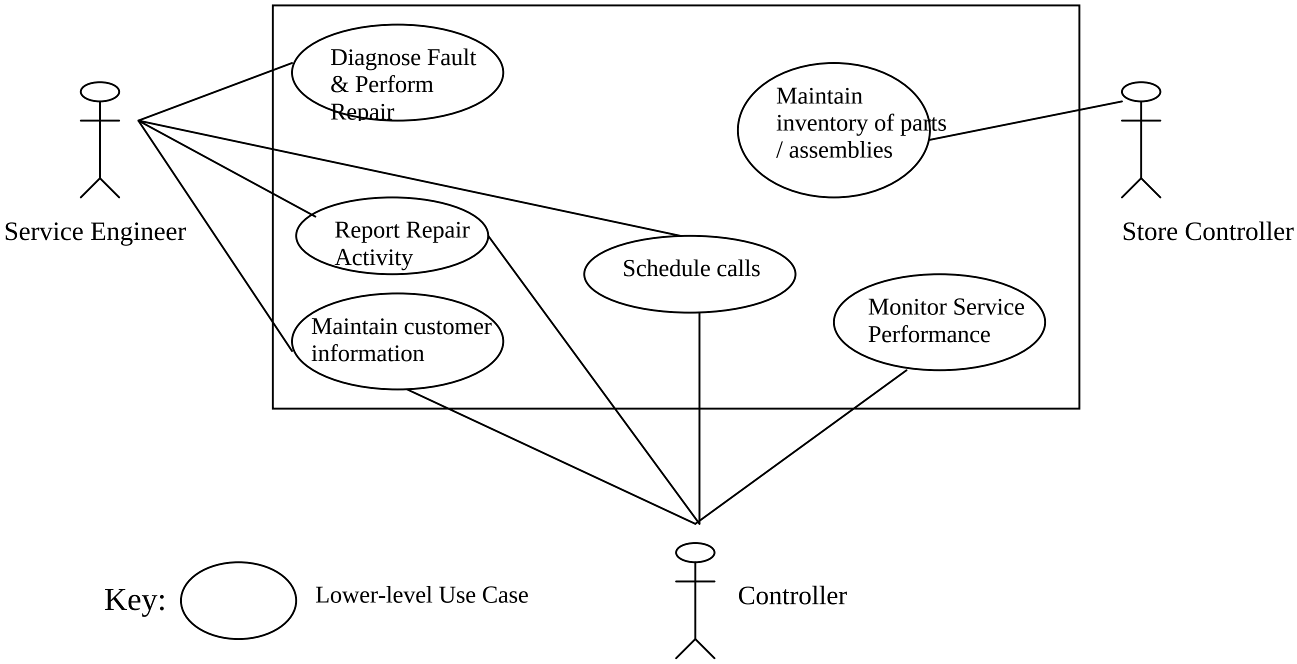

Figure 4 Use Case Diagram of SESS

3 Description of the CREWS-SAVRE Method

3.1 Inbound Events Dependency Analysis

The system environment model and domain scenarios are used for requirements elicitation and validation by a walkthrough technique. Taking each external source of system input in turn, number of different events types that originate from each agent/object are recorded. For instance, a customer may communicate the event types, requests, amends, cancel, agree and pay to a ticket reservation system on a single event flow 'reservations'. Requirements are added to deal with the inbound event. Typical generic requirements are validation routines, calculations, matching input to the database, retrieving data, or invoking an algorithm or sub routine. Specification proceeds by adding requirements for the necessary outputs in response to the input type (see Table I). The following questions are applied to each inbound event-type to ascertain possible system requirements:

- Is there a system function to deal with the input event? If not, then a new functional requirement should be elaborated.
- Does the event require the system to attain a goal state, e.g. an inventory control system tries to satisfy all outstanding orders ? What activity is necessary to attain the desired state ? A process should exist to carry out the necessary change.
- Does the event imply that the system should maintain a goal state, e.g. a thermostat control system has to keep room temperature within 5 degrees of a setting ? If so, then
    - Can the system interpret deviation from the desired state?

- Can the system take remedial action to return to the desired state? Answers to this question suggests functional requirements to monitor normal states and correct deviations. More details on goal-oriented analysis is given in (Sutcliffe & Maiden 1993) and in the KAOS Method (van Lamsweerde et al. 1995)

For each inbound event ask “Can an error occur ?” This question initiates analysis of validation requirements, and alternative paths in use cases. Three main classes of error-events imply the different responses (and hence requirements) by the system:

- Events that should arrive in a set order: In this case the system requirement is to detect the order of event arrival against an expected life history and then take corrective action.
- Events which may arrive in any order: The requirements in this case are to check for the plausibility of events and provide undo facilities to correct user mistakes or warning messages when events appear to be unusual, e.g. an aircraft descends after leaving an airport.
- Unexpected events: The system should continue to function correctly so the requirement is for an exception capture procedure; or a reporting mechanism so that human operators can investigate exceptional events.

If the dependencies between the scenario and components in the intended system are predictable then validation requirements for mal-formed inputs are easily specified; however, if event arrival is random then requirements are more difficult to elaborate. The last part of inbound event validation is to trace the event of the source to define event capture and system access requirements when security or safety are important non-functional requirements.

Table I Event Types and Generic Requirements

Event Type	**Generic Requirements**
Data entry	Validation, input dialogue, create record
Data edit/change	Editor function, locate & update record, validation
Data delete/ cancel	Locate and delete record, warning dialogue
Query	Search function, query dialogue + editor, retrieve and display data
Display data	Display/ print/ output function
Command	Execute function
Receive message	Find addressee, display message
Send Message	Input dialogue, capture address, dispatch message, network protocol

Case study

For the inbound event dependencies one of the use cases - ‘Diagnose fault and repair machine’ is investigated (see description of use case 1 and Figure A in *Appendix*). This assumes a prototype design of SESS that provides information for the engineer. The goal states are to retrieve appropriate information to support the engineer's task of finding out what is wrong with the machine, where the fault is located, removing the faulty component and replacing it with a new part. The input actions and the requirements implications are shown in Table II.

Table II Inbound Event Analysis for the Diagnose Fault and Repair use case

Input (from Engineer) (see Table I for Input Types)	**Requirements implied**	**Output**
Machine type (data query)	Look-up table of Machine types, validation	Display detail of Machine type
Fault code (data query)	Validation, matching fault code to diagnostic advice, data retrieval	Text description of fault, possible reasons for fault
Part/sub assembly code (data entry)	Validation, matching code to repair/maintenance instructions, data retrieval	Text and diagrams for repair/maintenance instructions
Spare parts used- code (data entry)	Validation, update parts inventory	Display parts to be used and inventory
Call sign-off- date and time (Send message)	Validation, date cues	Display call ID, call details, date and time

This analysis only covers the normal course use case in which a repair proceeds to the goal state. Alternative courses need to be added if the engineer can not diagnose or fix the fault. Analysis of requirements implied by inbound events is inextricably linked with the outbound analysis. Also assumptions made in the use case can have radically different implications; for instance, the level of automation is assumed to be an information system rather than a diagnostic expert system. The first event, entering the machine type is a simple input, requiring the system to validate input and to match the input code against a database of machine types. The output requirement in response to this input event is a display of the machine details. The next input is an error code which the engineer reads from the machine's on-board diagnostic system or is inferred from preliminary tests. The requirements implications are for validation, database matching, and information retrieval with output displays of the probable problem causes and advice on repair strategies. The subsequent input events depend on this assumption about the output. For instance, if the system displays a diagram of the machine then the engineer's next input event might be pointing at the part/sub assembly on the diagram rather than entering a part code. This would add a requirement for an event interpreter in the graphical user interface. Assuming a part code is entered, the requirements implications are validation, matching, and data retrieval. The use case proceeds with inbound events to record the parts used with implications for updating the inventory database, and then recording the repair.

This case study illustrates some of the complexities of inbound event analysis. Event sequences depend on the assumptions made about the level of automation and on the user system dialogue. Furthermore, inbound events have many levels of detail as the specification of the user interface proceeds, (e.g. input commands for undo, escape, and responses to erroneous input). Inbound event dependencies are analysed iteratively with outbound events to add for more detailed requirements.

3.2 Outbound Events Dependency Analysis

Using the domain scenarios and the system environment model the following questions are employed to analyse user requirements:

- Which users/stakeholders require information in the scenario and when it will be needed ? The analyst should check that there is a system requirement to produce the information.
- Is the information content of the system output appropriate for the user's task or goal ? The answer to this question may require a detailed task analysis and is beyond the scope of this paper; however, more detail can be found in the Task-based Information Analysis Method (Sutcliffe 1997).
- Which users require information for decision support ? This points to output requirements to provide information or an interactive simulation to help the user's decision making.
- Is the user's goal in the scenario information seeking ? This points out information browsing and retrieval requirements.

The questions help to identify the information dependencies between the users and the system, and in doing so, help to define output requirements. Further questions then focus on who should receive the output information and potential errors of output being lost, directed to the wrong person, arriving too late, in the wrong order etc. A more complete list of questions in given the method tutorial (Sutcliffe and Minocha 1998). System output dependencies are checked using the exceptions depicted in Table III. Exception types are linked to problems that may arise in satisfying non-functional requirements and to generic requirements to deal with these problems.

Table III Output Exceptions and Generic Requirements

Validation Check - Exceptions	Problems / Issues	Generic Requirements / Implications
Agents' ability to respond	timing, capabilities	workload planning /scheduling, training requirements
Agents' willingness to respond	motivation, responsibility, authority	assess responsibilities and power relationships
output arrival timing - too early /late	accuracy, timeliness of information	delivery prioritisation, database update integrity, on demand access

output arrival order - incorrect	accuracy, task relationship	sequential delivery logs, delivery control checks
output information quantity and quality	accuracy, utility, usability	appropriate presentation techniques, (ISO 9241 part 12)

- The next step is to establish whether the users are likely to comply with system commands. Possible impediments are lack of motivation, or commands that clash with their responsibilities. Pointers to answers may be found by tracing the responsibility and authority relationships of the agent.

Responsibility and authority relations can be traced to check whether the system output will fulfil users' goals, and that authority relationships are clear and do not contradict each other (e.g. two agents in charge of one activity, conflict in authority). The agent's properties are inspected to determine whether they have the appropriate capabilities and responsibilities to undertake the tasks they are responsible for.

The validation checks, potential problems arising and generic requirements are illustrated in Table IV. The agent's properties are inspected to determine whether they have the appropriate capabilities and responsibilities to undertake the task or take the decisions they are responsible for. The implications part of the table draws attention to the need to plan workload according to the arrival timing of output, to ensure agents have the appropriate training, and to investigate changes that introduction of the system may have on power and authority relationships among the stakeholders. Generic requirements are suggested for addressing problems that may arise from output exception types.

Table IV Impact Analysis for Coupling and Agents Response

Validation Check	**Problems / Issues**	**Generic Requirements / Implications**
Agent's ability to respond	timing, capabilities	workload planning / scheduling, training requirements
Agent's willingness to respond	motivation, responsibility, authority	access responsibilities and power relationships
Number of system-agent couples	workload, response time prioritisation	assess agents' work schedule and priorities
Volume of events couple	workload	plan agent resources to deal with volume
Quality of system - agent couples	authority, responsibility, brittle systems, control bottlenecks	reduce control couples by increased autonomy

Case study

The same use case is considered and the outbound events are taken from column 3 of Table II. The system is intended to be loaded on a laptop machine that the engineer will take with him/her on the job, hence the first two problems of tracing and losing output are a consequence of forgetting to take the portable or machine failure. The latter three questions are more pertinent as they focus on the acceptability of the data in terms of its accuracy and presentation. The results of the outbound event analysis are given in Table V.

Table V Outbound Event Analysis for the Diagnosis and Perform Repair Use Case

Output event/display	Potential Problems	Generic requirements
Display details of Machine type	accuracy, duplicated information	Check database updating
Text description of fault, possible reasons for fault	accuracy, usability, utility- not being able to find correct information	Presentation of information with diagrams, graphics, clarity of text
Text and diagrams for repair/maintenance instructions	usability and utility- misinterpreting information	Presentation of information with appropriate media, highlight key items -(ISO 14915)
Display parts to be used and inventory	accuracy- database integrity, usability	Highlight key items, structure display
Display call details and date, time.	accuracy, usability	Highlight key items, structure display

Most of the implications for the outbound event analysis are for user interface design and presentation of information. One possible problem arises from inaccurate information in the databases, which reinforces the need for a requirements already stated, to ensure the update integrity of data. Other problems are part of the presentation design requirements. The user requires appropriate information for the task, but the system must not provide too much or too little information. In addition there are further requirements to select appropriate media to display parts of the machine, repair instructions, etc. In this case the method points the designers towards guidelines for information presentation and multimedia design (ISO 14915). The last three outputs are information displays providing feedback for user input, so requirements are for a clear, structured user interface.

System outputs for the call reporting use case (not shown in Table V) have implications for the accountability and authority relationships between the engineer and the controller. Output from the call allocation system must be accurate otherwise it will undermine the controller's authority, likewise the input supplied by the engineer has to be up to date to ensure effective scheduling of service calls.

3.3 Coupling Analysis

Further requirements are discovered by assessing coupling between the required system and its users/stakeholders with the environment model and the use cases. Coupling analysis commences by a qualitative analysis using the scale illustrated in Table VI. Events flowing between the system and user agents, represented in the interaction diagram of the top level use case, are counted and each event is assigned a coupling factor. Information coupling is low and makes few impositions on users; however, command coupling places more constraints on actions depending on the type of command. Commands may constrain an agent's freedom to act or take decisions. For instance, the system might set a strategy that dictates how stakeholders must act. Command couples are rated for the strength of the obligation they impose on the recipient agent (optional, mandatory commands, commands with constraints) and the restriction on freedom of action imposed on the recipient. Where high levels of command coupling are apparent these indicate areas of possible conflict and system failure that should be investigated.

Table VI Coupling Analysis Levels

Weighting	**Event Flow- Input (I) or Output (O)**	**Implications for users / stakeholders**
1	O- discretionary information	discretionary use
2	I- discretionary input	discretionary input- may lead to performance problems
3	O- decision-related information	agents needs information to take decisions
4	O- essential information	necessary for task or user action
5	O- indirect command	warning or message that requires attention, and possibly action
6	I- mandatory input	system needs data to continue
7	I- report command	agent must report completion of a task
8	O- command action	agent must carry out an action
9	O- command + constrained actions	agent's way of working is controlled by the system
10	O- command + multiple constraints	command dictates the type/ sequence of another agent's actions
11	O- command + constraints on many agent's actions	one agent's command controls several other agents

Where high levels of command coupling are apparent these should be investigated to reduce commands where possible. Table VII summarises some implications of pathological coupling and possible remedies. When coupling scores are high, commands imposed by the system on the user should be reduced; for instance, by reallocating the work so only the user is responsible. Increasing autonomy of agents and decomposing the system into sub systems and also reduce coupling. This analysis is carried out with use cases and by tracing relationships on the system environment model.

Table VII Coupling Analysis Implications

Problem	**Possible Solutions / Requirements**
Agents ability to respond to commands	check agent's responsibilities, capabilities, workload
Agent's willingness to obey commands	investigate agent's motivation, responsibility and accountability
High coupling scores	Reduce commands, increase local decision making, sub-divide system
Many constrained commands	increase user training, increase local responsibility, control by objectives
Many report commands	monitor by objectives, gather events automatically, increase local responsibility
Many essential inputs	use defaults, reference files, integrate and share databases
Many commands / agents	investigate timing and schedule work, split responsibility
One agent creates many commands	review responsibility and authority, investigate work schedule

A large number of constrained commands indicates that users are not being trusted to carry out their work without direction. Increased training and giving users responsibility should be considered. A large number of reporting commands indicates excessive system monitoring, the necessity of this should be questioned to see whether it benefits the users. Analysis of the convergence of many commands on one agent should trigger a review of responsibilities, workload, and operational schedule.

Case study

Two scenarios are analysed, each represents a different management policy for controlling the system. In the first scenario, control is centralised and all customer calls are sent to a controller who allocates calls to service engineers. The engineers have to report their location and availability to the controller, as well as their progress when undertaking repairs, and other activities such as replenishing spares from the stores. The controller schedules the work of the engineers who have to obey commands. Coupling analysis for the reporting use case for this scenario is shown in Table VIII.

Table VIII Coupling analysis for the controller scenario version of the SESS

Controller-engineer Event	**Coupling**	**Controller-system Event**	**Coupling**
Availability- I	6	Eng availability- I	6
Location -I	6	Eng location -I	6
Call schedule—O	8	Customer call-I	6
Arrive at customer- I	6	Engineer-call allocation-O	9
Start job- I	6	Update progress-I	6
Finish job-I	6	Report complete-I	6
Report Stock - I - replenishing	6	Update engineer status-I	6
Total	44	Total	45

In this scenario the coupling between the engineer and the controller (see Figure B in the *Appendix*) and between the controller and the automated call dispatch system (see Figure C in the *Appendix*) is high. The system commands for call allocation constrain the controller's choice, and in turn the schedule is passed onto the engineer who has no discretion in his or her work. Coupling between the controller and the system could be reduced from 45 to 33 if the system's function is changed to a decision support role (see Figure D in the *Appendix*) in which the system displays the engineer's status, locations and customer calls, leaving the controller to decide on the allocation. This reduces the Engineer call allocation coupling to 3 (discretionary decision) and makes the 'Update progress input' unnecessary, hence saving 12 points overall. However, the coupling between the engineer and the controller would remain the same unless some autonomy is granted to the engineers. The high coupling in this scenario indicates a possible brittleness in the system, and more autonomy is desirable. Furthermore, the controller has to handle all the engineers in a branch and this indicates a problem of too many commands converging on one agent.

In the second scenario, a decentralised system is investigated. Small groups of engineers co-operate to handle customer calls in their area. Each engineer is assigned a district and customers call the engineer they have been assigned to directly (see Figure E in the *Appendix*). When an engineer receives more calls than he can deal with he posts them onto a bulletin board which is shared with all engineers in the branch. An engineer with no calls who is reasonably close by is expected to take the call, otherwise the customer has to wait. Limited reporting of completed calls is carried out for management statistics. The coupling analysis for this scenario is shown in Table IX (see Figures E and F in the *Appendix*).

Table IX Coupling analysis for the workgroups scenario version of SESS

Engineer - customer events	Coupling	Engineer - bulletin board events	Coupling
Call-I	6	Request -I	2
Acknowledge-O	4	Job list-O	3
Sign-off- O	4	Report job	2
Total	14	Total	7

In the decentralised scenario, local autonomy, coupling between the customer and the engineer, and between the engineer and the supporting bulletin board is low. Coupling between the service engineer and the controller (not illustrated), who now has a management supervisory role would also be low as this is restricted to discretionary reporting and mandatory reporting of jobs completed (coupling total 4). This scenario contains an incentive system for the engineers to complete jobs as quickly as possible and this increases coupling, but not to the extent of the first centralised control scenario.

The dramatic differences between the two scenarios demonstrate the work organisations that are possible for this system. The coupling analysis shows that a decentralised approach would be more flexible and imposes fewer restrictions on the service engineer's job. However, coupling analysis is no panacea for work design on its own. The advantages of the lower coupling in the second scenario would need to be assessed in light of engineer performance, resource costs and customer satisfaction.

The two scenarios have different requirements for technical system support. The first, centralised control scenario implies functional requirements for an automated call allocation system, a matching algorithm and an accurate database of engineers' location, work activity and training. This in turn necessitates a call reporting and progress tracking system. In contrast the second scenario requires a simpler system composed of an electronic bulletin board to record calls that engineers can not deal with and a limited reporting system to capture details of completed calls for management reports and the incentive scheme. In our ongoing work we are investigating how different patterns of coupling and organisation design can be linked to appropriate requirements templates of co-ordination and workflow systems.

4. Discussion

The analytic techniques described in this paper provide the basis for systematically investigating socio-technical system requirements. These build on the i* framework (Yu 1994) that analyses requirements by reasoning about relationships between agents, tasks and goals. The metric based approach we have adopted complements the i* style analyses. One advantage of the metrics is that they can be applied to high level scenarios of prospective system designs to establish the strengths and weaknesses of different options. Coupling analysis has many applications in organisation design that we are only beginning to explore, such as span of command, and different command structures in organisations (Robbins 1990). In our future work we will incorporate this analysis into design rationale representations so trade-offs can be assessed in quantitative as well as qualitative terms.

The other contribution of this paper is to suggest how generic requirements could be linked to metric based analyses. Generic requirements (GRs) by their nature are not detailed, hence the utility of such advice needs further evaluation; however, we believe that GRs add value by raising requirements issues, even if they do not always provide solutions. The method spans a wide range of issues which GRs can not deal with in depth, so we see the method as a framework that points the requirements engineer towards other sources for more detailed advice including generic models of requirements for classes of application which we have partially explored for information retrieval (Ryan & Sutcliffe 1998).

The coupling analysis draws on theories of autonomy and work organisation from management science (e.g. Clegg et al. 1997). High level requirements for groupware and workflow systems can be proposed as a result of this analysis, however, we have to improve the connection between generic requirements and organisational design. Furthermore, the recommendations of coupling analysis need to be interpreted in a domain context. While increasing autonomy might help many business organisations the converse may be true for military command and control systems.

Acknowledgements

This research has been funded by the European Commission ESPRIT 21903 long term research project 'CREWS' - Co-operative Requirements Engineering With Scenarios. The project partners include RWTH-Aachen (project co-ordinator), City University, London, University of Paris I, France, FUNDP, University of Namur, Belgium.

References

(Clegg et al.1997) C. Clegg, C. Axtell, L. Damodran, B. Farbey, R. Hull, R. Lloyd, J. Nicholls, R. Sell and C. Tomlinson, 'Information Technology: A study of performance and the role of human and organisational factors', Ergonomics, vol. 40, no. 9, pp. 851-871, 1997.

(DeMarco 1978) Tom DeMarco, Structured Analysis and Systems Specification, Englewood Cliffs, New Jersey, Prentice Hall, 1978.

(Harker et al. 1993) S. D. P. Harker, K. D. Eason and J. E. Dobson, 'The Change and Evolution of Requirements as a challenge to the practice of Software Engineering', IEEE Int. Symposium on Requirements Engineering (RE'93), pp. 266-272, Jan. 1993.

(Hsi & Potts 1995) I. Hsi and C. Potts 'Towards Integrating Rationalistic and Ecological Design Methods for Interactive Systems', Georgia Institute of Technology, Graphics, Visualisation and Usability Centre *Technical Report*, 1-15, 1995.

(ISO 9241) ISO 9241, Ergonomic requirements for office systems visual display terminals, Parts 10, 11, 16 International Standards, parts 1-9, 12-15, 17, draft standards;

International Standards Organisation, Switzerland, available from National Standards Organisation, 1996.

(ISO 14915) ISO 14915-1 Multimedia User Interface Design - Software Ergonomic Requirements - Part 1: Introduction and Framework, Part 3: Selection of Media and Media Combination, Committee Drafts, September 1998.

(Jacobson 1992) I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, *'Object-Oriented Software Engineering: A Use-Case Driven Approach'*, Addison-Wesley, 1992.

(Lubars et al. 1993) M. Lubars, C. Potts and C. Ritcher, 'A review of the state of the practice in Requirements Modelling', IEEE Int. Symposium on Requirements Engineering (RE'93), pp. 2-14, 1993.

(Macaulay 1996) L. Macaulay, Requirements Engineering, Springer Verlag, Berlin, 1996.

(Mintzberg 1979), H. Mintzberg, 'The Structuring of Organisations', Prentice-Hall Inc., 1979.

(Potts et al. 1994) C. Potts, K. Takahashi and A. I. Anton, 'Inquiry-Based Requirements Analysis', IEEE Software, vol. 11, no. 2, pp. 21-32, 1994.

(Robbins 1990) S. P. Robbins, 'Organisation theory', Prentice Hall, Englewood Cliffs, NJ.

(Ryan & Sutcliffe 1998) M. Ryan and A. G. Sutcliffe, 'Analysing Requirements to Inform Design', People and Computers XIII - Proceedings of the BCS-HCI Conference, Sheffield, H. Johnson, L. Nigay and C. Roast (Eds.), Springer Verlag, pp. 139-158, 1998

(Sommerville & Sawyer 1997) I. Sommerville and P. Sawyer, 'Requirements Engineering: A Good Practice Guide', John Wiley & Sons, 1997.

(Sutcliffe & Maiden 1993) A. G. Sutcliffe and N. Maiden, 'Bridging the Requirements Gap: Policies, Goals and Domains', Proceedings of the 7th International Workshop of System Specification & Design, pp. 52-55, 1993.

(Sutcliffe 1997) A. G. Sutcliffe, 'A Task-Related Information Analysis', International Journal of Human Computer Studies', vol. 47, pp. 223-257, 1997.

(Sutcliffe & Minocha 1998) A. G. Sutcliffe and S. Minocha, 'CREWS-Scenarios for Acquisition and Validation of Requirements - The Method', Tutorial, 1998.

(UML 1997) UML, 'Unified Modelling Language: Method', Rational Corporation, 1997. Rational's web site: http://www.rational.com

(van Lamsweerde et al. 1995) A. van Lamsweerde, R. Darimont and P. Massonet, 'Goal directed elaboration of requirements for a meeting scheduler: problems and lessons learnt', IEEE Int. Symposium on Requirements Engineering (RE'95), pp. 194-203, 1995.

(Yu & Mylopoulos 1994) E. Yu and J. Mylopoulos, 'Towards modelling strategic actor relationships for information systems development - with examples from business process reengineering', Proceedings 4th workshop on Information Technology and Systems (WITS'94), Vancouver, B.C. Canada, December 17-18, 1994.

(Yu 1994) E. Yu, 'Modelling Strategic Relationships for Process Reengineering', *Technical Report DKBS-TR-94-6*, University of Toronto, 1994.

Appendix

Use Case 1: Diagnose Fault and Repair

1. The engineer enters the code for the machine type. 2
2. The system displays details of machine type.
3. The engineer enters the code of the fault.
4. The system displays diagnostic information for the fault.
5. The engineer enters code of the sub-assembly.
6. The system displays instructions for repair.
7. The engineer enters code of the part to be used.
8. The system displays the availability details of the spare part.
9. The engineer collects the spare part from the van.
10. The engineer performs the repair.
11. The engineer checks the machine performance.
12. The engineer logs off from the call.
13. The system displays the call ID, call details, date and time.

Figure A Interaction Diagram for the Use Case 1

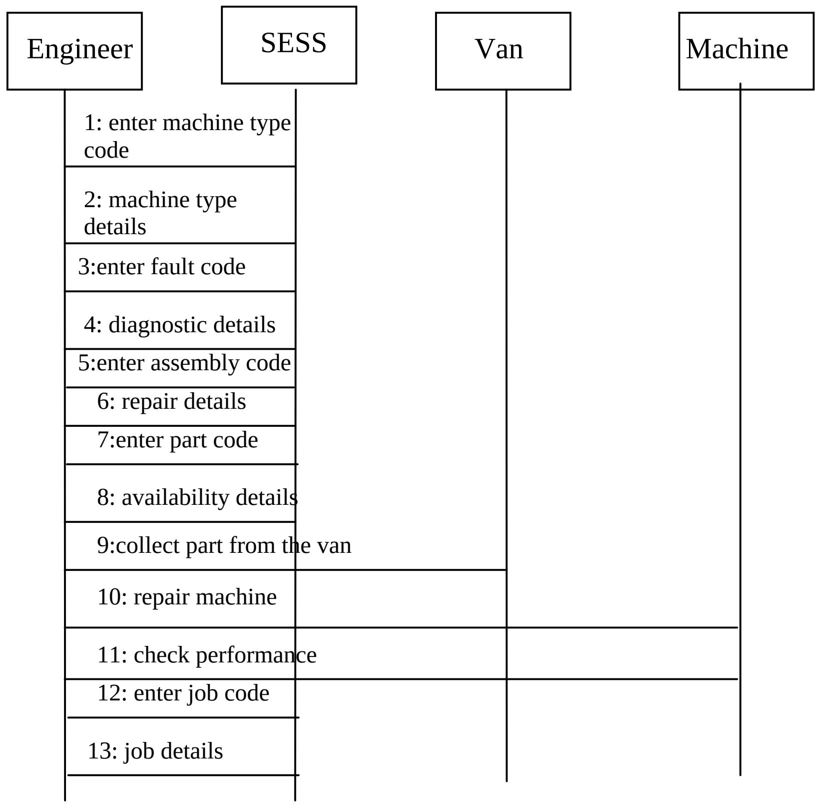

Figure B Interaction Diagram for the Use Case: Reporting to the Controller

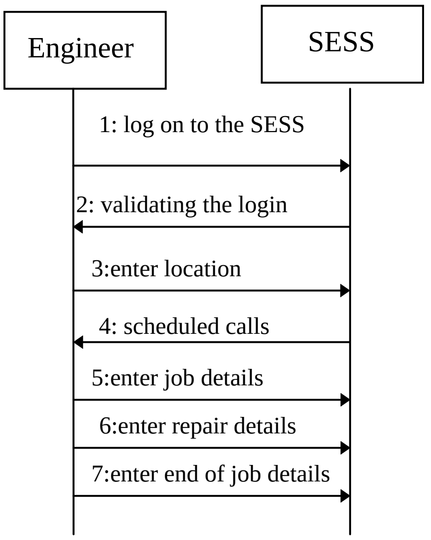

Figure C Interaction Diagram for the Use Case: Allocation of a call to the engineer by SESS

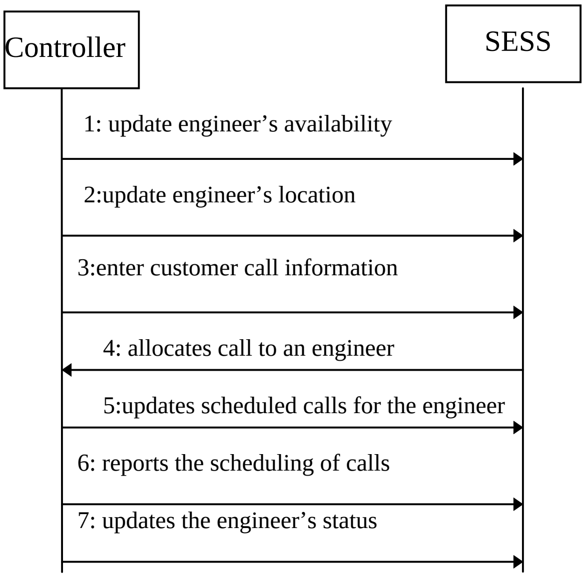

Figure D Interaction Diagram for the Use Case: Allocation of a call to the engineer (SESS as a decision support system)

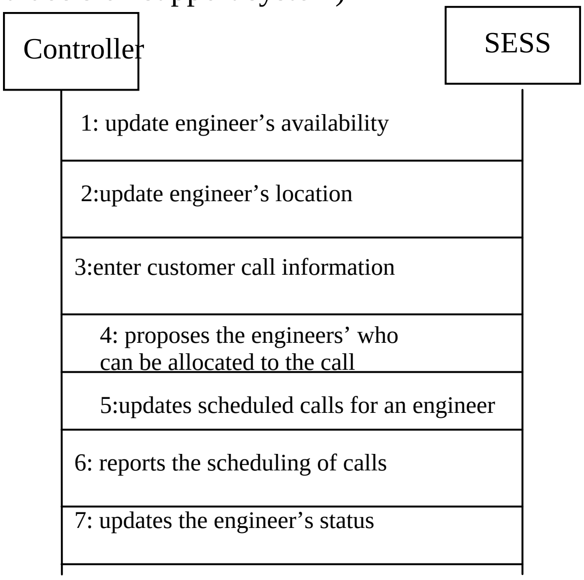

Figure E Interaction Diagram for the Use Case: Service Engineer attends to customers' calls

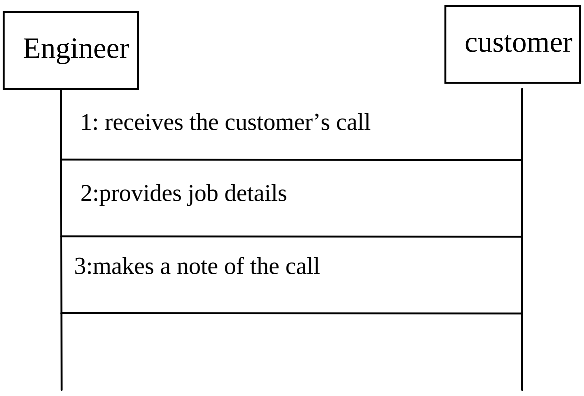

Figure F Interaction Diagram for the Use Case: Service Engineer accesses Bulletin Board