A Collaborative Learning Experience in Modelling the Requirements of Teleoperated Systems for Ship Hull Maintenance[1]

Joaquín Nicolás, Joaquín Lasheras, Ambrosio Toval
Software Engineering Research Group. Departamento de Informática y Sistemas.
Universidad de Murcia. Campus de Espinardo. 30071 Murcia (Spain)
{jnr, jolave, atoval}@um.esmailto:atoval}@um.es

Francisco J. Ortiz, Bárbara Álvarez
Systems and Electronic Engineering Division. Universidad Politécnica de Cartagena.
30202 Cartagena (Spain)
{francisco.ortiz, balvarez}@upct.es

Abstract: This paper presents a join experience in modelling the requirements of the product line of teleoperated systems for ship hull maintenance, which are basically robotic systems used for ship maintenance operations, such as cleaning or painting the ship hull. It is proposed to specify the product line requirements through a feature model, a conceptual model, and a use case model, which together allow domain understanding, derivation of reusable product line requirements, and efficient decision-making in the specification of new systems developed in the product line. Action Research, a qualitative research method in software engineering, has been applied to define the collaborative research process.

Key Words: Domain Analysis, Product Lines, Requirements Reuse, Teleoperated Systems, Feature Modelling
Categories: D.2.1, D.2.13

1 Introduction

This paper presents an experience on modelling the requirements of the product line of the teleoperated systems for ship hull maintenance (TOS hereafter) through a collaborative learning process carried out by two Spanish research groups of different but complementary fields: the Systems and Electronic Engineering Division (SEED) of the Polytechnic University of Cartagena and the Software Engineering Research Group (SERG) of the University of Murcia.

In recent years, SEED has gained considerable experience in developing software reference architectures in the TOS domain [Fernández et al., 2004]. To date, SEED has paid less attention to requirements reuse in this product line than to architectural components reuse.

1 Partially financed by the CICYT (Science and Technology Joint Committee), Spanish Ministry of Science and Technology, project *DYNAMICA (*DYNamic and Aspect-OrienteD Modeling for Integrated Component-based Architectures, TIC2003-07804-C05).

Independently, SERG has defined a reuse-based requirements engineering method and has developed catalogues of reusable, textual requirements in security [Toval et al., 2002a] and personal data protection [Toval et al., 2002b] domains.

An increment in the abstraction level of knowledge management could improve future reuse (see for example [Cybulsky and Reed, 2000]), so that collaboration between SEED and SERG appears in a natural way. The goal is to obtain a TOS domain model to accelerate domain knowledge acquisition and to facilitate the specification of new systems in the product line and the reuse of architectural components.

Experience of SERG with the personal data protection and security requirements catalogues is insufficient. Both domains can be considered well structured, if the sources of requirements considered are, respectively, Spanish and European legal documents related to personal data protection, and a well documented method for risk analysis and management in Spanish public administration. This is why it was relatively straightforward (in an *ad hoc* manner) to structure the knowledge of these domains into two textual requirements catalogues. However, what do we do when the problem domain is complex and knowledge is not previously structured, as occurs in the TOS? In these cases an ad hoc approach is not enough to obtain a quality requirements catalogue. Hence, the need for a better representation and organisation of the knowledge in wide, complex and slightly structured domains appears.

This work presents a modelling of the TOS product line requirements. Several existing domain analysis techniques have been adapted in this approach, which basically consists of: (1) a feature model, as a high level interface that favours the reuse of the product line requirements; (2) a conceptual model, which leads to greater domain understanding; and (3) a generic use cases model, enabling descriptions of the scenarios related to the execution of certain functional features in the feature model. The use of features and generic use cases makes it possible to capture variability in the product line. A qualitative research method in software engineering, Action Research, has been applied to define the research process formally.

This paper is structured in the following way: Section 2 shows how the collaboration between SEED and SERG has been designed by means of Action Research. Section 3 briefly outlines the TOS domain. Section 4 discusses the general approach to the case study, and Sections 5, 6 and 7 show feature, conceptual and use case models. Finally, conclusions and future works are given in Section 8.

2 Research framework

This contribution reports the results of an experiment designed to obtain a TOS domain model. Action Research [Baskerville, 1999], one of the most well known qualitative research methods in software engineering, was used to design the experiment. The application of Action Research produces a cyclic process in which all the parts involved in the research participate, examining the existing situation with the intention of changing and improving it. Action Research is a valid approach for studying the effects in human organizations of specific changes in systems development and maintenance methods.

In line with the Action Research terminology, the following roles have been considered in this experiment:

- The *researcher* is the SERG.
- The *researched* is the TOS product line, from a domain analysis point of view.
- The *critical reference group* (CRG) is SEED, i.e. the researched for (in the sense of having the problem the research is to solve). According to Action Research, the CRG has to participate in the research process too, although it can be involved less actively than the researcher.
- The *stakeholders* are all those organizations that might benefit from the results of the research: in this case, the CRG and, in general, companies that perform ship hull maintenance tasks.

The activities performed in each cycle of Action Research can be summarised as follows:

- To begin, a *planning* is made, in which the questions to guide the research are identified and the actions to solve those questions are specified. In this case study, first the interest of performing an analysis of the TOS domain was justified and then the state of the art in the domain analysis field was studied in order to propose an approach to tackle the problem.
- An *action* task follows, in which the researcher takes part in the real situation through a careful, deliberate, and controlled variation of the practice. In this case study, the researcher, together with the CRG, built an initial model of the TOS.
- Then, an *observation* or *evaluation* is made, in which information on the effects of the action is collected.
- The cycle ends with a *reflection*, in which the results are shared and analysed by all the interested parts, and new important questions can be raised which can be tackled in a new cycle of Action Research.

3 Teleoperated Systems for Ship hull Maintenance

The global objective of the European project EFTCoR (*Enviromental Friendly and Cost-Effective Technology for Coating Removal,* Fifth Framework Programme) [Fernández et al., 2004], carried out by SEED, is the development of a new technology for ship coating removal. The project tries to solve a critical problem in the European naval industry: the preparation of the hull surface for painting in an environmentally friendly way.

TOS consist of the following subsystems: the *Robotic Device Control Unit* (RDCU), which controls the devices used for coating removal; the *monitoring subsystem*, which carries out the tasks of management of information related to ship maintenance; the *vision subsystem*, which performs the visual analysis of the hull; and the *recycling subsystem*, which is in charge of removing wastes from the work area and recycling it.

4 A Framework for Domain Analysis

In planning the collaboration, the most important decision was to base modelling on features (cf. FODA [Kang et al., 1990], FORM [Kang et al., 1998] or PLA [Kang et al., 2001]), which intuitively specify the vision of the product line that the stakeholders have. With features the domain can be explored quickly in order to know the main issues and the common and variable points. In addition to the feature model, the following have also been used in the modelling of the TOS:

- A *conceptual model*, showing the concepts of the domain and their relationships.
- A *use case model*, describing in detail the interactions between the external actors and the system that can occur during the execution of some functional features of the feature model.

In contrast to previous experiences [Toval et al., 2002a, Toval et al., 2002b], the structure of the TOS catalogue of requirements does not consist of a list of (mainly textual) requirements which are structured in a document hierarchy. Now the requirements are directly structured in the catalogue starting from the feature model: features are related to use cases and non-functional requirements through traceability links. By using features instead of natural language requirements, a more agile reasoning –without considering the location of the requirements in the requirements documents– is sought. Thus, during the specification of a product belonging to the product line, customers can “navigate” in the “problem space” through the “decision space” that features make up, selecting one feature or another, in an easier way than scanning a requirements list in order to select one requirement or another. Moreover, during the initial requirements specification, it is more convenient to perform a feature analysis than to write quality textual requirements, which have to be unambiguous, complete, consistent, and verifiable, as the IEEE 830 standard establishes.

5 Feature Model

The functional and non-functional capacities and the technology constraints that can appear in the products of the product line are modelled in the feature model. This feature model plays an essential role in the domain analysis of the TOS: clients usually specify their necessities intuitively in terms of the “features that the new system has to have or provide”, understanding these as abstractions of the capacities of the system that must be implemented, proven, given and maintained [Kang et al., 1998]. Both clients and developers can intuitively understand the feature model.

In this case study, the feature model defined in FORM has been adopted, and extended as follows:

- With the goal of simplifying the feature model, the four layers of features of FORM (whose complexity is criticised in [Trigaux and Heymans, 2003]) have been reduced to only two: capability and implementation. The latter gathers the original layers of “operating environment”, “domain technology", and "implementation technique", which are very close and sometimes seem to overlap, giving rise to confusion.

- In [van der Maβen and Lichter, 2002] the requirements are described which any notation must satisfy to model variability in a product line. Subsequently, [Trigaux and Heymans, 2003] add another requirement: the graphical representation of variability, and in particular, of variants, variation points and cardinalities of variation points. With the goal of satisfying all the requirements, the original notation of FORM has been extended with the graphical representation of variation points (as per [Griss et al., 1998, van Gurp et al., 2001]) and cardinalities (following [Riebisch et al., 2004]).

A part of the feature diagram is shown in Figure 1, where the services offered by the family product are reflected. The relationships between the layers of *capability* and *technology* are specified through the *implemented-by* traces, such as the one that shows the types of technology used to make cleaning (*Sand-Blasting* and *Hydro-Blasting*). This *implemented-by* trace also represents a variation point within the product line, identified as *Coating Removal Technology*. The cardinalities of the variation points can be also observed in the diagram: in this case study, only alternative features of which at least one must be chosen have been identified, so each has cardinality 1.

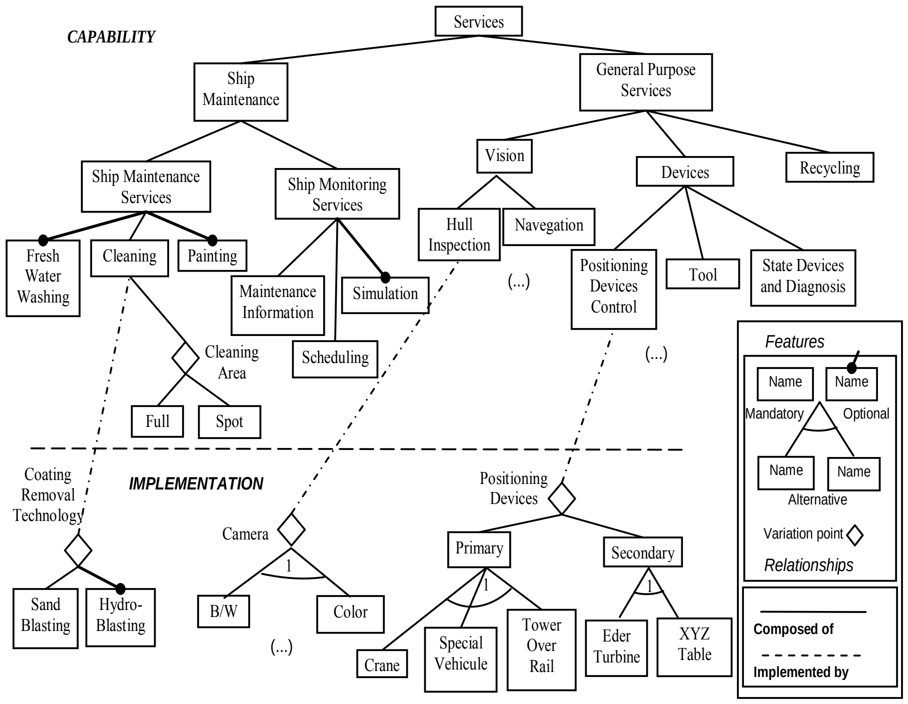

Figure 1. A part of the feature model.

Each feature is described textually by means of a template (Table 1). Then it is presented the textual description of the *Spot* feature (see Figure 1), which belongs to the capability layer and is mandatory.

FEATURE *Spot* (capability, mandatory)
Description: cleaning performed only in isolated points of the surface of the ship hull.
Rationale: It is often necessary when the cleaning of the complete hull (*full-cleaning*) is not of interest or it can not be performed, because it is more expensive or there are parts of difficult access.
Composition rules: Does not have (*i.e., does not break down into other ones*)
Binding time: at definition time
Trade-offs: Requires *Primary* AND *Secondary*. It is necessary to have secondary and primary positioning systems, since the primary positioning system is not accurate enough by itself.
Trace to requirements: *Spot Cleaning* use case

Table 1. "Spot" feature.

6 Conceptual Model

The complex TOS domain was alien to SERG, so a conceptual model was first used to begin modelling. A classic vision in software engineering (described e.g. by [Larman, 2005]) was used, according to which the conceptual model describes the vocabulary of the domain, i.e. the concepts of the problem space and the relationships between them. Requirements (features and use case) must have as "direct objects" the concepts of this "information model".

A part of the conceptual model for the RDCU within the EFTCoR system is presented in Figure 2, including interactions between the RDCU and the other subsystems.

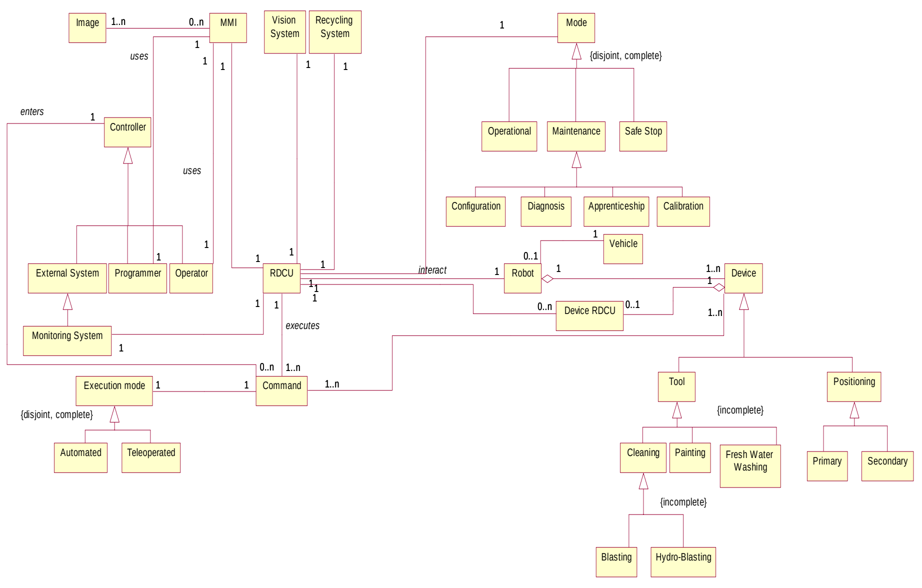

Figure 2. A part of the RDCU conceptual model.

7 Use Case Model

The execution of some functional features of the feature model can be naturally specified as a use case: when a system's actor requires the execution of a feature to obtain an observable value, causing a set of interactions with the system that can be captured within a use case (e.g. *Cleaning* and *Tool calibration* features of Figure 1). An N:M relationship between features and use cases can be established: for example, the *Cleaning* feature can be related to several use cases in Figure 3 (*Full Cleaning* and *Spot Cleaning*), and the *Spot Cleaning* use case can be related to several features (*Cleaning, Spot, Automatic, Semi-Automatic* and *Teleoperated*).

The use case model is structured through features, expressing with these the variation points of the product line, and thus avoiding as much as possible an overload of the use case model with the complexity associated to the product line variability. However, there is variability intrinsic to use cases in a product line –that associated to the possible variations in the steps of the different scenarios captured in the description template of the use case–.

After reviewing different approaches to capture the variability of a product line in the use cases [Gomaa and Shin, 2002, John and Muthig, 2002, Halmans and Pohl, 2003, van der Maβen and Lichter, 2002, Eriksson et al., 2004], that of [Eriksson et al., 2004] has been adopted, since it is in line with the focus previously described, and because it proposes the use of two interesting techniques:

- *Change cases,* which capture the possible impact in the use cases of the adoption of future, anticipated extensions of the system that are still unavailable. Such possible changes are grouped in special use cases denominated change cases [Ecklund et al., 1996]. What use cases can be affected by each change case is indicated by means of the *impact link* trace relationships. For example, Figure 3 shows a change case, *Hydro-Blasting*, implying a change in the cleaning technique used (*Sand-Blasting* until now).
- *Modelling of the variability in the description of use cases*, using (1) (local and global) *parameters* in the description of the use case, and (2) an extended version of the textual description of the black box flow of events of the RUP SE (*Rational Unified for Process Engineering Systems*) [Cantor, 2003], in which the steps of the scenario where variation can appear are expressed in a special notation (see Table 2). Firstly, the description of the flows of events is made considering the system as a black box (Table 2). Later, the description is reviewed and each black box step is replaced by a sequence of white box steps, showing the interactions of the different subsystems to support each black box step.

In order to make the use case diagram more legible, and following [Gomaa, 2005], optional and alternative use cases are labelled with the «optional» and «alternative» stereotypes. The details of the variability are specified in the feature model.

It can be observed that several steps are alternatives in the textual description of the use case in Table 2 (they use the same number), evincing that the action can be made in a teleoperated or automatic form: a step 2 would be traced to the

Teleoperated feature and the other step 2 to *Automatic* (analogously to step 3). Notice that trace relationships can be established between complete use cases –or steps in use cases– and features. Moreover, optional steps such as (4) are also captured. In addition, a global variable ($MAX_TIME_BUTTON) together with two local ones (@MAX_TIME_TOOL, @MAX_TIME_SAFE_STOP) have been used to express the maximum response time to certain actions within the use case.

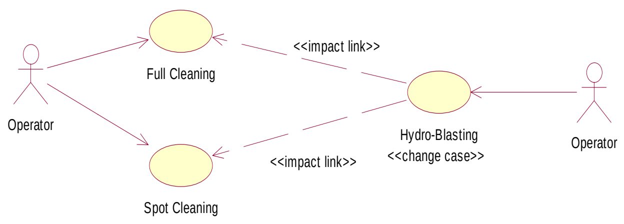

Figure 3. A part of the use case diagram.

Step	Actor Action	Black box	Black Box Budget Requirement
1	This use case starts when the operator pushes the cleaning button.	The system is started to carry out the cleaning operation	Max. response time is $MAX_TIME_BUTTON
2	The operator sees images of the ship hull surface and executes commands to place the tool in the cleaning area.	The tool is placed in the cleaning area.	
2	The vision system executes commands of the positioning systems to place the tool in the cleaning area.	The tool is placed in the cleaning area.	
3	The operator pushes the button to activate the tool.	The cleaning tool is activated.	Max. response time is @MAX_TIME_TOOL
3	The system activates the cleaning tool.	The cleaning tool is activated.	Max. response time is @MAX_TIME_TOOL
(4)	The operator pushes the button of emergency stop.	The system stops securely.	Max. response time is @MAX_TIME_SAFE _STOP
5	The operator pushes the cancellation button.	The cleaning stops at that point.	

Table 2. Complete description of the "Spot Cleaning" use case.

8 Conclusions and Further Work

An experience of cooperative research work has been presented where a reusable requirements catalogue of the TOS product family has been obtained. The CRG affirms that the documentation of requirements generated for the product line has an added value through its spreading to clients and developers, given that it is not usually developed in this type of systems.

According to the experience of the researcher in the field of personal data protection and security, one risk in the elaboration of a reusable requirements catalogue for a wide domain is that a catalogue can be obtained, formed by a long list of textual requirements, which may be correct and very precise but as the same time difficult to handle. In order to avoid this problem, a feature model has been incorporated into the catalogue, which can be used as the starting point to structure it. This model acts as an interface which facilitates requirements selection, permitting an intuitive navigation through the space of the problem through the feature model. The natural integration between features and use cases has been seen.

The graphical representation of the variation points in the feature model −extending the FODA and FORM notations− has been useful to stress the decisions that have to be taken in the instantiation of the family. Nevertheless, the use of a notation for the cardinalities has not been crucial: in this respect the FODA and FORM notations would have been enough in this case study.

The CRG considers that the approach would be more useful with a tool to navigate easily through the feature model and to manage its graphical complexity, which quickly makes it difficult to handle.

A possible line of further work consists of assessing the adoption of a process model to define, develop and maintain the requirements of a product line, like for example that proposed by PuLSE (*Product Line Software Engineering*) [PuLSE].

References

[Baskerville, 1999] Baskerville, R. L. *Investigating Information Systems with Action Research,* Communications of the Association for Information Systems, **2**. 1999

[Cantor, 2003] Cantor, M. *Rational Unified Process for Systems Engineering. Part III: Requirements analysis and design*. 2003. http://www-128.ibm.com/developerworks/rational/library/content/RationalEdge/oct03/m_rupse_mc.pdf

[Cybulsky and Reed, 2000] Cybulsky, J. and Reed, K. *Requirements Classification and Reuse: Crossing Domains Boundaries,* 6th International Conference on Software Reuse (ICSR'2000), Vienna 2000

[Ecklund et al., 1996] Ecklund, E., Delcambre, L. and Freiling, M. *Change cases: use cases that identify future requirements,* ACM SIGPLAN Notices, **31,** 342 - 358. 1996

[Eriksson et al., 2004] Eriksson, M., Börstler, J. and Borg, K. *Marrying Features and Use Cases for Product Line Requirements Modeling of Embeded Systems,* Fourth Conference on Software Engineering Research and Practice in Sweden (SERPS'04), 2004

[Fernández et al., 2004] Fernández, C., Pastor, J. A., Sánchez, P., Álvarez, B. and Iborra, A. *Co-operative Robots for Hull Blasting in European Shiprepair Industry,* Robotics and Automation Magazine (RAM). 2004

[Gomaa, 2005] Gomaa, H. *Designing Software Product Lines with UML: from Use Cases to Pattern-Based Software Architectures,* Addison-Wesley. 2005

[Gomaa and Shin, 2002] Gomaa, H. and Shin, M. *Multiple-view meta-modeling of software product lines,* Eighth International Conference on Engineering of Complex Computer Systems, 2002

[Griss et al., 1998] Griss, M., Favaro, J. and d'Alessandro, M. *Integrating Feature Modeling with the RSEB,* Fifth International Conference on Software Reuse, Vancouver, Canada 1998

[Halmans and Pohl, 2003] Halmans, G. and Pohl, K. *Communicating the variability of a software-product family to customers,* Software and Systems Modeling, 2003

[John and Muthig, 2002] John, I. and Muthig, D. *Product Line Modeling with Generic Use Cases,* International Workshop on Requirements Engineering for Product Lines (REPL'02), 2002

[Kang et al., 1990] Kang, K., Cohen, S., Hess, J., Novak, W. and Peterson, A. *A. Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-021, ADA235785).* Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. 1990

[Kang et al., 2001] Kang, K., Chastek, G. and Donohoe, P. *Product Line Analysis:A Practical Introduction.,* Software Engineering Institute. Carnegie Mellon University. 2001

[Kang et al., 1998] Kang, K. C., Kim, S., Lee, J., Kim, K., Kim, G. J. and Shin, E. *FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures.,* Annals of Software Engineering 5, 5**,** 143-168. 1998

[Larman, 2005] Larman, C. *Applying UML and Patterns,* Prentice-Hall. 2005

[PuLSE] PuLSE. *Methodology (Product Line Software Engineering)* Fraunhofer IESE (Institut Experimentelles Software Engineering). http://www.iese.fhg.de/PuLSE/. Last access: January 2006

[Riebisch et al., 2004] Riebisch, M., Streitferdt, D. and Pashov, I. *Modeling Variability for Object-Oriented Product Lines,* ECOOP2003 - Workshop on Modelling Variability for Object-Oriented Product Lines, Germany 2004

[Toval et al., 2002a] Toval, A., Nicolás, J., Moros, B. and García, F. *Requirements Reuse for Improving Information Systems Security: A Practicioner's Approach,* Requirements Engineering Journal. Springer, **6,** 205-219. 2002a

[Toval et al., 2002b] Toval, A., Olmos, A. and Piattini, M. *Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection,* IEEE Joint International Conference on Requirements Engineering (ICRE'02 and RE'02), Essen, Germany 2002b

[Trigaux and Heymans, 2003] Trigaux, J.-C. and Heymans, P. *Modelling variability requirements in Software Product Lines: a comparative survey*, FUNDP - Equipe LIEL Institut d'Informatique. 2003.

[van der Maβen and Lichter, 2002] van der Maβen, T. and Lichter, H. *Modeling Variability by UML Use Case Diagrams,* International Workshop on Requirements Engineering for Product Lines (REPL'02), 2002

[van Gurp et al., 2001] van Gurp, J., Bosch, J. and Svahnberg, M. *On the Notion of Variability in Software Product Lines,* IEEE/IFIP Conference on Software Architecture (WICSA'01), 2001