A diversity-based approach to requirements tracing in new product development

Loris Gaio

Dipartimento di Informatica e Studi Aziendali
Università degli Studi di Trento
Via Inama, 1 — 38100 – Trento (I)
loris.gaio@economia.unitn.it

Trento, 7th June 2005

PRODUCTION models emerged in recent times have stressed the need to face complex production contexts, characterized in particular by the rise in internal and environmental variability.

In this work, a stylization of some elements concerning analysis and design of new products is given, and in particular those that involve definition and transfer phases in the development of innovative goods, where change and variability in requirements along development process are often high.

This analysis has a twofold goal: first, to supply a conceptual frame for the close examination of some dynamics of requirement's integration into an artifact's design, in order to give account of their variability along development cycle; on the other side, to propose an approach based on simple similarity metrics, to be applied to linguistic descriptions of artifacts in the early phases of development process, in order to identify components in an artifact that undergo larger variability and therefore are to be paid more attention in the subsequent phases of life cycle.

1 Introduction

The context considered in this work concerns certain modalities through which the initial phases of development of an innovative product are articulated, those marking the beginning of realization process of a new good, characterized in a systematic way by particular aspects of complexity.

A dimension taking on a particular importance in the life cycle of an innovative product is the set of activities focused on managing the requirements asked for the specific field of utilization of the final good, and the transformation of such requirements into characteristics and features of the product.

The ability to fix the required specifications in a correct way from the beginning of the development process, to integrate them together and transform them into final goods's features, is essential to assure efficiency right from the initial phases of the life cycle, primarily because any rework or adjusting activity turn out to be costly and often difficult to realize[1].

Literature concerning requirements management in industrial firms has focused on some well defined research streams: (1) modalities and conditions through which is possible to identify the notion of requirement (Akao 1990, King 1989, Pugh 1990, Urban and von Hippel 1988, von Hippel 1986, 1988), (2) the issue of integration between marketing and operations, in particular those concerning research and development activities, (Gupta and Wilemon 1986, 1990, Souder 1988), (3) the interactions occurring between the universe related to design and that concerning the customer (Clark 1985, Clark and Fujimoto 1991, Leonard-Barton 1995).

[1] For the rate of failure in innovation products development projects and their causes, see among others Burgelman et al. (1996), Ottum and Moore (1997).

The analysis of literature permits to derive three prominent conclusion: first, an essential factor is the presence of organizational structures or units capable of realizing requirements integration into characters of the final product, especially in contexts characterized by strong uncertainty and variability (Clark et al. 1987, Clark and Fujimoto 1991). In second instance, the domain of design seems to be orthogonal to problem-solving[2] one: activities devoted to design, according to literature, are mere receivers of information derived in former phases of requirements definition and articulation. In third place, it has not been defined yet a clear and persuasive enough framework able to stylize and explain the modalities through which information identified and organized in requirements by definition practices is later transformed into design details and in final product's features. The information turning on in design phase is often inconsistent to which generated by requirements gathering, despite the latter one served as starting base for design details.

In particular, this work is focused on the theme of requirements variability determined by internal and external change in the early development process, often derived from change in the organizational structure, workgroup's composition, environmental and external variables or in the problem domain, affecting the development life cycle. Such variability can be measured by diversity metrics, having in mind the goal of appraising the variation in the requirements's set along the development process, and to assess this incidence on the features of different components of the final good.

Finally, measures of diversity in time of different part of the product being developed may help as government tool for the definition of new product's features, on the basis of the persistence of the requirements along the life cycle, thus supplying an instrument to set out the level of desiderable variety in the dynamics of requirements evolution (Weitzman 1992, 1993).

Next section attempts to define the scope of the analysis and to fix some elements, independent from a particular application domain, which will be used to build an interpretation in terms of diversity for requirements management, with the final aim to develop a conceptual framework useful to give account of requirements's definition and later incorporation in product design. The following sections will be devoted to the analysis of some of these issues for the field of software production; some empirical evidence of this approach, applied to the analysis of a case study will be shown in the last section.

2 Artificial, Artifacts and Requirements

Design and production of the artificial, that is, a totally new object, has been properly framed and stylized in the work by Simon (1996): he identifies two key factors in design and realization process for artificial ojects created by man: in first instance, the goal or aim in artifact's nature; secondly, its working conditions, defined by the environment. In the set of relations needed to build and use a new product, it is necessary to divide the internal environment from the external one, so as to allow steady working conditions for the artifact, and isolate it from variations coming from the external environment. In alternative, it is possible to use a preventive adaptation approach, or in third instance through mechanisms using both solutions. In any case, the external environment contributes in defining the conditions for reaching the artifact's goals.

If the artifact owns the correct operating features, it will be able to adapt itself to the external environment; in any case, its behavior is mainly influenced from the latter. The artifact's performance depends both from the environment's variability range and from internal traits useful to identify its properties, say the features fixed in the development process and incorporated in the product.

[2]This domain is often referred as analysis in many industrial contexts.

The elements of the problem-solving process devoted to the definition of an artifact's features can be identified using the interpretation frame supplied by stakeholder theory (Freeman 1984). In such perspective, stakeholders represent the agents involved in different activities, roles and phases of an artifact's production[3]. Among those are the buyers, who purchase the final product and take on the economic cost of realization; the final users, who eventually use the product and are mainly interested to features such as its usability and reliability; people bearing the responsibility of product's maintenance and management, in the case of capital goods or services, who often are also responsible for the artifact's future evolution, and are more directly interested in the product's architecture and its innovation potential; finally, the agents involved in its definition, that is designers, who have to identify and realize the artifact's features.

A further component of this approach is stakeholders' ability to enact relational patterns to be used to raise requirements' sets in an efficient and effective way, which will be later incorporated in the artifact during its design and production. The traditional methods of planning are often inefficient in the management of the process aimed to an artifact's requirements definition: indeed, it is rather difficult to identify coordination and control variables in such process, and it is equally problematic to assess and manage its activities with a deliberate strategy.

A possible approach is suggested by the concept of *generative relationship* and has been introduced by Lane and Maxfield (1994). In detail, Lane and Maxfield identify the importance assumed by particular set of relations stated as generative, that are particular conditions where the process of requirements definition, organization and incorporation can be governed and managed; such conditions are strictly connected to the solution domain, to different complexity levels of the problem faced, to the chronological horizons typical of the requirements' definition process. Those factors influence the emergence of generative relationships, permit the definition of viable strategies for appropriate changes in the space of solutions/artifacts and eventually allow to pursue, identify and use those relations among stakeholders that are able to generate value.

A further aspect in this process of problem-solving is the ability of define the resources, capabilities and activities to be used for an artifact's definition inside the extended organizational structure in which this process has to take place (Baldwin and Clark 2000). An artifact's realization needs coordination structures, both inside and among organizational units, in order to assure that activities of *see and seek* in the problem and solution domains be not vain, but allow the company to create value. This means that resources involved in artifact's definition must be able to create and use internal and external coordination mechanisms. Such mechanisms represent driving systems which allow to concentrate efforts of human and physical resources in particular directions, with the goal of reaching appropriate design solutions.

The approach proposed by Baldwin and Clark (2000), defines two technological dimensions for instruments which have to supply support coordination in design activities: the former corresponds to technical knowledge, that allows to understand how an artifact works and what variables are used by stakeholders to evaluate it; the latter one is defined as managerial knowledge and permits to stimulate and manage agents engaged in design activities. Thus, the first should be used to "see" the design space, depicting it in a full and deep way; the second is to be used to address stakeholders' efforts and vision and permit a more efficient "seek" in the space of alternatives.

[3]It is to be noted how they are not necessarily completely identified during an artifact's development process; in many cases only the potential recipient of a final good is identified: it is the case of future customers of a large number of commodities which features are not yet completely and explicitly expressed: for a formal interpretation of this issue, see the seminal work by Lancaster (1966), subsequently investigated and stilized in depth (Lancaster 1979).

2.1 The role of language and information

A first relevant condition affects the definition of an artifact's requirements: the traits of communication channels among agents involved in definition process (Holtzblatt and Beyer 1995). The use of linguistic tools are an essential coordination modality in artifact's production activity, particularly in contexts where the artifact is a good or service with huge information content.

This does not mean that the organizational context has to choose an unique communication language, or that the adoption of a singular language be an optimal solution for such context: in many empirical situations many languages of different nature and origin can be observed: different languages must coexist because of their efficiency when used where the context is characterized by differentiated and specialized functions, capabilities, interaction structures and identities. Moreover, some research streams assert that variety and nature of languages used in an organization depend on its bond configuration, both in functional and structural terms.

In principle, language pursues two main functions:

a communication: the importance of communication in new products development is stressed, among the others, by Brooks (1975), with reference to the ability that linguistic tools must show in addressing and manage communication dynamics, in particular changes, which characterize such process[4]. Thus, language is an essential tool in this context, to allow an efficient and effective communication among agents. In this perspective, language spreads information exchange among subjects and enables the resolution of conflicts and asymmetries among units involved in the process;

b production: the information content of products and processes in present production paradigms is continuously increasing: languages used in production cycles play a prominent role in conveying, extracting and managing an amount of information and knowledge growing in size and complexity. Therefore, language becomes a primary production factor in processes characterized by strong information content, as it should have efficient and effective primitives in managing contextual information. This role has caused the emergence of a plethora of specialized languages used for the treatment of large amounts of situated information[5].

One modality through which design operates, is the definition of an internal environment that permits to replicate working conditions of the artifact, with the goals of defining and assessing its structure, operating modes and eventually its value (Simon 1996). If this modality involves a relevant number of agents, it is necessary to apply communication tools able to identify, among all viable alternatives, the parameters to be used for building functional characters and structure of the artifact and to assess its value[6].

Each stakeholder represents a separate observer, which interpretation of the events is not necessarily agreed by other agents. When different stakeholder – the final user, the analyst or different organizational units – have conflicting interpretations about goals, functions or other elements of the relation, a

[4]In this respect, it is shown how such changes can involve – and normally do – stakeholders in autonomous and independent mode, particularly in contexts where analysis and design activities are organized in work groups: *"As work proceeds, the several teams slowly change the functions, sizes and speeds of their own programs, and they explicitly or implicitly change their assumptions about the inputs available and the uses to be made of the outputs."* (Brooks 1975, p.75).

[5]This is the case of languages delivered to define technical features of industrial artifacts along a manufacturing chain, through the whole life cycle of a good, and the consequent definition of linguistic standards in CAD/CAE/CAM applications, that are used to manage information flows along the whole production process.

[6]For instance, experiments, prototypes, simulations and other testing tools can be realized to gather informations and judgments around the artifact and its working effects.

coordination break and the consequent failure in the relation are inevitable. The chance of explicating ideas and absorbing and sharing new informations about an artifact's design, are thus strictly tied to the chance of sharing a common representation among the stakeholders. The essential dimension for coordination is represented in first instance by the traits of the language, but also by the possibility to build a structure of conversation which permits an effective convergence and sharing of meaning and representations (Brooks 1975).

How the information is used in artifact's definition is also crucial. Gathering and other activities devoted to features definitions traditionally use a large amount of information in the early steps of a new product development. This approach, that traditionally stems from engineering disciplines, entails two conditions: first, it implies that the problem space be well defined *ex-ante*, and it be reducible and decomposable in subproblems. Secondly, it is assumed that the nature and scope of the problem will not change during an artifact's development, at least from the moment where its structure and features is identified. Stage-gate models, for example, postulate that entire information necessary to define the artifact's features be available from the initial phase of development process, and their nature and content will not change along development cycle.

Information complexity is thus managed by a process of decomposition and recomposition, through reduction of a complex problem in subproblems, which can be defined and managed in an easy way. Approaches normally used in this context are hierarchy Simon (1996) or the use of frames Polya (1957). The most relevant limit in complexity reduction stems from the context: the structure of problems and subproblems is built with regards to the particular viewpoint where it occurs. Furthermore, in many cases the amount of information is too wide to be reduced ad articulated in a convenient way.

A viable strategy to face the issue of information complexity in such context can be built *ex-post*, through an adaptive approach, where final working conditions of the artifact are defined and created, and non reducible knowledge and information is spread and transformed in features through the use of tools and methods which allows a feedback among the involved stakeholders, particularly final users and designers. In this respect, the management of information complexity is performed by using non linguistic communication, the use of prototypes and mockups and the adaptation of the artifact to stakeholder's requests. This approach allows for anticipation of working conditions of the artifact, and permits the transfer of tacit information or information not codified in previous phases of development process. This is a form of adaption guidance, where specifications are not kept in a definitive manner, but rather they can be changed in some extent, and the process allows, in a gradual way, for incorporation and integration of new features in the artifact[7].

Another dimension in information management in a new product development is its size. The traditional engineering models imply the aggregative character of this activity: the initial phases of the process have to identify any available information source, elicitating and organizing information to be used in artifact's definition. The treatment *ex-ante* of large information amounts has been the normal and widely followed practice in new products' development[8] (Burgelman et al. 1996). Other approaches follow an alternative strategy: the idea is to abandon the treatment of large datasets to extract information from, and to use information sources smaller in size and complexity, simpler to manage, where problems of interpretation and integration are much simpler: this approach seems to be

[7]Interactive design employs such an approach: tool such dedicated software and shared monitors permit to interdisciplinary teams (marketing, operations, R&D) to continuously interact with customers, catching their responses through audio and video channels. Customers take part to the development of new or customized products, such as tissues, furniture, entertainment services, automotive or airplane parts, insurance policies, law or accountancy services. This form of coordination is a critical factor both in reducing the process and product risks, and in raising the value of the final artifact.

[8]This orientation seems to inform many approaches in requirements management, such as *Quality Function Deployment*, scenario analysis, case-based (use case and use story) analysis and methods known as *User-centered design*.

inspired by some recent development methods from software industry.

Thus, granularity of information and modalities of its aggregation along the process of an artifact's definition represent two relevant issues in new product development. In many recent methods it is suggested to replace the traditional approach of decomposition/recomposition, *ex-ante* planning and massive information treatment, with an alternative method based information *chunks* (following the definition given by Simon (1974)) smaller in size and with reduced complexity, and their incorporation to form a robust core of specifications, gradually built as analysis process evolves in time. This approach seems to follow an adaptive and *lean* philosophy, where features emerge gradually and information is treated in smaller units, and artifact's traits emerge as result of a gradual an constant aggregation process.

3 Requirements and production of software

Traditional paradigms in software development process[9] imply a succession of initial activities addressed to definition of design specifications, starting from requirements analysis documents, produced through a process of gathering, elicitation and organization, often using a plethora of tools (interviews on the field, analysis of descriptive documents, simulations, use cases, etc.) which use large amount of codified information and tacit knowledge.

Much recent empirical evidence confirms how one of the most critical and problematic issues in software development is the correct identification and organization of requirements. Particularly, the development of correct specifications and the management of processes which involve customers in requirements definition represent the most important issues for European companies involved in software development (Lee et al. 1999).

The specific structure of software development cycle and the interdependence of its phases have moreover evidenced the motivation for shared linguistic instruments along the development process, with the perspective to minimize or to get rid of risk of information losses through the process, and the emergence of ambiguity issues[10].

3.1 Requirements management, measurement and traceability

Requirements engineering represents the result of efforts of software engineering to define the optimal conditions in which realize the activities and use the tools required to requirements management in software and information systems development (Byrd et al. 1992).

In such context, a strategy of requirements elicitation is made up of a set of guidelines to set out the information sources, extract requirements and resolve conflicts and coordination problems which possibly emerge. Elicitation phase is characterized by the presence of many communication issues and high iteration intensity among stakeholders. Consequently, techniques used in this phase do not come

[9] Software development is described and analyzed using a set of paradigms which allows to emphasize the main traits of this activity. Life cycle as tool to depict and manage the process of software development has evolved through the use of even more elaborate and articulate models: early waterfall model has been substituted by iterative, incremental, spiral and evolutionary approaches and RAD, CMM and a plethora of other development methods. However, the evolution of such process exceeds the scope of this work.

[10] In this view, one of the major efforts realized in late years is the unification of different object oriented methods towards a unique development standard (UML, *Unified Modeling Language*)(Booch et al. 1998). The idea underlying this standard is to use a singular linguistic approach for the analysis, design and implementation of software applications, in order to define and spread requirements along the process of software development, thus minimizing risks of information losses and ambiguities arising from the use of different linguistic approaches in different phases of the process.

directly from computer science mainstream, but rather from organization theory, from research on groups interaction and from cognitive science (Potts 1991).

Some of the most recent contributions to requirements management stress the contextual aspect, that is, the environment where the artifact will work. Along this lines, which seem to retrieve Simon's suggestions previously depicted about the interaction between the artifact and its environment, places the concept of *conceptual modeling* (Borgida et al. 1985) and some other more recent contributions (Jackson 1995, Goguen 1996). In first instance, it is recognized that requirements have to be identified respect to the environment where the artifact must be placed: on the other side, software features have to be identified in function of the characters of the problem and not in function of the solution: requirements concern the goal intended to be attained, and goals of an artifacts – in this respect software applications – are to be defined outside the artifact, in the problem domain and not in the solution one.

Requirements are information: any information is situated, and the context defines the meaning of requirements. Considering context means to pay attention to technical and social factors. An effective strategy for requirements management must consider and accommodate both technical aspects, influenced by context in a lesser extent, and social ones, characterized by context in marked ways. The emergence of requirements is strictly tied to the nature of interaction among stakeholders: information should be collected in the context where the final users act and where the application will operate.

Finally, in the management of software requirements a principle of incompleteness is widely accepted: there is no definition such those of complete requirement, but rather it is to be decided the acceptable level of incompleteness (Brooks 1975, Jackson 1995). Some approaches tried to cut down this gap through the application of techniques and tools able to enhance an artifact's description anticipating its behavior: along this lines places prototyping and the development of use cases[11].

The salient mainstreams of requirements' management develop in three directions: to address stakeholders' contributions towards a shared system of constraints and goals; to reach an understanding clear enough of conditions required for an artifact's realization, such as functional and structural properties and direct and side effects; finally, to record in detail such understanding in order to allow stakeholders to comprehend them and make artifact's development easier.

The importance of such goals is widely recognized: analysis of empirical data evidence how 60% of total errors introduced in software applications stem from early phases of requirements definition. Those analysis also show that as late an error following from poorly or not specified requirements is spot, greater will be the cost to fix damage caused by such error (Boehm 1981). More recent contributions permit to identify a set of elements and behaviors, stemming from operational and organizational practices and from empirical studies, that confirm this phenomenon:

1. many specification mistakes are identified long time after they have been made;
2. delay in identification increases fix costs and makes requirements redefinition more problematic, as the effect spreads through different components of the artifact;
3. the taxonomy of such mistakes includes wrong assumptions, omissions, inconsistencies, ambiguities which could have been identified in the early stages of artifact's development (Davis et al. 1993).

Furthermore, empirical data (Kelly et al. 1992) allow to validate the symmetrical hypothesis, that is, quick and accurate identification of mistakes in requirements allows to reduce the emergence of defects

[11] A company's ability to ideate and design high quality prototypes represents a key competitive factor in many industrial sectors, and not only in information technology industry. The arrangement of working prototypes in the early stages of an artifact's life cycle allows from one part to fix specification and design errors immediately, and to speed up and make more efficient the whole process of new products development. (Clark et al. 1987, Clark and Fujimoto 1991, Clark and Wheelwright 1993).

in the following stages of a new product development cycle, and dramatically lowers redesign and rework costs.

The ability to describe, identify and exhibit the evolution of a requirement along the artifact's development cycle, that is to identify and control a "path" in requirements in development process, is stated as requirements traceability. This approach is mainstream for process control in software applications development: organizations which develop applications are integrating their development process with a set of traceability methods and techniques that allow to define and control origins and nature of requirements in an artifact, in any moment of the development process: for example, relations and interdependencies among different requirements, links between different requirements and the stakeholders who generate them, the evolution of a requirement respect to a particular stage of the artifact, the level of complexity and ambiguity in a particular set of requirements.

At the same time, traceability adapts to and uses information flows which establish development process, and thus expand towards two directions, forward and backward. Moreover, requirements can represent the initial point of a management activity or otherwise its recipient.

Table 1: Directions and dimensions of traceability

	Forward	Backward
From	responsibility definitions for components implementation; assessment of change in requirements	reconstruction stakeholders' importance to requirements definition
To	changes in stakeholder's requirements and in technological constraints; redefinition of requirements' structure	testing of artifact's responsiveness to requirements; avoid gold plating

The development of different coordination and control activities which traceability is divided into, permits to organize a requirements traceability matrix (RTM) depicted in Table 1. Activities which lie on the main diagonal (*forward-from* and *backward-to*) are classified as post-traceability: they allow to link requirements to artifact's design and implementation activities, supplying with documentary evidence responsibility assignments, permitting to verify conformity of requirements to the artifact's characteristics and allowing the evaluation and analysis the impact of requirement on the features of the final product. Other two classes of traceability are classified as pre-traceability and are oriented towards documenting and managing fundamentals and elements of the technological context and the organizational and social circumstances connected to requirements definition and change.

Introduction and operational use of those coordination and control tools implies an *ex-ante* planned design: it is intuitive how this approach is not always convenient, as it implies the definition of an *ex-ante* control and coordination system in both directions between the domain of requirements and the state of the artifact and its environment, along the whole development cycle: moreover, the costs implied by this control system may not be justified by the low benefits that this system would attain.

3.2 Traceability measurement and simple similarity metrics

The approaches to traceability, view as tools to define regular and structured links in the system of relations stakeholders-artifact, and the models underlying such relations, expect and use quantitative metrics to measure and give account of evolution of an artifact in time. Such metrics are used in a relevant number of operating contexts, in order to describe the size growth of the artifact, the relational structure which generated it, its internal complexity. Simpler metrics are used to evaluate size and

permit to identify dimension and complexity reached by the set of requirements which describes the artifact:

- number of processes or their components, such as flows and actors;
- overall number of final requirements;
- number of mistakes found in different phases of the development cycle;
- number of *to-be-determined* (TBD) requirements;
- number of requests of change in requirements (volatility).

These metrics can explain only growth dynamics in the system, but non the effects caused by shifts in design or operating variables, which can influence requirements state: for example, change in the number and composition of information sources or stakeholders, the change in the number of interfaces among them, the effects deriving by changes in languages and domains. To identify the effect of such factors it is necessary to use metrics able to catch the internal variability of the artifact's representation, and to explain its causes.

A perspective for the analysis and identification of variability factors experienced in time along variations in information sets, stems from the study of evolution of similarity: when information is in textual form, those techniques are provided by computational linguistic. Similarity is a property, measurable in quantitative way, which allows to define the degree of relation between two or more objects with an information content. There is a plethora of similarity measures defined to quantify this character[12]; simpler metrics are used for the evaluation of binary information, where attributes for the objects can in alternative assume values [0|1]. In this case, for any pair of objects a, b each attribute can assume alternative states $00, 10, 01, 11$.

The simplest comparison of characters defined on a binary domain can be computed starting from cardinalities of such states, as depicted in Table 2.

Table 2: Cardinality of states on a binary domain

M_{01}	= number of attributes where $a = 0$ and $b = 1$
M_{10}	= number of attributes where $a = 1$ and $b = 0$
M_{00}	= number of attributes where $a = 0$ and $b = 0$
M_{11}	= number of attributes where $a = 1$ and $b = 1$

Starting from these simple information, more synthetic index can be computed, used to compare different stages which a particular system is situated. A first elementary and widely used index is *simple matching coefficient*, defined as

$$\mathcal{S}^{\text{SMC}} = (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) \tag{1}$$

which permits to compare characteristics both present and absent in objects, and all possible states.

A more complex measure used to evaluate the degree of overlapping for the features of two objects is Jaccard's index, which is the ratio between the values in common for two objects (intersection) by the sum of those in common (intersection) and those exclusive to one of them (union), that is

$$\text{Jaccard} = \text{number of } 11 \ / \ \text{number of values different from } 00$$

[12] A wide number of metrics is provided by specialized literature in this respect: for a deep and detailed exposition, see (Lee 1999).

or, more formally

$$\mathcal{S}^J = (M_{11})/(M_{01} + M_{10} + M_{11}) \tag{2}$$

This ratio represents a coefficient for binary variables where void unions are excluded both from the denominator and from numerator, and intersections an unions have equal weight.

An extension of this index is extended Jaccard's coefficient, the version of binary index for cardinal or continuous characteristics: here, two vectors of features d_1, d_2 are used to compute their inner product, and it is divided by the sum of cardinalities of two vectors and raised to square, and then added to the norm:

$$\mathcal{S}^J(d_1, d_2) = \frac{d_1 \cdot d_2}{||d_1||^2 + ||d_2||^2 - d_1 \cdot d_2}. \tag{3}$$

Extended Jaccard's index was originally developed in biology to evaluate similarities in distributions of floral species in different geographical areas (Jaccard 1912).

Another measure widely used in similarity is Cosine index, used for cardinal or continuous variables. If d_1 e d_2 are vectors of ordinal values (linguistic terms thus), this index corresponds to the cosine of the angle projected between two vectors just mentioned: if these are perfectly superposed, then the angle is equal to 0 and cosine is 1; in the opposite case, if two vectors are orthogonal, the cosine of the angle is 1. In case of text documents, the coefficient is obtained computing the cardinality of each term in the vectors, then calculating the ratio between the inner product of two vectors and the product of the norms of both vectors, that is

$$\mathcal{S}^C(d_1, d_2) = \frac{d_1 \cdot d_2}{||d_1||\ ||d_2||}. \tag{4}$$

This index is widely used for computing the similarity in texts, as it allows to normalize the characteristics through a covariance matrix. It also is scale invariant and does not depend from data length, that is $\mathcal{S}^C(\alpha d_1, d_2) = \mathcal{S}^C(d_1, d_2)$ for $\alpha > 0$; this property allows the treatment of documents with discordance in composition or size of texts.

There is a salient difference between cosine index and Jaccard coefficient: both permit to compare two texts, but the latter takes into account non overlapped information in the denominator. For this reason, cosine index is much more sensible towards size evolution of two documents.

Both measures are widely used for computing comparison coefficients on broad masses of textual documentation, and applied in a diffuse manner in information retrieval applications and tools (Salton and McGill 1983, Dhillon and Modha 2001).

4 An empirical evidence: The development of SS

In next section is exposed an empirical analysis carried out on the development process of a software application implemented between the second half of nineties and the end of 2003: the application's scope is the management of whole range of services supplied to the students by an university.

The goal of the analysis is to describe the modalities of the application development, completed with a detailed investigation about some aspects of volatility and change in the production process, and how these elements can be connected with some measures of similarity computed on description documents produced along the development process.

The analyzed project is a component in a larger programme composed of a series of initiatives carried on from late 1996 in the University of Trento (UniTN) and afterwards involving other organizations. The programme was characterized by two principal goals:

- the temporary substitution of the current application, by that time at the end of the life cycle, characterized by an obsolete hardware and maintained by retiring staff;
- the definition of requirements for a new application to be used for management of all activities addressed to university students, and the subsequent implementation of a software application.

Clearly, the main need of second goal was to build *ex-novo* a corpus of consistent structured information, as complete as possible, to identify functionalities and features for an university's secretarial services, to be used in a following phase for building a software application. One particular driver of such an initiative was the reform of the university regulation, at that time only in the initial phases, which constituted both a source of opportunity and a relevant risk for the system being developed.

In chronological perspective, the whole programme was split into four specific initiatives, articulated in the layout depicted in Table 3.

Table 3: The Programme "Segreteria Studenti"

Acronym	Goals	Contents
SS0	short duration project for maintenance of existing application in order to migrate data and applications to a new software and hardware architecture	partial and limited reengineering of data structures and procedures, minimizing information losses in the passage to new system
SS1	rewriting of a new application starting from existing requirements, reviewed by results of project SS0	design and implementation strictly integrated, very small team of three persons (two *key user* and one developer)
SS2	*ex-novo* definition of functional, structural and technological requirements for secretarial services to an university's students, and to be used in a following project; redefinition of typical and partially known processes and identification of elements (flows, actors, data)	challenging goals, as the project aimed to anticipate many traits of university reform then introduced 2 years later (Legge 509/99); definition of an initial team with small size (3/6 persons)
Esse3	use of specifications of previous project and realization of a software application to be used in an university organization; possible integration of specifications with other sources	transfer of specifications produced in SS2 to an external software house for implementation of the application

The projects where our analysis focalizes are SS0 and Esse3, evidenced in gray. Such initiatives showed the characteristics of a new product development project. Moreover, it was characterized by a high environmental uncertainty: the reform law (L. 509/1999) being defined not completed and approved yet, would have introduced in a short time an operating and structural didactic model completely different from the existing one, which at time only some marginal elements could be identified. Thus, the issue of requirements identification and organization of the new system was characterized by uncertainty generated by at least two factors: first, the need to define *ex-novo* a set of requirement from scratch, without a previous experience in the field; in second instance, a continuously changing environment, as the reference model for didactics was uncertain and constantly moving: only some elements of the future regulation (Legge 509/99) were known: a bad shift in the first phase of requirements management process would have meant big efforts and high costs in redefinition and rework activities.

The operations of SS2 begin with the raising of a small team at the University of Trento, which size

always remained small during the project, with a maximum dimension of six people[13], composed by internal and external persons. Staff turnover will increase in time but will however remain very low along this project. Initial activities of gathering and elicitation take up the time for twelve months, and has been carried on almost exclusively through textual data gathering, through use of proper templates, checklists and description of actions and activities; these were distributed to final and key users; no other advanced tool of analysis and design was used at this stage.

Aside requirements gathering, elicitation and organization of requirements were carried on: the team arranged and integrated specifications by (1) preliminary control of properties of requirements (consistency, robustness, redundancy) by using control lists, simulations, model checking tools; (2) extended usage of use cases and simulations for requirements validation.

In September 1998 this phase was closed. Requirements were organized by typical processes, in a hierarchical structure composed by processes, subprocesses and extents. The meta-structure of the application will remain roughly identifiable along the whole development cycle. Any requirement was described in a deep and extended manner, but without recourse to formalized and structured methods, rather using natural language to describe characters and distinctive features.

The second phase of SS2 began in fall 1998, and was aimed to formally organize and transfer requirements to CINECA[14] for the following phases of application development. An important decision was to adopt UML (Booch et al. 1998) to formalize and document requirements, then represented in an informal way. The transfer of SS2 requirements from initial context to an external organization takes place gradually and lasts almost two years. The development process in CINECA is at first managed by a larger group of variable size up to ten people; it will grow in time, as it has to translate requirements to internal development methodologies used in that organization. Particularly, development methods used in CINECA make use of proprietary tools of advanced modeling and implementation, distinct for data and procedures.

The growth in size and complexity of the project lead to a spillover of CINECA and the foundation of an external software house, named Kion for the subsequent development of the application. The new corporation began operations in Fall 2000 with some people coming from CINECA and hiring of new resources, and the mission to design and implement a commercial application named Esse3 . Kion inherited the whole information developed at the time, and integrated it with new requirements defined from other information sources, namely other Italians Universities. The main phases and details of the SS2/Esse 3 projects are summarized in Table 4.

A further key element marks this phase: the entry of other stakeholder respect to the initial ones, that is some Italian Universities, and the consequent need to manage (gather, extract, formalize, integrate) other requirements and incorporate them in the initial set, which originated from UniTN[15]. In this phase there is the intersection of three specific groups of activities: (1) formalization of requirements between UniTN and CINECA, (2) systematization of requirements towards internal methodologies used in CINECA and subsequent transfer to Kion, (3) integration an arrangement with third part requirements.

No formal approach to requirements tracing has been realized during the whole development process: it is thus not possible to reconstruct structure and dynamics of requirements process along the initial

[13]The team was composed by a project manager, a technical analyst, a "formal" analyst and up to three key-users, who would shift in time.

[14]It is a consortium among some Italian Universities and other public organizations for the purpose of realizing and managing some common information and computation services. CINECA is located near Bologna.

[15]The collaboration with other stakeholders could be continuous and frequent in time and devoted to deepen particular issues: it is the case of the collaboration with the University of Verona for defining structure and characters of student's fee module.

Table 4: From SS2 to Esse3

Start	Contents	Details
Fall 1998	Transfer of Project SS2	Combined analysis CINECA/UniTN. Requirements transfer and periodical meetings. Conceptual Analysis of requirements (ER, DFD, support documentation). Choice of technology domain and application architecture.
June 2000	First Prototype	Release of first prototype (SS2); testing by CINECA and some Italian Universities to assess implementation of newly released reform (L. 509/99) in the application.
September 2000	Spillover of Kion	First core of Kion; initial design and first product development. Software architecture and design of first modules.
January 2001	First beta	Release of version 0.0.1 of ESSE3 (incomplete, with some initial modules).
Spring 2001	Conjoint Team	Formation of conjoint team among Universities of Trento, Trieste, Modena and Salerno to validate design decisions and address following development.
Whole 2001	Implementation	Application development; release of subsequent modules to various area (administrative, student's career).
June 2002	Operating Version	First release at the University of Urbino.

phases of the development. Nevertheless, it is possible to perform some kind of post-traceability, analyzing post-mortem documentation available for the development process, by using some tools of computational linguistic seen before in paragraph 3.2. The goal of this analysis is to depict the most critical components in the artifact's structure and investigate some hypothesis of their variability, in order to evidence and assess the incidence of environmental factors and linguistic variables in system description and in requirements formation.

4.1 Empirical evidence in requirements' evolution

The analysis has been performed on three document set of application requirements, related to three typical chronological stages of the development cycle, as depicted in Figure 1. The content of such documents describe the system in terms of requirements in three instants of the development, respectively:

September 1998 when internal team in UniTN concluded their work and presented requirements set collected and organized, but yet not formalized; di UniTN, con la presentazione del set di requisiti raccolto ed organizzato, ma non ancora formalizzato e trasferito a CINECA;

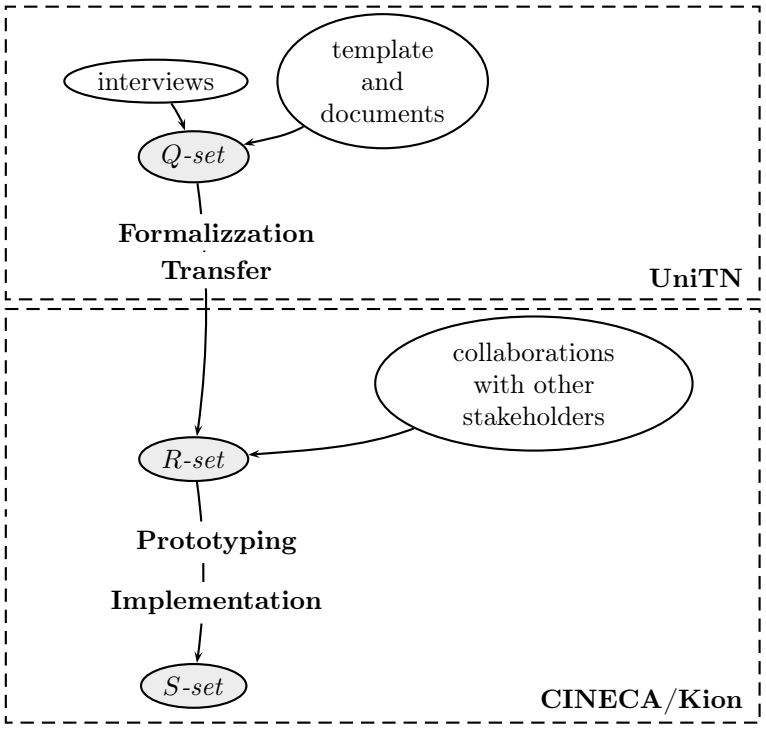
April 2002 in partial version, with requirements' transfer completed and partially integrated by requirements fom third parties;

December 2003 in a version partially operating of the system introduced in the University of Trento.

Tree document sets, referred respectively with Q, R e S, are mainly composed by word processor files containing formatted text, tables, pictures and diagrams. Frequency and similarity analysis has been performed on whole text present in documents, without regard to context where text was collocated.

A first comparison among document sets allowed to identify the main structure of the application, and subsequently some typical processes which span the whole development process. The main areas of the application are classified as:

Figure 1: Data set used in requirements analysis



1. structure and articulation of didactical supply from an University's point of view;
2. interaction of student with didactical domain (curricula, exams, graduation);
3. administrative (orientation, enrollment, acceptance, matriculation).

The comparison allowed to clearly identify nine typical processes which compose the application, homogeneous and consistent enough by structure and composition, on which focalize the following detail analysis. Some other processes have been discarded because their structure (subprocesses, components) could not be recognized in different document sets: other process have been completely redefined or abandoned during the development cycle.

The processes identified, marked with the acronym used in the set Q, have been classified by their nature and collocated in one of the areas just defined:

University-centered: where the University is the main actor in term of supplier of formative and didactical services (P6, P7, P8, P9);

Student-centered: processes where the student is the key actor, and represent the main focus (P10, P11, P12);

Administrative: related to clerical, bureaucratic or repetitive activities (P1, P2);

Table 5 shows a first evidence of the quantitative analysis performed of document sets. Ad can be noted, size dynamics is growing in time for all processes, even though in differentiated way in time and by process considered.

A relevant variability in the quantitative evolution of requirements documents concerning different processes can be seen: documents of class "University-centered" grow up to five times respect to the

Table 5: Size evolution of requirements documents

	Q set			R set			S set		
ID	Sentences	Words	Characters	Sentences	Words	Character	Sentences	Words	Characters
P1	960	10925	69492	2016	24635	177366	4608	50045	350808
P2	1200	13915	76584	1308	14710	97140	1676	17640	134556
P6	1620	14930	82284	6972	73890	421782	7680	82150	517140
P7	1324	9915	59760	4384	45525	298746	16952	141295	777720
P8	1392	12525	75294	5800	38710	192192	11976	98895	516288
P9	1812	24780	157398	5632	74160	480078	12292	184145	1195938
P10	860	9110	58614	1704	13350	81420	4160	39510	239460
P11	1928	19990	127074	2576	22840	146364	5024	56495	373110
P12	1356	19270	115092	1440	22510	150384	1824	29510	205392

previous phase, while size of other processes remains substantially stable, with a maximum growth rate of 2.5 times respect to the previous stage (P1, P10). Some indications of dimensional growth differentiated by application components already emerge here.

The analysis of similarity between groups of processes has been performed by using metrics computed by cosine index and Extended Jaccard coefficient. Such elaboration was carried out on texts previously filtered using stop-word lists, in order to exclude common linguistic terms such as propositions, conjunctions, articles. The text has been then processed with a stemming algorithm, in order to collect common terms and give account for root elements in text, eliminating dispersions caused by gender variables or by conjugation of verbs.

The first metric is cosine index, computed for each process and pairs of document sets related to different phases of development cycle and shown in Table 6.

Table 6: Similarity — Cosine Index

ID	$Q \longrightarrow R$	$Q \longrightarrow S$	$R \longrightarrow S$
P1	0.63479	0.60158	0.79560
P2	0.59376	0.57626	0.62135
P6	0.23571	0.20900	0.22451
P7	0.28202	0.12365	0.28998
P8	0.27453	0.20738	0.25443
P9	0.24708	0.18254	0.23316
P10	0.47822	0.41907	0.54363
P11	0.48970	0.35725	0.51437
P12	0.63220	0.43483	0.66720

As can be noted, some processes evidence strong similarities along the whole application development (P1,P2,P12), other components are placed in an intermediate range (P10,P11) while some other show very low and varying similarities along the whole development cycle.

The values of extended Jaccard coefficient, shown in Table 7, allow to confirm this situation. In this case, similarity levels are generally lower than cosine index, as Jaccard's coefficient considers disjoint elements in two sets: in this case, elements present in only one of two sets – thus a growing set of requirements – affects negatively the coefficient, lowering its value.

The analysis of similarities allows to identify two classes of processes with different characters:

- a first set with processes P1, P2, P7, P10, P11 e P12, with $\mathcal{S}(Q,S) < \mathcal{S}(Q,R) < \mathcal{S}(R,S)$ (where $\mathcal{S}(\cdot)$ is an overall similarity metric (cosine, Jaccard));
- a second set contains processes P6, P8 e P9 with last relation reversed, $\mathcal{S}(Q,R) > \mathcal{S}(R,S)$.

Table 7: Similarity — Extended Jaccard coefficient

ID	$Q \longrightarrow R$	$Q \longrightarrow S$	$R \longrightarrow S$
P1	0.39551	0.34641	0.40910
P2	0.36330	0.30256	0.36516
P6	0.11128	0.09960	0.10807
P7	0.12287	0.10832	0.12523
P8	0.11072	0.09801	0.10551
P9	0.14069	0.12954	0.13453
P10	0.21490	0.18331	0.23083
P11	0.22319	0.18323	0.23812
P12	0.31428	0.27258	0.32973

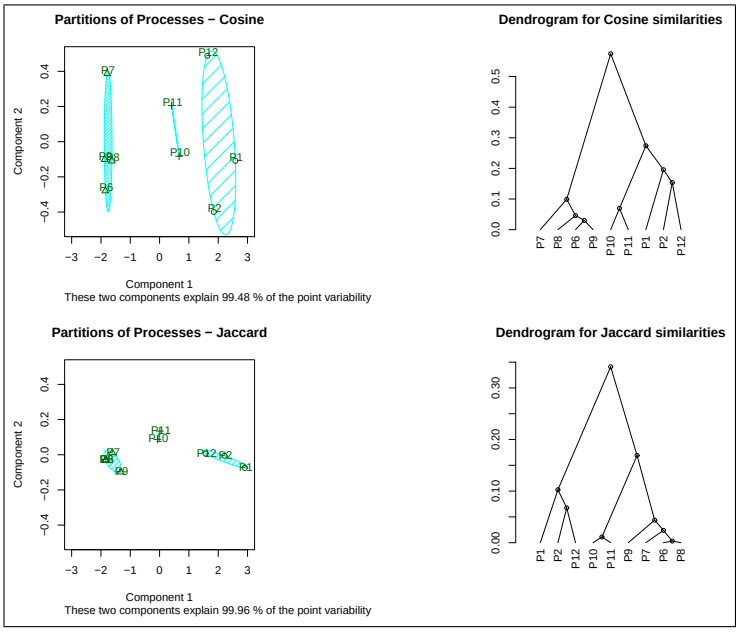
Pair comparisons of similarities should allow to test and control some hypothesis in the dynamics of requirements management, and particularly:

consistency: evolution of similarity metrics should reflect time proximity between to requirements' sets: closer sets should be in principle more similar than farther ones; that is, $\{Q, R|R, S\} < Q, S$; this hypotesis is verified for all components of the application;

reinforcement: management of requirements should show similarities growing in time, through the definition of information sets more and more homogeneous and compact and less volatile in time formally we should have $\mathcal{S}(Q, R) < \mathcal{S}(R, S)$; this not be the case, we would face a phenomenon with a certain degree of volatility with variations in requirements' composition or in the use of linguistic tools. This seems the case of processes P6, P8 and P9, where both similarity metrics get smaller in time.

Furthermore, cluster analysis with hierarchical and partition methods has beel performed on similarities to group processes with behavior: evidence of this analysis is shown in Figure 2.

Figure 2: Clusters of Processes



As can be noted, the formation of clusters in both hierarchical and partition analysis confirms the intuition of three groups of processes, differentiated for persistence of features in time.

Besides these conclusions on incidence of diversities in time, there is some evidence about some internal and environmental factors which could explain differences in similarity metrics. Processes which show higher constant similarities concern simpler and less complex application extents, characterized by reduced uncertainty and low variability. They concern bureaucratic and administrative processes (P1, P2) and the regulation of activities with certain content and defined behavior (P12). Those processes seem to keep information in a more persistent manner and are less interested by integration of new information in time.

Conversely, the development of processes with lower similarity values in time, depends strongly by contextual factors with uncertain nature, in particular from reform law (509/99), which represents the greatest factor of uncertainty. The incidence of uncertainty ascribed to reform law seems to be greater for processes where the key actor is the university, rather than for processes in which this role is played by the student.

Finally, there are internal factors concerning the development process, which contribute do explain difference in similarity data: these are summarize in Table 8.

Table 8: Comparison elements for some phases of development process

Activity	Period	Origin	Actors	Style	Language
Gathering and first formalization	1997-1998	UniTN	~ 5	narrative	natural
Formalization on model	1999-2000	UniTN, CINECA	$5 - 10$	formal	UML
External requirements' gathering	2001-2002	Kion, Universities	~ 25	formal/narrative	proprietary/natural
Prototype development	2001-2002	Kion	$30 - 50$	formal	proprietary

As can be noted, size in human resources and staff for development process continuously increase in time, up to a maximum during the prototype realization and implementation of application. The variety in actor composition, not shown in the table, increases in time: from one homogeneous team of people operating in the same organization, teams become increasingly different by origin and competence: their organizational context is different (Universities, CINECA, Kion) and their expertise and competence scope is in many cases very different. A further variability is induced by linguistic formalisms and tools used along development process: in general, they change from non-formal modes of description and the use of natural language toward the use of standard or proprietary languages and formal styles of information representation. These elements seem to explain only partially the evidences of similarity measures computed during the analysis of texts: indeed, the differences among different processes tend to remain more persistent through the development process, while differences between single stages in a particular process are never particularly relevant. Furthermore, in six cases out of a total of nine similarities for requirements' sets managed inside a single organization (CINECA/Kion) are be greater than similarities between UniTN and CINECA. Anyway, the environmental variable affected by strong uncertainty and frequent variability in problem domain (reform law) seems to have a greater influence on requirements stability, and to constitute the most important explanation factor in process' requirements volatility.

5 Concluding remarks

In this work a framework using the paradigm of diversity is proposed, to give account how an artifact's development process can be analyzed and explained using the linguistic representation of its structure

and expected behavior in different stages of its development cycle. The analysis of requirements description and evolution using simple similarity metrics can explain the different influence of internal and external factors over components of the artifact. In this lines, we tried to investigate how the processing of data that concern the description of a new product becomes a tool for coordinating *ex-post* product variability and requirements management, thus becoming a feasible instrument of post-traceability and allowing to reconstruct requirements formation dynamics in a systematic way.

This approach allows to define which components in a product being developed are the most critical ones, using similarity metrics to assess stability in time of linguistic information sets based on requirements and used to describe different stages of development of an innovative product. Empirical evidence resulted from the analysis on some *post-mortem* datasets regarding a software application is shown.

While the model presented in this paper may appear to be overly simplistic, it can be easily extended in several directions. For example, testing the approach using rework data to verify hypothesis of major efforts over more problematic components of the application is a possible way to assess and complete the framework proposed here.

References

Akao, Y. (1990), *Quality Function Deployment QFD: Integrating Customer Requirements into Product Design*, Productivity Press, Cambridge MA.

Baldwin, C. and Clark, K. (2000), *Design Rules. Volume I: The Power of Modularity*, MIT Press, Cambridge Mass.

Boehm, B. (1981), *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs , NJ , USA.

Booch, G., Rumbaugh, J. and Jacobson, I. (1998), *The Unified Modeling Language User Guide*, Addison-Wesley.

Borgida, A., Greenspan, S. and Mylopoulos, J. (1985), 'Knowledge representation as a basis for requirements specification', *IEEE Computer* **18**(5), 82–101.

Brooks, F. J. (1975), *The Mythical Man Month: Essays on Software Engineering*, Addison-Wesley Publishing Company.

Burgelman, R., Maidique, M. and Wheelwright, S. (1996), *Strategic Management of Technology and Innovation*, 2nd edn, Irwin, Chicago.

Byrd, T., Cossick, K. and Zmud, R. (1992), 'A synthesis of research on requirements analysis and knowledge acquisition techniques', *MIS Quarterly* **16**(1), 117–138.

Clark, K. (1985), 'The interaction of design hierarchies and market concepts in technological evolution', *Research Policy* **14**, 235–251.

Clark, K., Chew, B. and Fujimoto, T. (1987), 'Product development in the world auto industry', *Brookings Papers on Economic Activity* **3**, 729–771.

Clark, K. and Fujimoto, T. (1991), *Product Development Performance: Strategy, Organizational and Management in the World Auto Industry*, Harvard Business School Press, Boston.

Clark, K. and Wheelwright, S. (1993), *Managing New Product and Process Development*, Free Press, New York.

Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, P., Ta, A. and Theofanos, M. (1993), Identifying and measuring quality on software requirements specification, *in* 'Proc. Software Metrics Symp.', IEEE CS Press, pp. 141–152.

Dhillon, I. and Modha, D. (2001), 'Concept decompositions for large sparse text data using clustering', *Machine Learning* **42**(1), 143–175.

Freeman, R. (1984), *Management: A Stakeholder Approach*, Pitman, Boston.

Goguen, J. (1996), Formality and informality in requirement engineering, *in* 'Proceedings: 2nd International Conference on Requirements Engineering', IEEE Computer Society Press, pp. 102–108.

Gupta, A. and Wilemon, D. (1986), 'A model for studying R&D-marketing interface in the product innovation process', *Journal of Marketing* **50**, 7–17.

Gupta, A. and Wilemon, D. (1990), 'Improving R&D/marketing relations: R&D's perspective', *R&D Management* **20**(4), 277–290.

Holtzblatt, K. and Beyer, H. (1995), 'Requirements gathering: The human factor: Introduction', *Communications of the ACM* **38**(5), 30–32.

Jaccard, P. (1912), 'The distribution of flora in the alpine zone', *The New Phytologist* **11**(2), 37–50.

Jackson, M. (1995), *Software Requirements and Specification: A lexicon of practice, principles and prejudices*, Addison-Wesley, Wokingham, England.

Kelly, J., Sherif, J. and Hops, J. (1992), 'An analysis fo defect densities found during software inspections', *The Journal of Systems and Software* **17**(3), 111–117.

King, B. (1989), *Better Designs in Half the Time: Implementing Quality Function Deployment in America*, GOAL/QPC, Methuen MA.

Lancaster, K. (1966), 'A new approach to consumer theory', *Journal of Political Economy* **74**(1), 132–57.

Lancaster, K. (1979), *Variety, Equity, and Efficiency*, Columbia University Press, New York.

Lane, D. and Maxfield, R. (1994), 'Strategy under complexity: fostering generative relationship', *Long Range Planning* **29**(2), 215–231.

Lee, L. (1999), Measures of distributional similarity, *in* 'Proceedings of the 37th conference on Association for Computational Linguistics', Association for Computational Linguistics, pp. 25–32.

Lee, M., Dutta, S. and Van Wassenhove, L. (1999), An empirical analysis of software production problems in european software units, Technical Report 99/28/TM/RISE, INSEAD, Fontainbleau, France.

Leonard-Barton, D. (1995), *The Wellsprings of Knowledge*, Harvard Business School Press, Cambridge, MA.

Ottum, B. and Moore, W. (1997), 'The role of market information in new product success/failure', *Journal of Product Innovation Management* **14**(4), 258–258.

Polya, G. (1957), *How to Solve It: A New Aspect of Mathematical Method*, 2nd edn, Princeton University Press, Princeton, NJ.

Potts, C. (1991), Seven (plus or minus two) challenges for requirements research, *in* J. Finance, ed., 'Proceedings of the 6th International Workshop on Software Specification and Design', IEEE Computer Society Press, Como, Italy, pp. 256–259.

Pugh, S. (1990), *Total Design: Integrated methods for successful product engineers*, Addison Wesley, Reading MA.

Salton, G. and McGill, M. (1983), *Introduction to Modern Information Retrieval*, McGraw-Hill, New York.

Simon, H. (1974), 'How big is a chunk?', *Science* **183**, 482–488.

Simon, H. (1996), *The Sciences of the Artificial*, 3rd edn, MIT Press, Cambridge, Mass.

Souder, W. (1988), 'Managing relations between R&D and marketing in new product development projects', *Journal of Product Innovation Management* **5**(1), 6–19.

Urban, G. and von Hippel, E. (1988), 'Lead user analyses for the development of new industrial products', *Management Science* **34**(5), 569–582.

von Hippel, E. (1986), 'A source of novel product concepts', *Management Science* **32**(7), 791–805.

von Hippel, E. (1988), *The Sources of Innovation*, Oxford University Press, New York.

Weitzman, M. (1992), 'On diversity', *The Quarterly Journal of Economics* **107**(2), 363–405.

Weitzman, M. (1993), 'What to preserve? an application of diversity theory to crane conservation?', *The Quarterly Journal of Economics* **108**(1), 157–83.