Spring 4-12-2016

J'Accuse! ATTRIBUTION OF BLAME WHEN SOFTWARE IS AN ACTOR (11)

Joy Garfield
Worcester University, j.garfield@worc.ac.uk

Daniel Dresner
University of Manchester, daniel.dresner@btinternet.com

J'ACCUSE!
ATTRIBUTION OF BLAME WHEN SOFTWARE IS AN ACTOR

J. Garfield and D. Dresner (Worcester University)

Abstract

The desire for closure after an accident may be hastened by the attribution of blame. This is particularly attractive in situations where complex factors may distance the understanding of attribution from those who may not be familiar with all vectors towards the failure causing the accident. The keyword here is 'accident' suggesting that deliberate action/s have not been the cause. It is pertinent to establish systems – such as those responsible for process control where it may be argued that the risk of remote, malicious intervention was not readily foreseeable at the time of their realization. The paper puts forward a framework for the elaboration of requirements with a focus on organizational factors as a way of teasing out problems in early development. The objective is to achieve a sense of assurance that due diligence is both done and seen to be done in an increasingly non-deterministic operational environment.

Keywords: accidents, attribution of blame failures, trustworthy software, requirements engineering, business IS/IT alignment.

1.0 Context: systemic consideration v. witch hunting

Checkland's *soft systems methodology* - SSM (Checkland, 1985) decomposes the analysis of the most complex system into what may be the simplest practical consideration of its components. These components are those which allow an understanding of the communication and control (Weiner, 1961) that goes on inside the system (transformation) and how that affects the world outside that system (environment). This would apply to the minor inconvenience caused by some deficiency in functionality through to a significant failure resulting in death of one or more victims of the system who should have been the system's beneficiary.

Staying with Checkland's nomenclature, information technology is designed to effect some form of transformation either of data for the sake of information (data processing) or for the sake of controlling any number of physical processes (for example SCADA[1] or fly-by-wire technology). A system's boundaries are defined by a combination of the environment (or environments) housing the system and its components. The potential for system complexity as a product of interacting subsystems provided by a combination of integrated and

[1] Supervisory control and data acquisition of industrial control systems.

segregated supply chains can be seen in Figure 1 which was created to explain a superseded version of system life cycle standard ISO/IEC 15288 (2015)[2].

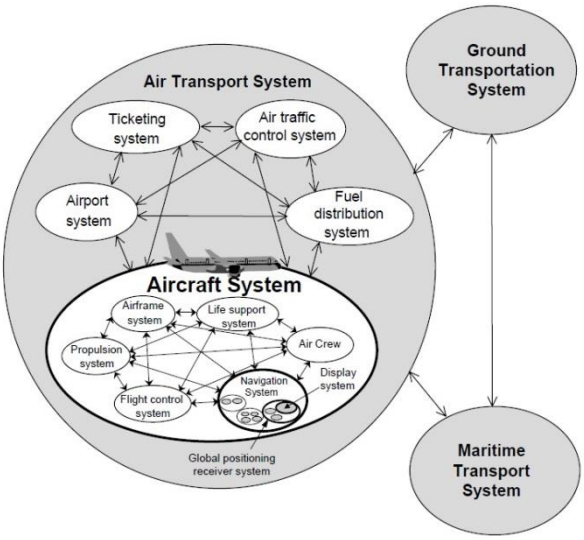

Figure 1: The system and the subsystems – the challenge of defining the boundaries ISO/IEC 15288 (2015)

The relevance of a system and its purpose are viewed by the interpretation of three classes of human components (Figure 2): the customer (or client) beneficiary (or victim), the actor – operator or creator, and the owner (who has the controlling stake as to whether the system has the right to continue to be). These roles suggest some natural divestment of responsibilities (ISO/IEC 38500, 2015).

Software continues to be until it is decommissioned (ISO/IEC 15288, 2015). The through-life cycle governance challenge is to define the responsibilities of individuals and groups in an organisation (or the inevitable supply chain) and to get them to understand and accept that these responsibilities must be met (ISO 31000, 2009 and ISO/IEC 38500, 2015). This ought to happen without the need to seek out accountabilities because this often signals deep failure and a desire to allocate a scapegoat (Barzun, 2014) in the event of a problem rather than a positive attitude to on-going problem solving. Governance demands that information technology policies, practices and decisions show respect for human behaviour (ISO/IEC 38500, 2015) so the attribution of blame for an incident where software control is part of the basic design concept should be tempered with consideration of the intention of the operators,

[2] The system lifecycle view is the foundation for trustworthy software according to PAS 754 (2014).

and the compliance of the developers and their governance and management structures (ISO 9001, 2008). So, we may posit that without deliberate, probably malicious, intention there is no fair attribution of blame to the focal actor in the software life cycle at the time of an accident.

The software development cycle is well documented within the Software Engineering Body of Knowledge (SwEBoK) (Bourque and Fariley, 2014) and standardized (ISO/IEC 12207, ISO:2008). It is custom practice to release significant software into the production environment with known defects and an expectation that further defects – often manifesting as security vulnerabilities – will emerge when the software is in use. Here, we define defect as a condition in the software where it fails to operate in a manner explicitly – or implicitly – defined by the software's requirements.

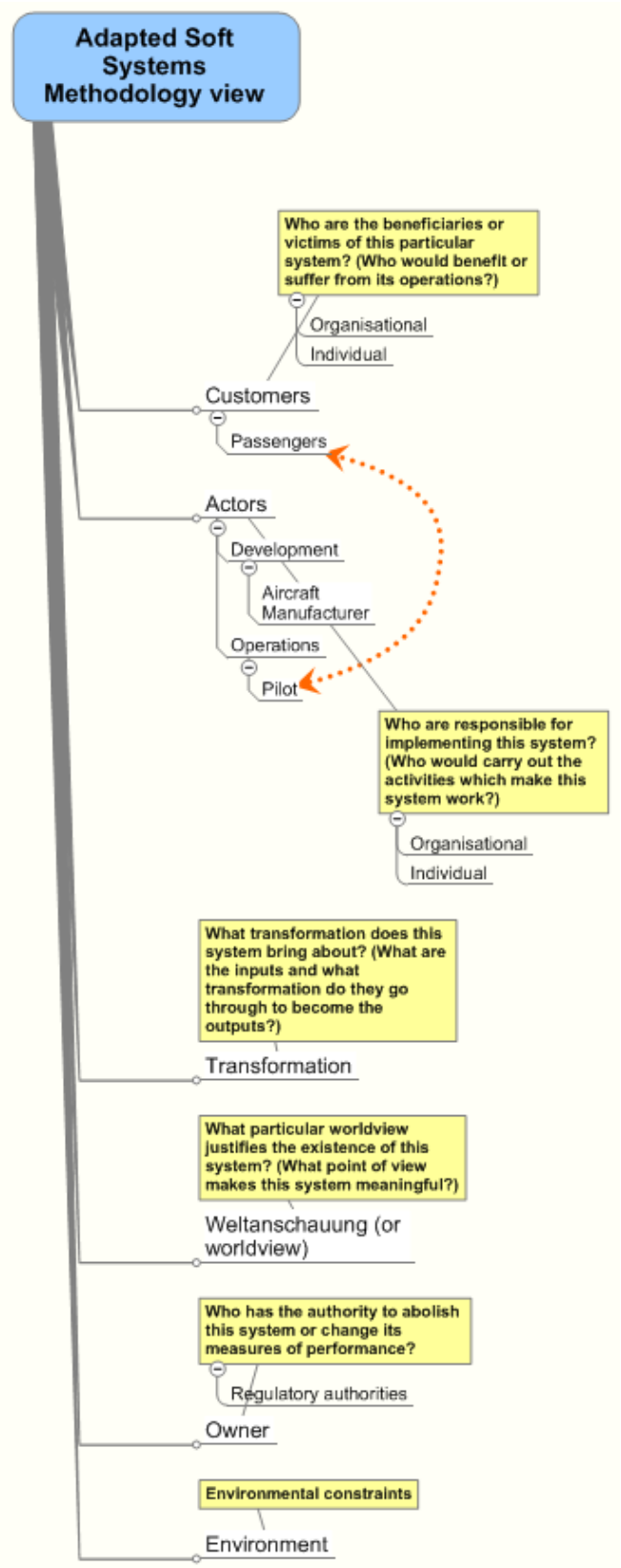

Figure 2 Roles in the software life cycle from a soft systems perspective

1.1 Failure of systems

Systemic failure – such as the destruction of an entity controlled by software (for example, the ill-fated flight AF447 *Bureau d'Enquêtes, 2012*) or the failure of an entity to successfully complete its mission (such as a patriot missile battery, GAO, 1992) or from some aspects of both – may at some time during post-morta enquiry be attributed to the cause popularly labelled 'human error'. Speculation as to the cause of loss of the 1986 Chinook helicopter in the Shetland Isles and flight AF447 over the Atlantic in 2009 were attributed (at least for some time) to this taxon of failure. Human error is not a helpful epithet. In its context as an adjective, the term human implies something socially acceptable to a community of people. We therefore propose that the label of human error is therefore an oxymoron unless it refers to a doing something inhuman by mistake. The operator user and the developer maintainer are all parts of the system.

Computing cases (.org) (Miller et al., 2016) categorises failures where people are to blame as those caused directly or where failure to act had disastrous consequences. Rather more helpfully, systemic definitions propose of a man-machine waring with man-task misfits with the cause typically being attributable to a design error. This supports a less biased inference of blame through suitable analysis of the life cycle. One might propose ISO/IEC 15288 as a systems model to label the stage - or stages - of the life cycle at which the defect was introduced and a life cycle based sequence of events which led to the risk from the defect being realised.

Channel 4's air crash investigation series (S 12 E 13) suggests that training has not made the transition to flying automated craft. Flight AF 447, from a quality airline using an Airbus 330, was scheduled to make the 11 hour flight from Rio de Janeiro to Paris. The expectations were that it would take off manually continue on autopilot, and then hand back to manual control for the final two minutes of landing. Air accident investigator Dr Matthew Greaves observes the healthy track record of the flying crew with 11,000, 6500, and 3000 hours of flying attributed to the captain, co-pilot, and first officer respectively.

The crash of AF 447 (Ministère del'Écologie, 2012) can be attributed to the formation of ice crystals in equipment designed to measure speed. This was the start of a number of events which were known but were expected to be within airmanship of pilots to identify. The pilots' reaction contributed to the loss of other sources of airspeed information. Here we find

challenge of describing incidents without biasing the description against the favour of the operators (the pilots).

The report by the French aviation authority lists the loss of information by known causes which should have been interpreted by the pilots. The report into the incident (Ministère del'Écologie, 2012) describes that difficult flying conditions at high altitude in turbulence resulted in excessive manoeuvres by the pilot which then further exacerbated interpretability of the instrumentation. The report goes on to say that any errors on the part of the pilot flying were no longer subject to warnings which might be reasonably expected from the instrumentation. And this seems surprising. If the conditions can be articulated in a report, one might expect that they may be modelled in software. The report continues to describe how the combination of a stalled aircraft, the recovery manoeuvre, and the ergonomics of how the crisis is reported to the crew, was not conducive to effecting the correct behaviours in such a situation (with various recommendations for improved training). The report may therefore be interpreted as blaming the pilots for not behaving accordingly. However it is explains that the Spartan information which was available was on the one hand providing insufficient data to act on, and on the other, providing information to which they could not be expected to act reasonably with.

Let us take a moment to consider financial information systems. Much is made of the propensity for human failures to lie at the root of fraud – poor management of passwords, careless handling of documentation for example – that the hue and cry to educate users draws up longer and longer pieces of advice for the user to compensate for the computerization (not necessarily the automation) of business processes which were designed to be secure when face-to-face transactions were commonplace or at least allowed for an anti-Turing consideration (Turing, 1950) of who was at the other end of a telephone line or standing in line at a telegraph office. Operational activity will often require codification into checklists (Gawande, 2010) as an assurance process. But this requires discipline. Although it is unreasonable (Sasse and Johnson, 1999) to expect it of digital natives and indeed most digital immigrants, there is certainly a place for checklists in planes and hospitals to name but a few.

There is not a deterministic solution but there is a problem solving philosophy where intention is everything. Ashby (Pickering, 2002) observes that only variation can force variation down. It is therefore the responsibility of every system designer to create systems

with pathways that that can grow to face down the emergent properties of the many pathways (Dresner and Jones, 2014) represented by the owners, actors, and customers and their anti-images who would do harm to the system or misuse that system for their own nefarious objectives.

Furthermore organisational factors very often have a profound effect on both the delivered system and the design process (Loucopoulos, 2005). Successful innovations must not only be technically feasible and desirable to consumers but also viable from a business point of view (Brown 2007). A business has to develop features consistent with its goals if it is to become an effective competitor (Pearlson and Saunders, 2012). In other words, the alignment of enterprise and systems increases the likelihood that an Information System will be created that provides maximum value for the enterprise and the IS will be able to be supported. Computing devices, human interaction with them, and people within the organisation create variety where pathways will be traceable to a human action with a tendency to label this as cause and effect.

This paper puts forward a framework for early systems development that focuses on organisational factors, increasing business IS/IT alignment and reducing the likelihood of system failure and associated blame. The subsequent sections are organized as follows: section two discusses further background details related to Requirements Engineering (RE) and associated knowledge management; the framework is put forward in section three and conclusions drawn in section four.

2.0 Requirements Engineering and knowledge management

There is a clear need for a solid foundation to be established early in systems development. It is here – at the latest – when the definition of system pathways will begin to emerge. Important detail missed at an early stage risk large problems later in development and subsequent system failure leading to the attribution of blame. Requirements Engineering (RE) could be compared to the construction of a house, in which the process of laying the foundations is particularly significant for supporting the walls and columns. Similarly the RE process provides a base for further systems development or, in the case of system-intensive organisations, change management. Any faults or weaknesses at the early stages tend to produce cracks, potentially leading to at best increases in time and money and at worst

project destabilisation and failure. Underpinning the foundations, or amendment of requirements in the case of RE, can be performed to add further support, although often costly, difficult and inconvenient.

In spite of many research efforts and the development of a range of RE techniques for a systems' functionality, system failures still continue to be attributed to, among other factors, requirements issues. When defining functional and non-functional requirements individuals often consider only their personal requirements, without thinking about the company's overall goals (Chmura and Crockett, 1995). Currently if an organisation practices strategic planning at all, a gap often exists between its strategic statements and the corresponding Information Systems (Chmura and Crockett, 1995). Like the acorn contains the fractal formula for the spreading oak, so we must imbue our systems development with the formulae to find implicit requirements amongst the explicit and control the emergent requirements so that they are duly implemented with respect to the original objectives of the system…as they continue to be derived.

Why should we want to do this? Because the prediction of pathways through any hardware, software, and wetware combination is going to be challenging. Work by Beer and Ashby (Pickering, 2002) in cybernetics demonstrate that to control those pathways, so that the original fractal can continue to grow safely, is another order-of-magnitude-challenging. The popular expectation – perhaps misconception - of computing power is determinism but the route to that end will comprise non-deterministic meanderings. To enable communications along that route through the pathways must be nurtured and curated – orchestrated - to enable control to take place.

'Knowledge itself is power' (Bacon, 1597, p.69). From an organisational point of view, properly used knowledge assets, for example, underlying skills, routines, practices, principles, formulas, methods, heuristics and intuition, whether explicit or tacit, enable an organisation to improve its efficiency, effectiveness and profitability (Jessup and Valacich, 2006). These assets can also help to create a sustainable competitive advantage (Nonaka et al.. 2005). Furthermore knowledge is recognised as one of the most important sources of innovation and new customer value propositions, emanating from individual, organisational and communal knowledge creativity and utilisation (Leibold et al., 2005).

The world has become too complex for individuals to have the knowledge necessary to tackle problems by themselves (Fischer, 2007). Reports of an accident at the Alton Towers theme park (Financial Times, 2015) ascribed blame to the ride operator misinterpreting a system message and allowing a ride to continue in a dangerous state. One must question if the design of the system put responsibility for hazardous-to-life situations on an operator who was not supported by electromechanical constraints to prevent operations in an unsafe environment.

Existing approaches typically concentrate on the representation of contextual knowledge about the usage world and neglect the system and subject or domain worlds (Pohl and Haumer, 1997). Developers do not always understand the problem domain, particularly at the beginning of a project, making it difficult to communicate (Ambler, 2016). Knowledge of the domain tends to be ad-hoc and thinly spread throughout the development team, potentially leading to misinformed models, conflict and ambiguities together with a compromise in productivity and quality (Curtis et al., 1988). This knowledge can be spread over a plethora of documentation, artefacts and technologies and stored in the minds of stakeholders. Indeed knowledge which is relevant to complex problems is often distributed among many people (Fischer, 2007 and Carr et al., 2003). Much of this knowledge is tacit in nature and needs to be made explicit in order to be used within RE. Gruenbacher and Briggs (2001) observe that if tacit stakeholder knowledge remains hidden the following problems occur: incomplete requirements because 'obvious' ideas are not captured; reduced ability to identify conflict because not all project-relevant knowledge is explicitly available; conflicting interpretations due to terminology differences; hidden stakeholder expectations and assumptions not explicitly available. Through conceptualisation, elicitation and ultimately articulation, typically in collaboration with others, some proportion of a person's tacit knowledge may be captured in explicit form (Marwick, 2001).

Jackson (1995) attributes stakeholders' requirements, which cannot be focused on, to the incorrect identification of the application domain during conceptual modelling. (Dano et al., 1997) notes that the description of Use Cases in natural language facilitates communication between analyst and domain expert, but increases the risk of ambiguity, inconsistency and incompleteness due to the variance of word meanings. In order to avoid such problems with natural language, it is important to use a more structured or formal technique for their description. EasyWinWin (Gruenbacher, 2000) provides clarity of the meaning of terms through the use of a glossary, enabling tacit knowledge to be shared among stakeholders.

(Weidenhaupt et al., 1998) also use a glossary by intertwining it with scenarios, facilitating a common understanding of terms among different stakeholder groups. In particular new project members were able to become familiar with project terminology. Fensel (2010) supports the provision of shared and common domain structures which are becoming essential and a key asset in information exchange.

3.0 A framework for elaborating requirements

The proposed conceptual framework aims to enable stakeholders to consider the multitude of issues relating to the impact of requirements on the organisation and societal environment during the early stages of development. The ultimate aim is the delivery of useful and sustainable systems that are aligned to enterprise strategy, together with reducing errors later in development resulting in the attribution of blame.

The proposed Requirements Elaboration Framework is shown as a meta-model in Figure 3.

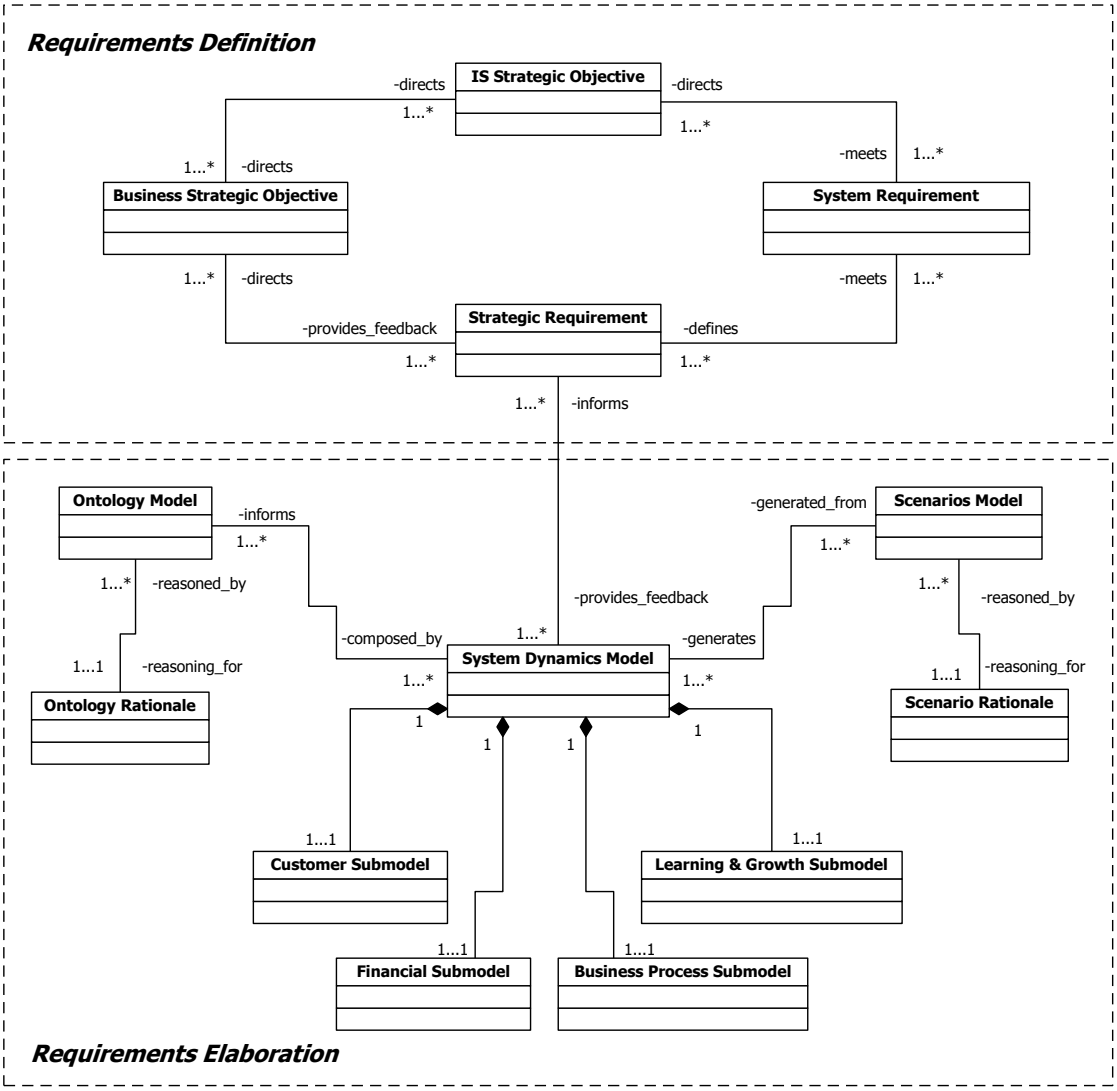

Figure 3 Requirements Elaboration Framework meta-model

The framework is comprised of two parts, namely *requirements definition* and *requirements elaboration*. It is considered that there are many existing methods (e.g. RUP, Booch, Agile) and tools (e.g. Together, Rationale Rose) that address the *requirements definition* issue, whereas *requirements elaboration* lacks support.

Requirements definition takes place prior to *requirements elaboration*. *Business strategic objectives* direct the composition of *strategic requirements.* *Strategic requirements* can inform *strategic objectives* following *requirements elaboration*. *Strategic requirements* are met by differing Information System *functional* and *non-functional requirements*, depending on the IS strategy. Goal decomposition graphs traditionally provide the pre-traceability

between high level strategic concerns and low level technical constraints; therefore facilitating the propagation of business changes onto system features (Rolland, 2005). However the proposed framework raises the point that *system requirements* are not simply derived linearly through an analysis of business goals but often these business goals are influenced through alternative implementation choices (Loucopoulos, 2009).

Requirements elaboration aims to facilitate the systematic modelling and evaluation of requirements on the organisation during early development for the purpose of aligning the organisation and Information System. The components and relationships between components within *requirements elaboration*, namely *System Dynamics modelling*, *ontology modelling*, *scenario modelling* and *rationale modelling*, are discussed in the following subsections.

3.1 System Dynamics Model

The development of a *System Dynamics model* is central to the process of elaborating requirements, providing a means for a shared understanding of the impact of requirements amongst stakeholders. (Gero and Smith, 2007) suggest that the agent for design should be dynamic and able to handle change in order to cope with the exploration of complexity that characterises design. In contrast the static nature of conceptual models does not reflect the relationships that comprise business reality which are non-linear and dynamic. The use of System Dynamics (Forrester, 1998; Sterman, 2000; Morecroft, 2015) is particularly suitable in meeting these requirements. "The System Dynamics approach is also pertinent to the engagement of multiple stakeholders due to its iterative nature and the enablement of the consideration of problems and causes within the system context" (Garfield and Loucopoulos, 2009). "Such modelling also assists in reducing biases, uncertainties and conflicts amongst stakeholders together with forming a foundation for the development of scenarios" (Loucopoulos and Garfield, 2009, p.355).

The model is constructed from *strategic requirements* and describes the dynamics of the organisation and its interaction to the proposed system. *System Dynamics model* development is informed by the *ontology model.* Through its relationship to *strategic requirements*, it can provide feedback on concepts such as legislation, finance, resources etc, that would be of value to the analysis of requirements and by extension to *business objectives*

(Loucopoulos and Garfield, 2009). The *System Dynamics model* is also used as the structure upon which alternative scenarios can be generated and explored.

Its method of construction is intended to enable the testing of key system parameters under different conditions and subsequently the observation of a holistic view of system behaviour under these conditions. "This implies that the model needs to be constructed in such a way so as to permit multiple interpretations of it" (Loucopoulos and Garfield, 2009, p.355). The construction of the model is therefore informed by *system functional* and *non-functional requirements* according to differing Information Systems strategies to support multiple interpretations.

The *System Dynamics model* is also intended to describe the feedback between various system components. Taking this into consideration and that a business strategy is likely to be influenced by different perspectives, the approach encourages the development of four interlinked *sub-model viewpoints*. These are similar to that suggested by the Balanced Scorecard (BSC) (Kaplan and Norton, 1992), namely customer, financial, business processes and learning and growth. The framework provided by the Balanced Scorecard enables an organisation's vision and mission to be translated into measurable parameters, which are largely indicators of future performance.

The consideration of such perspectives provides a comprehensive view of a business. Therefore by forcing senior managers to consider all important operational measures together, the Balanced Scorecard lets them see whether improvement in one area may have been achieved at the expense of another (Kaplan and Norton, 1992). In contrast to the BSC, earlier methodologies focused on financial measures alone, such as Activity-Based Costing (Consortium-for-Advanced-Manufacturing-International, 2009). Financial figures alone suffer from two major drawbacks: they are historical; it is common for the current market value of an organisation to exceed the market value of its assets. Managers want a balanced presentation of both financial and operational measures (Kaplan and Norton. 1992).

The use of System Dynamics, which focuses exclusively on feedback structures, provides an excellent vehicle for evaluating the interrelations between sub-model viewpoints. In particular feedback within and between sub-model viewpoints can provide means for stakeholders to visualise trade-offs, by considering all important operational measures

together. (Bianchi and Montemaggiore, 2008) matched the System Dynamics methodology with the Balanced Scorecard framework in the Dynamic Balanced Scorecard (DBSC). A case study of an Italian city water company was used to demonstrate benefits of its use for enhancing strategy design and planning. The use of a dynamic Balanced Scorecard enabled managers to better understand cause and effect relationships between variables pertaining to the four traditional Balanced Scorecard perspectives. Furthermore it successfully enhanced "managers' learning and capability to identify causal relationships between policy levers and company performance, and better communicate strategy with stakeholders" (Bianchi and Montemaggiore, 2008, p.200).

3.2 Ontology Model

The explicit representation and management of knowledge in the conceptual framework, in the form of an *ontology model*, provides a shared stakeholder understanding of concepts relating to the application and enterprise domain. This is particularly useful as stakeholders' knowledge is not uniform and different meanings can be attached to the same concept. Furthermore technical jargon can act as a barrier to communication (e.g. (Knott et al., 2000)) and conceptual misunderstandings can block or distort communication (Burton, 1980). Indeed it is difficult to carry out meaningful designing activities during model development if the domain is unclear. This fulfils the need for a shared understanding of knowledge advocated by (Fischer, 2007), who notes that the world has become too complex for individuals to have the knowledge necessary to tackle problems by themselves. Knowledge that is thinly spread throughout the development team, which is frequently the case, can lead to misinformed models, conflict and ambiguities together with a compromise in productivity and quality (Curtis et al., 1988). Coupled with this it is difficult for developers to understand the problem domain at the beginning of a project (Ambler, 2016). Therefore formalised knowledge can assist in solidifying concepts within the application and enterprise domain. Wiegers (2003) notes that analysts that understand the application domain often detect unstated assumptions and implicit requirements together with suggesting ways that users could improve their business processes and minimise miscommunication with users.

The *ontology model* provides a strategic context for requirements. It is used for the articulation and negotiation of concepts during the designing and development of the *System Dynamics model.* This increases the probability that all relevant knowledge is used when requirements are modelled.

Structuring knowledge makes the identification and location of concepts by stakeholders easier. An ontology is considered a particularly appropriate way of representing and structuring knowledge, due to its knowledge management capabilities and encouragement of the standardisation of terms used to represent knowledge about the domain (Chandrasekaran et al., 1999; Jurisica et al., 1999). If knowledge is not formally structured to inform decisions during RE, it is easy to overlook important details and leave modelling more prone to conflicts, misunderstandings and incompleteness (Loucopoulos and Garfield, 2009). Concepts are represented and structured in a semantically rich superclass-subclass hierarchy of invariant components related to the domain in the *ontology model*. Properties, e.g. object and data type, together with individuals or instances, form the basis of each class. Assertions (e.g. restrictions/constraints) and rules assist in determining relationships between concepts.

The *as_is ontology* can be an existing ontology or created from new. The structure of the classes of this ontology use a typical business structure, in the form of actors, goals, processes and objects and are linked via assertions. This enables different business attributes to be formalised and structured.

3.3 Scenarios Model

The *scenarios model* facilitates stakeholders' evaluation of the trade-off effects of different possible requirements futures among on the organisation and societal environment for the determination of strategic viability. The ultimate aim being the alignment of the organisation and IS.

The behaviour of a range of alternatives under different conditions needs to be enabled in order to understand how an organisation changes over time, in relation to the proposed IS. "This means that areas for improvement can be identified, new ideas tested and most importantly get an understanding of how a system works without taking any significant risks" (Loucopoulos and Garfield, 2009, p.355). Indeed in order to negotiate, a range of possibilities need to be explored (Easterbrook, 1991). Furthermore the generation of ideas and alternative solutions on early design questions can assist with stakeholder engagement (International-Finance-Group, 2007), by providing a catalyst for communication, helping to bridge the gap between various stakeholders and requirements engineers (Pohl and Haumer, 1997). The observation of the patterns of behaviour provide an indication of the basic feedback structures during the period covered (Sterman, 2000).

Scenarios enable a reduction in complexity (Weidenhaupt et al., 1998). Furthermore (Rowe et al. 1994) observe that scenarios have become increasingly powerful tools for developing strategic vision within organisations and for assisting executives to identify critical future paths. Scenarios can also help to balance the need for flexibility and informality with reference to creativity, as there is a need to progress systematically toward creating effective and usable computer systems and applications (Carroll, 1995, p.15).

Experience has shown that stakeholders have difficulties in comprehending qualitative models and, even more significantly, have difficulties in extrapolating from the model to the potential behaviour of the system according to different design options available to them (Loucopoulos and Prekas, 2003). "Whilst qualitative-based conceptual modelling approaches seem to be an improvement on purely linguistic-based approaches, they fail to bridge the communication gap between client stakeholders and analysts" (Loucopoulos and Garfield, 2009, p.357). There is therefore a need for the quantitative analysis of requirements to determine essential elements of the system and possible measures of future performance on elements of the system. The quantitative dimension of System Dynamics compliments its qualitative nature, enabling stakeholders to subject the model to simulation. The alteration of critical variable values facilitates the testing of scenario behaviour under different conditions. Nevertheless qualitative and quantitative properties are not separate concerns and both are required to support business scenario analysis (Lang, 2000).

3.4 Ontology and Scenario Rationale Model

"Rationale provides a way of documenting decisions together with underlying arguments, promoting critical reflection, negotiation and location of inconsistencies" (Garfield and Loucopoulos, 2009). Essentially it provides stakeholders with reasoning regarding how and why decisions are made, which frequently is not the case during modelling activities (e.g. (Conklin and Yakemovic, 1991; Gotel and Finkelstein, 1994; Pohl, 2010; Ramesh et al., 1997)). Stakeholders are able to understand one another's perceptions of the negotiation together with the point of view from which the system is constructed (Kangassalo, 1999). In contrast an undefined, unclear rationale is more likely to be associated with poor design (Burge and Brown, 2000), and can lead to assumptions about a model that are conflicting. This can potentially increase uncertainties, leading to the delivery of a flawed system. Stakeholders need to be able to visualise theirs and others arguments (Karacapilidis and

Papadias, 2001). Rationale provides a mechanism for representing assumptions explicitly. Visualisation of arguments assists with stakeholder communication. In particular visualisation of negotiation information facilitates stakeholders' understanding of negotiations through the simplification of the complex and massive negotiation of data (In and Roy 2002).

The type of rationale selected for this research takes the form of collaborative visualised argumentation, based on the principles of (Rittel and Webber, 1973). This form of rationale is particularly suitable as it consists of the problems and issues that arise in the course of a design, along with pros and cons for each alternative (Shipman and McCall, 1997). The collaborative element is applicable to the negotiation process, which needs to take place in a collaborative environment to allow the elicitation of all relevant alternatives and arguments (e.g. (Easterbrook, 1991; Robinson and Fickas, 1994; Pohl, 2010)).

Rationale provides support for traceability. This is important as the development process leading to the requirements specification must be traceable, so that the requirements themselves can be understood, i.e. to trace a requirement back to its origin (Pohl, 2010). It must also be assured that the life of every single requirement can be reconstructed and that people not involved in the process can understand how and why the requirements specification was produced in a particular way (Pohl, 2010).

The following two sub-sections outline the rationale components (*ontology rationale* and *scenario rationale*), which work in parallel with ontology and scenarios in the proposed framework. Both *ontology* and *scenario rationale* are a by-product of the design process, rather than a separate task which, as suggested by (Burge and Brown, 2000), reduces the workload for stakeholders.

4.0 Conclusions

One is reminded of the poet John Donne's (2012) assertion that 'no man is an island' from the aptly named 'Devotions Upon Emergent Occasions'. And apart from extreme cases where evidence points to deliberate intervention of an operator (pilot) to destroy 'the system' (the plane) such as the Germanwings disaster (2015), and crashes in Namibia (2013) and a flight from Indonesia to Singapore (1997) (BBC, 2015). If the computer cannot understand what's going on - the system is outside the envelope of normal operations - and the computer passes

control back to the operator, does the operator appreciate that computer-controlled safeguards may be neutralised too? Although one of course would not advocate the flying of a plane with what is essentially an instruction manual in one hand, there is a case to model the correct actions in training situations using checklists where patterns are known (Gawande, 2010). These remove the reliance on the need to hastily remember vital sequence-based information when under pressure. Without deliberate, probably malicious, intention there is no fair attribution of blame to the focal actor in the software life cycle at the time of the incident.

We should question the design decisions which lead to the segregation of requirements into those which are designed in and those which are declared as within the decision making capability of the operator. As for example happened with Air France 447 where the loss of the first air speed indicator to ice crystals was declared within the knowledge of pilots to identify and therefore not programmed into any of the Airbus A330-203 warning systems.

By enabling reasoning and decision-making to be made explicit (i.e. through the *ontology* and *scenario rationale* in the proposed framework) and formalising knowledge within the *ontology model*, the understanding of the system and specification can begin to loose its traditionally opaque nature and potentially assist in the reduction of system failure and associated blame. Furthermore stakeholders can become familiar with alternative futures in the *scenarios model*. The *ontology model* can assist with the typically unsure nature of stakeholder system views, through detailing what already exists within the enterprise domain, to provide a benchmark for innovations.

Have we chanced upon the dilemma that to blame human error is to blame being human. The brief to the engineer is not to design with unnatural expectations in mind. The desirable system will comprise systems which restrict genuine errors without operators becoming blasé to the frequent alerts that these will generate.

References

Adams, A. and Sasse, M. A. (1999) *Users are not the enemy*, Communications of the ACM, 42(12) 40-46.

Ambler, S. W. (2016) *Overcoming requirements modelling challenges. Agile modelling*. [online]. Available from: http://www.ambysoft.com/onlineWritings.html [Accessed 21 February 2016].

Bacon, F. (1597) Meditationes sacrae.

Barzun, M. (2014) *Keynote: International perspectives*, Information Assurance Advisory Council Symposium, London, 11 September 2014.

BBC (2015) Germanwings plane crash: Co-pilot 'wanted to destroy pilot' [online] Available at: http://www.bbc.co.uk/news/world-europe-32063587 [Accessed 21 February 2016].

Bianchi, C. and Montemaggiore, G. B. (2008) *Enhancing strategy design and planning in public utilities through "dynamic" Balanced Scorecards: insights from a project in a city water company*, System Dynamics Review, 24(2) 175-213.

Bourque, P. and Fairely, R.E. (2014) SWEBOK v3.0. Guide to the Software Engineering body of knowledge, IEEE Computer Society, London.

Brown, T. (2007). Strategy by design: how design thinking builds opportunities. *Stanford Breakfast Briefings*, Stanford University, Stanford, California, USA. 19 December 2007.

Burge, J. and Brown, D. C. (2000) *Reasoning with design rationale*, In: J. Gero, ed. Artificial Intelligence in design, Kluwer Academic Publishing, London. 611-629.

Burton, S. H. (1980) People and communication. Longman Group Ltd, Harlow, Essex.

Carr, M.J., Konda, S.L., et al. (2003) Taxonomy-based risk identification, Software Engineering Institute.

Carroll, J. M. (1995) *Introduction: the scenario perspective on system development*. In: J. M. Carroll, ed. Scenario-based design: envisioning work and technology in system development. John Wiley & Sons, Inc., Chichester, 1-18.

Chandrasekaran, B., Josephson, J. R. and Benjamins, V. R. (1999) *What are ontologies, and why do we need them?* IEEE Intelligent Systems, 14(1) 20-26.

Checkland P. (1985) *From optimizing to learning: a development of systems thinking for the 1990s*, Journal of the Operational Research Society, 36 (9) 757 – 767.

Checkland P. and Holwell, S. (1998) Information Systems and Information Systems: making sense of the field, Wiley and Sons, London.

Chmura, A. and Crockett, H. D. (1995) *Tools to align goals and Information Systems*. IEEE Software, 12(3) 108-111.

Conklin, E. J. and Yakemovic, K. C. B. (1991) *A process-oriented approach to design rationale*, Human-Computer Interaction, 6(3 & 4) 357-391.

Consortium-for-Advanced-Manufacturing-International. (2016) *Global leadership in cost, process and performance management*. [online]. Available from: http://www.cam-i.org [Accessed 21 February 2016].

Curtis, B., Krasner, H. and Iscoe, N. (1988) *A field study of the software design process for large systems*. Meta Models for Requirements Engineering. Nature Team.

Dano, B., Briand, H. and Barbier, F. (1997) *An approach based on the concept of Use Case to produce dynamic object-oriented specifications*. Proceedings of the 3rd IEEE international symposium on Requirements Engineering, Annapolis, Maryland, USA. 6-10 January 1997.

Donne, J. (2012) Devotions upon emergent occasions and death's duel, CreateSpace Independent Publishing Platform, London.

Dresner, D. and Jones, N. (2014) The three laws of cyber and information security, Softbox.

Easterbrook, S. (1991) *Handling conflict between domain descriptions with computer-supported negotiation*, Knowledge Acquisition: An International Journal, 3(3) 255-289.

Fensel, D. (2010) Ontologies: a silver bullet for knowledge management and electronic commerce, 2nd ed, Springer, Berlin.

Financial Times (2015) 'Human error' at root of Alton Towers accident, 24 November 2015 [online] Available at: http://www.ft.com/fastft/2015/11/24/human-error-root-of-alton-towers-accident/ [Accessed: 21 February 2016].

Fischer, G. (2007) Meta-design: coping with ill-defined problems and emerging requirements in collaborative design. *Design requirements workshop*, Case Western Reserve University, Cleveland, Ohio, USA. 3-6 June 2007.

Forrester, J. W. (1998) Designing the future, Sevilla, Spain: Universidad de Sevilla.

GAO (1992) Patriot missile defence, software problem led to system failure at Dhahran, Saudi Arabia, Information Management and Technology Division, Washington.

Garfield, J. & Loucopoulos, P. (2009) *Requirements elaboration for system co-development,* Ingénierie des Systèmes d'Information (ISI), 14(4) 77-98.

Gawande, A. (2010) *The Checklist Manifesto*, Profile Books, London.

Gero, J. S. and Smith, G. J. (2007) *A cognitive and computational basis for designing,* International conference on engineering design, *ICED'07*, Paris, France. 28-31 August 2007.

Gotel, O. C. Z. and Finkelstein, C. W. (1994) *An analysis of the requirements traceability problem*. Proceedings of the 1st international conference on Requirements Engineering, Colorado Springs, Colorado, USA. 18-22 April 1994.

Gruenbacher, P. (2000) *Collaborative requirements negotiation with EasyWinWin*. Proceedings of the 11th international workshop on database and expert systems applications, Greenwich, London, UK. 4-8 September 2000.

Gruenbacher, P. and Briggs, R. O. (2001) *Surfacing tacit knowledge in requirements negotiation: experiences using EasyWinWin*, Proceedings of the 34th Hawaii international conference on system sciences, Maui, Hawaii, USA. 3-6 January 2001.

In, H. and Roy, S. (2002) *Visualisation issues for software requirements negotiation*, IEEE international computer software applications conference (COMPSAC), Chicago, Illinois, USA. 8-10 October 2002.

International-Finance-Group. (2007) *Stakeholder engagement.* A good practice handbook for companies doing business in emerging markets, International Finance Group, World Bank Group, Washington DC.

International Standards Organisation. (2002) *ISO/IEC 15288:2002 Information Technology – Life Cycle Management – System Life Cycle Processes.*

International Standards Organisation. (2008) *ISO 9001:2008. Quality management systems. Requirements.*

International Standards Organisation. (2008) ISO/IEC 12207:2008 *Information technology — Software life cycle processes*, International Standards Organisation

International Standards Organisation. (2009) *ISO 31000:2009 Risk management — Principles and guidelines.*

International Standards Organisation. (2015) *ISO/IEC 38500:2015 Information technology — Governance of IT for the organization*.

Jackson, M. (1995) Software requirements and specifications. Addison Wesley, Harlow.

Jessup, L. M. and Valacich, J. S. (2006) Information Systems today, 2nd ed., Pearson Education Inc, New Jersey.

Jurisica, I., Mylopoulos, J. and Yu, E. S. K. (1999) *Using ontologies for knowledge management: an Information Systems perspective*, Annual conference of the American Society of Information Sciences, Washington, DC, Columbia, USA. 1-4 November 1999.

Kangassalo, H. (1999) *Are global understanding, communication and information management in Information Systems possible? A conceptual modelling view - problems and proposals for solutions*. In: P. P. Chen, J. Akoka, H. Kangassalo and B. Thalheim, eds. Lecture notes in Computer Science, conceptual modelling: current issues and future directions. 1565. 105-122.

Kaplan, R. S. and Norton, D. P. (1992) *The Balanced Scorecard - measures that drive performance*, Harvard Business Review, 70(1) 71-79.

Karacapilidis, N. and Papadias, D. (2001) Computer supported argumentation and collaborative decision making: the HERMES system, Information Systems, 26(4) 259-277.

Knott, R. P., Merunka, V. and Polak, J. (2000) *Process modelling for object-oriented analysis using BORM object behavioral analysis*, Proceedings of the 4th IEEE international conference on Requirements Engineering, Schaumberg, Illinois, USA. 19-23 June 2000.

Lang, K. (2000) *Simulation of qualitative models to support business scenario analysis*, Proceedings of the 18th international conference of the System Dynamics Society, Bergen, Norway. 6-10 August 2000.

Leibold, M., Probst, G. and Gibbert, M. (2005) Strategic management in the knowledge economy. New approaches and business applications. Publicis Corporate Publishing and Wiley-VCH-Verlag GmbH & Co KGaA, Erlangen, Germany.

Loucopoulos, P. (2005) System co-development through requirements specification. In: O. Vasilecas, W. Wojtkowski, J. Zupančičet al, eds. Information Systems development: Advances in theory, practice and education, Springer-Verlag, Berlin, 1-13.

Loucopoulos, P. (2009) *Requirements intertwining: introduction*, In: K. Lyytinen, P. Loucopoulos, J. Mylopoulos and W. Robinson, eds. Design Requirements Engineering: a ten-year perspective. Design requirements workshop, Cleveland, Ohio, USA. Springer-Verlag Berlin Heidelberg, Berlin. Lecture Notes in Business Information Processing 14, 302-304.

Loucopoulos, P. and Garfield, J. (2009) *The intertwining of enterprise strategy and requirements*. In: K. Lyytinen, P. Loucopoulos, J. Mylopoulos and W. Robinson, eds. Design Requirements Engineering: a ten-year perspective. Design requirements workshop, Cleveland, Ohio, *USA.* Springer-Verlag Berlin Heidelberg, Berlin, Lecture Notes in Business Information Processing 14. 352-373.

Loucopoulos, P. and Prekas, N. (2003) *A framework for Requirements Engineering using System Dynamics*, Proceedings of the 21st international conference of the System Dynamics Society, New York City, New York, USA. 20-24 July 2003.

Marwick, A. D. (2001) *Knowledge management technology*, IBM Systems Journal, 40(4) 814-830.

Miller, K., Camp, T., Smith King, L., Johnson, D., and Moskal, B. (2016) Computing Cases.org. [online] Available from: http://computingcases.org/ [Accessed 21 February 2016].

Ministère del'Écologie (2012) *Final report on the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro – Paris*, Ministère del'Écologie, du Development durable, des Transports et du Logement.

Morecroft, J. (2015) Strategic modelling and business dynamics, John Wiley & Sons Ltd, Chichester, UK.

Nonaka, I., Toyama, R. and Konno, N. (2005) *SECA, Ba and leadership: a unified model of dynamic knowledge creation*. In: S. Little, P. Quintas and T. Ray, eds. Managing knowledge, 2nd ed. SAGE Publications Ltd, London, 41-67.

Pearlson, K. E. and Saunders, C. S. (2012) Managing and using Information Systems: a strategic approach, 2nd ed. John Wiley & Sons Inc, New Jersey.

Pickering, A. (2002) *Cybernetics and the mangle: Ashby, Beer and Pask*, Social Studies of Science, 32 413-437.

Pohl, K. (2010) Requirements engineering: fundamentals, principles and techniques, Springer, London.

Pohl, K. and Haumer, P. (1997) *Modelling contextual information about scenarios*, Proceedings of the 3rd international workshop on Requirements Engineering: Foundations for software quality (REFSQ'97), Barcelona, Spain. 16-17 June 1997.

Ramesh, B., Stubbs, C., Powers, T. and Edwards, M. (1997) *Requirements traceability: theory and practice*, Annals of Software Engineering, 3 (September) 397-415.

Rittel, H. W. J. and Webber, M. (1973) *Dilemmas in the general theory of planning*, Policy Science, 4(2) 155-169.

Rittel, H. W. J. and Webber, M. (1984) *Planning problems are wicked problems*. In: N. Cross, ed. Developments in design methodology. John Wiley and Sons, Chichester, 135-144.

Robinson, W. N. and Fickas, S. (1994) *Automated support for requirements negotiation*, AAAI-94 Workshop on models of conflict management in cooperative problem solving, Seattle, Washington, USA. 4 August 1994.

Rolland, C. (2005) *Reasoning with goals to engineer requirements*. In: O. Camp, J. B. Filipe, S. Hammoudi and M. Piattini, eds. Enterprise Information Systems V. London: Kluwer Academic Publishers, 12-20.

Rowe, A. J., Mason, R. O., Dickel, K. E. and Mann, R. B. (1994) Strategic management. A methodological approach. 4th ed. Addison-Wesley Company, Reading.

Sasse, M.A. and Johnson, C. (1999) Human-computer interaction INTERAGT '99 - IFIP TC.13 International Conference on Human-Computer Interaction, 30th August-3rd September 1999, Edinburgh, UK - Editors' preface: A perspective on failure. International Conference on Human-Computer Interaction, Edinburgh (INTERACT 99).

Shipman, F. M. and McCall, R. J. (1997) *Integrating different perspectives on design rationale: supporting the emergence of design rationale from design communication*, Artificial Intelligence for Engineering Design Analysis and Manufacturing, 11(2) 141-154.

Sterman, J. D. (2000) Business dynamics, systems thinking and modelling for a complex world. Irwin McGraw-Hill, London.

Turing, A. M. (1950) *Mind,* Computing Machinery and Intelligence, 59 (236) 433-460.

Weidenhaupt, K., Pohl, K., Jarke, M. and Haumer, P. (1998) *Scenario usage in system development: a report on current practice*, IEEE Software, 15(2) 34-45.

Wiegers, K. E. (2003) *So you want to be a requirements analyst?* Software Development, 11(7) 1-6.

Wiener, N. (1961) Cybernetics or control and communication in the animal and the machine, 2nd ed., MIT Press, Cambridge, Massachusetts.

Zola, E. (1958) J'accuse "...Lettre ouverte au Président de la République, 13 Janvier 1898, Edition du cinquantenaire, Fasquelle.