An artifact-centric framework for software development skills

Jack Downey, NORAH POWER

An Artifact-centric Framework for Software Development Skills

Jack Downey
University of Limerick
Castletroy
Co. Limerick, Ireland
+353 61 213072
jack.downey at ul.ie

Norah Power
University of Limerick
Castletroy
Co. Limerick, Ireland
+353 61 202769
norah.power at ul.ie

ABSTRACT
While the specific knowledge, skills and abilities needed to develop software can be determined, it is much more difficult to decide what skill set is required for any given software development role. This paper suggests that progress may be made if, instead of trying to relate knowledge, skills or abilities to individual roles, efforts are made to understand what knowledge, skills and abilities are required to create and use the artifacts associated with software development. To this end, a framework incorporating two relationships is presented: The first relates software development artifacts to organizational functions, while the second relates knowledge, skills and abilities to different phases of an artifact's lifecycle. This framework leads to a new taxonomy of skills.

Categories and Subject Descriptors
K.7.1 [**Occupations**]:

General Terms
Human Factors, Theory

Keywords
Irish Telecommunications Software, Knowledge, Skills, Abilities, Software Artifacts, Grounded Theory

1. INTRODUCTION
The question of what knowledge, skills and abilities (KSAs) are needed to develop software is an important one and the answer is not simple. Because professional software development is a complex task, many different skills are needed, skills that are not necessarily possessed by a single individual.

Because of its inherent complexity, software development is an activity that is normally carried out by teams of workers who are assigned to different roles in the team, such as analyst, architect, programmer, or tester [1]. Employers engaged in software development generally recruit and hire personnel to fill specific software development roles and expect them to have the skills needed to fill those roles. But this is problematic for a number of reasons.

Role-names differ from one organization to another, making it hard to identify similarities and differences. Role fragmentation has led to considerable overlap between roles, so the same skill may belong to different roles, and roles are not necessarily distinguishable by their associated skills. In general, it is not easy to determine which set of skills is needed to fulfill any particular software development role.

Researchers have tried to focus on specific roles in tackling this problem, but this approach has highlighted the gap between research and practice in software development. For example, the term 'requirements engineer' occurs repeatedly in the research literature of software requirements engineering [2], yet few if any software development organizations employ anyone called a requirements engineer. Instead, the skills associated with requirements engineering may be found in various roles such as systems analyst, business analyst, system architect, product manager or analyst/programmer.

In many organizations, requirements engineering is done by systems analysts. Downey [3] analyzed the difference between the roles of the systems architect and the systems analyst and found many similarities. The assignment of role names seems to depend more on the industry than on the responsibilities of the role or the skills employed.

Previous efforts to identify software-related skills have tended to avoid rather than deal with problems with role identification. For instance, the British Office for National Statistics places team leaders, systems architects, software developers and testers under the same heading – standard occupational classification code 2132 [4]. Similarly, Irish studies (that make use of the British codes) offer recommendations for the Information and Communications Technology (ICT) sector as a whole [5, 6].

Professional bodies have also avoided role distinctions. The Irish Computer Society [7] is promoting the use of the Skills Framework for an Information Age [8]. This framework provides a list of seventy-eight skills and asks practitioners to assess themselves on each one. They can then plan their development by improving existing skills or by gaining new ones. However, what

constitutes an ideal skills profile for a particular role is not discussed.

This paper presents an alternative approach, based on the idea that software development is concerned with the production of various artifacts, including the software itself. The remainder of the paper is organized as follows: Section 2 describes the research that led to the development of this framework. Section 3 summarizes the types of artifacts that are used. Section 4 deals with artifact dynamics. Section 5 relates artifacts to development functions and then Section 6 relates artifacts to knowledge, skills and abilities. Finally Section 7 summarizes the artifact-centric framework.

2. Research Approach

This research study set out to investigate the skills required to develop software in the telecommunications domain. The research design is fully described in Downey [9] and is only briefly summarized here. Individual members of project teams in four different telecommunications software companies were interviewed and the resulting data were analyzed qualitatively using a grounded theory approach. As a result, it was found that software development roles differ widely between companies (even between projects) and they also overlap significantly with other roles. It was concluded that software development roles cannot be defined in a generally applicable manner.

Further analysis of the data showed that the activities and the artifacts of the software development process were largely the same across each of the companies studied. Because the companies follow variations of the familiar 'V' model [10], it was clear that the study of project phases would simply explore well covered territory. However, changing the focus to the artifacts associated with the development process proved to be much more informative, leading to a conceptual framework where artifacts are central.

Artifacts have been studied in the software literature before. For instance, Cluts [11] describes artifacts as the means of relating people and activity systems. Artifacts also hold the history of those relationships within them. Maurizio, Stamelos and Tsoukias [12] are concerned with the attributes of software artifacts, arguing that these can be measured and these measurements used to support the decision-making process. A particular type of artifact, called a 'boundary object' is the focus for Mambrey and Robinson's [13] study. Such objects "inhabit several intersecting social worlds *and* satisfy the informational requirements of each" (p.119). As will be seen, most of the artifacts identified in this study are boundary objects, providing interfaces between the development team and other departments within the organization, or between the organization and external entities.

Artifacts are used throughout a development project to embody stakeholder knowledge and contribute to the development process. No one member of the development team is involved in the creation of all the artifacts. Some are produced by other team members; some originate in other departments within the company and more are externally sourced – customer requests, for instance.

3. Artifacts used in telecommunications software development

Appendix 1 shows the full set of artifacts identified in this study of telecommunication projects and how they relate to one another. While that diagram is complex and extremely detailed, when the work is looked at from an organizational point of view, four distinct phases of work can be identified:

1. **Definition.** At the beginning of the project, much work is done to identify what exactly is being demanded by the marketplace. The culmination of this work is what is being termed here the engineering requirements document. This specifies the requirements in a testable and measurable format.

2. **Selection.** The products being developed by these companies can evolve in a variety of ways. Decisions must be made to choose the features that yield maximum revenue and provide the most customer satisfaction. As well as being commercially feasible, they must also be technically possible. All the factors influencing these decisions appear in feasibility reports. These reports contain input from sales and marketing people as well as technical contributions from the programming team. Once a feature is deemed feasible, it is placed on a product roadmap and scheduled to be developed as part of a release.

3. **Management.** Once a project release has been defined and the approval given to develop a set of features, the construction of the project is guided by the project plan. This consists of a schedule and a work breakdown structure. Some companies include mitigation and contingency plans to cope with identified risks. It should be noted that the definition and selection activities are carried out before the project officially exists.

4. **Construction.** The remainder of the project artifacts relate to the task of producing and installing the finished product. It must be emphasised that a commercial software product involves user manuals and training materials as well as working software.

Having identified four phases to development, a more abstract picture of the artifacts and their interaction is possible. Figure 1 outlines how the four principal artifacts relate to one another. The engineering requirements document is the result of the Definition phase; the feasibility report is the result of the selection phase; the project plan is the principal Management artifact and the installed product is the main outcome of the Construction phase.

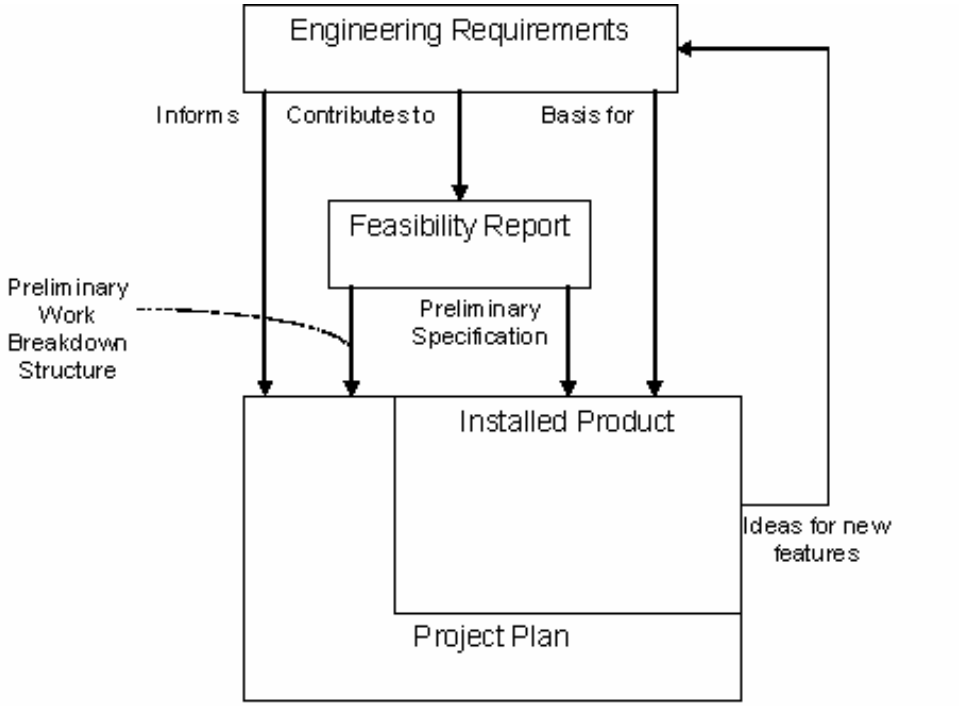


Figure 1. The Principal Artifacts

A noticeable feature of these key artifacts is the way so many different roles are associated with them. The project plan, for instance, is created by the project manager and receives input – such as time and headcount estimates - from architects, programmers, testers, technical writers and customer support personnel. It is also reviewed by the product manager. In effect, the artifact acts as a communications medium, allowing collaboration between parties, as well as supporting the decision-making process. In other words, the artifact functions as a boundary object, marking the interface between different organizational functions.

3.1 Artifact Dynamics

Artifacts represent milestones achieved during the project. However, they need to be created in a particular order. For instance, in the development projects described during the interviews, the feasibility study cannot take place until the requirements are understood and the construction effort cannot proceed without a project plan. Thus one major artifact depends on its predecessors, suggesting that development proceeds in a linear fashion. However, for each artifact, a cycle of activity needs to take place:

1. A trigger event occurs. This could be an external stimulus or the review of another artifact.
2. The goals for the triggered artifact must be defined. This process involves gathering the necessary information – other artifacts, for instance – and may trigger the creation of intermediate artifacts (such as prototype systems) to assist in the definition process.
3. The concept to be contained in the new artifact must be synthesized.
4. This concept must be articulated by producing an artifact.

This cycle resembles Kurt Lewin's experiential learning model [14]. The steps taken to learn something new from experience - where learning is defined "as a change in cognitive structure (knowledge)" [15, p.66] - involve having an experience, reflecting upon that experience, devising a course of action based on these reflections and then testing this course of action. The result of this experiment is analyzed in order to refine understanding, i.e. it triggers another learning cycle.

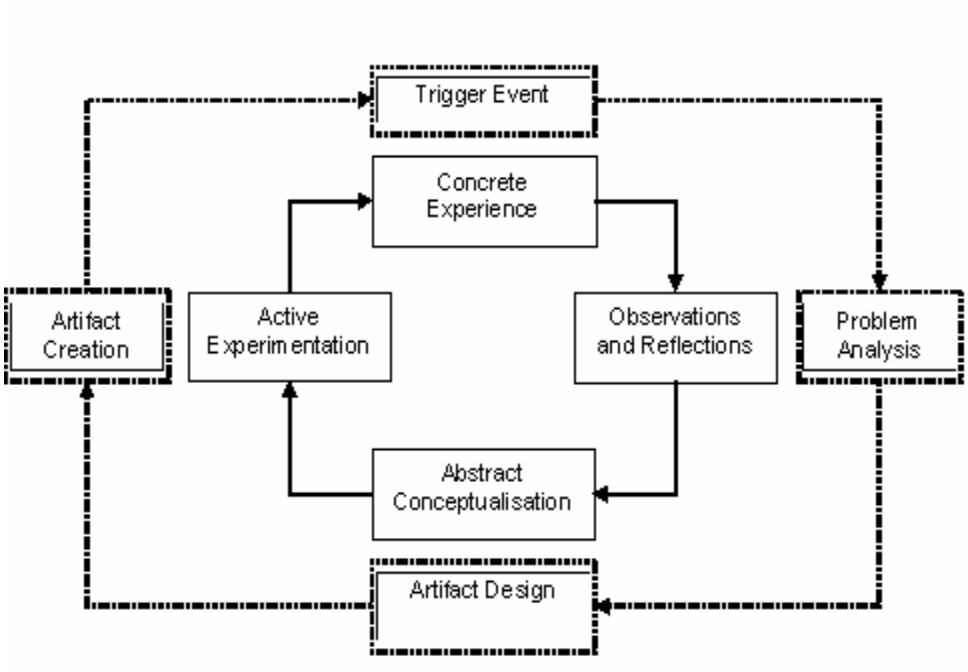


Figure 2. Artifact Dynamics

Similarly, the construction of an artifact involves some initial trigger; say a request from a customer for a new feature. Reflecting on the request, the practitioner may notice that several other customers have asked for similar functionality. S/he may also notice that the request is already on the product roadmap. In this case, the marketing requirements artifact already exists, so the course of action is to forward the requirements to the customer. If the customer finds these requirements lacking in some way, this triggers a possible revision of the marketing requirements. Alternatively, if the request has not been seen before, analysis of the request can result in a variety of actions.

If the request is clearly understood, then the practitioner will be able to design a new marketing requirements artifact by paraphrasing the original request. If the request is obviously unacceptable – i.e. it goes against the product strategy – a response to its originator must be framed, ideally pointing out how the product strategy will solve the problem in a different manner.

If the overall request looks interesting, but contains ambiguous elements, this will trigger a series of clarifying actions, such as producing a prototype and presenting it to the customer for feedback:

In terms of KSAs then, this cycle requires practitioners to:

- Evaluate the trigger event. This could involve simply recognising a trigger, such as a news story that in some way affects the company, or a scheduled review of a planned artifact, such as a design document.
- Gather information and build up the knowledge necessary to understand what is required of the target artifact. Sufficient information may be available from existing artifacts, but it might be necessary to create intermediate artifacts - such as prototypes - to obtain clarification.
- Design the artifact. This requires the creative, high-level skills needed to devise new concepts. Also required here are the decision-making and negotiation skills mainly associated with management but are required when investigating alternative courses of action. For instance, detailed analysis of a particular requirement may show that the initial estimates were too optimistic. In this case, the design part of the cycle will have to conceive a solution to satisfy the requirements as well as calculating revised schedule estimates. The update to the project plan is triggered by a report from the programmer.
- Produce the artifact. These KSAs supplement the creative skills and include technical writing, prototyping, coding, testing, proof-reading, presentation and reporting.

Another way of looking at the lifecycle of an artifact is to consider it in terms of knowledge management. Demarest [16] proposes a process where knowledge is embodied in the form of an artifact and disseminated throughout the organization. Although this process is not cyclical like the previous models, it is suggested that the use of an artifact will generate new knowledge that, in turn, needs to be organized and embodied into a new sort of artifact.

Thus the trigger event is the dissemination of some sort of knowledge and this is used to analyze the problem of creating the next artifact. The analysis and design of the new artifact is termed construction, which is defined by Demarest as "the process of discovering or structuring a kind of knowledge: how to sell a particular product to a particular market, for example, or how to diagnose a particular kind of customer problem" (p.376). The actual creation of the artifact maps onto embodiment.

Table 1. Relating the Models

Knowledge Management (Demarest)	Artifact Dynamics	Learning Cycle (Lewin)
Dissemination	Trigger Event	Concrete Experience
Use	Problem Analysis	Observations and Reflections
Construction		
	Artifact Design	Abstract Conceptualization
Embodiment	Artifact Creation	Active Experimentation

To summarize, these models illustrate the dynamic nature of artifacts. They begin after a trigger event, which may well be the dissemination of a previous artifact. The goals of the new artifact must be tied down and all necessary information gathered before it can be designed. The design is conceived and then articulated, or embodied, in the form of a tangible artifact. This artifact, in turn, must be disseminated and used to make decisions or base further artifacts on.

Thus, from a knowledge, skills and abilities perspective, an artifact should not be considered merely in terms of the skills needed to design it, but also in terms of the knowledge and the other artifacts that must be acquired before any sensible synthesis can take place. Having created, or embodied, the artifact, it must be made available to others on the project team and reviewed by them. Therefore, a single artifact draws on research, analysis, design, implementation and evaluation skills.

3.2 Relating Artifacts to Organizational Functions

The practitioner interviews have shown that it is the organization and the individuals concerned who dictate which person works on what artifacts. Although it is not possible to formulate rules relating artifacts to individuals, a level of understanding is possible if attention is given to the organizational functions that are related via artifacts.

Judging by the interview data, it is useful to consider the corporate structure as a set of overlapping functions. It is also clear from these data that the overlap between functions is accommodated by means of artifacts. Indeed, the interface between the company and its external customers is also facilitated through artifacts. This suggests that the bulk of the artifacts reside in overlapping areas, signifying that the software development process is truly a multi-functional team effort.

The term 'organizational function' has been chosen deliberately rather than department, as each company may have different departmental structures. Thus, for the purposes of this study, the organizational functions will be defined as:

1. **Management.** This includes project management and senior management. It also covers the finance and legal departments. Essentially, it is where the major decision-making activities take place.
2. **Front-end customer interface.** Here we find the product managers and the sales and marketing personnel. These people interact directly with the customer and with the marketplace.
3. **Back-end customer interface.** This is where the product is installed in the customer site. The installation and the subsequent maintenance of the product are under the control of the customer-support function.
4. **Development.** This area includes all personnel who contribute directly to the creation of the delivered product. That is: programmers, testers, technical writers and trainers.

Locating artifacts within the organization is achieved by studying the interview data and determining which actors are involved with each artifact. A person may be associated with a particular artifact if s/he designs and creates it, contributes data to it, reviews it or makes use of it to create another artifact or to make a decision.

To illustrate the process, the four principal artifacts identified in Figure 1 are analyzed:

- **Engineering Requirements.** This artifact is the responsibility of the systems architect, who is part of the development function. S/he is assisted by programmers, who are also in this function. The trigger for the engineering requirements is the marketing requirements document, which is produced by a product manager. This establishes an overlap between development and the front-end customer interface. Because the engineering requirements document informs the project plan, the management function is involved. Finally, because the requirements might contain aspects that affect the external interfaces of the product, the customer-support personnel need to review the document. This creates an overlap with the back-end customer interface. These relationships place the engineering requirements artifact firmly in the intersection of all the organizational functions.
- **Feasibility Report**. As this is the responsibility of product management, it involves the front-end customer interface function. As time and headcount estimates and technical feasibility input come from the architects and programmers, it overlaps with the development function. Its purpose is to provide the basis for the go/no-go decision, made by the management function. There is no evidence in this study that the back-end customer interface is involved.
- **Project Plan.** This is the responsibility of the project manager in the management function. It depends for its time and headcount estimates and details of the work breakdown structure on both the development and back-end functions. It is also of concern to product managers,

who need to be able to relay to potential customers what features are currently under development and when they are likely to become available. These overlaps place the project plan in the center of all organizational functions.

- **Installed Product.** The installed product relates the back-end customer interface function with the customers themselves, providing a boundary with the outside world. It seems surprising that the development function is not involved with this artifact, as programmers and testers often accompany the installation team. However, the installed product is literally that which is installed in the customer site. It is the end product of the development lifecycle and is being used by the customer. If there are problems with the product, the bug report/code update mechanism is employed.

Table 2. Locating the Principal Artifacts

	Feasibility Report	Engineering Reqs	Project Plan	Installed Product
Front-End	x	x	x	
Management	x	x	x	
Development	x	x	x	
Back-End	x	x	x	x
Outside World				x

As can be seen from Table 2, the principal artifacts are all boundary objects. Study of the principal artifacts also shows how some of them contain data that is needed for decisions – such as the go/no-go decision – to be made. If knowledge, skills and abilities are now considered in terms of artifacts, it is likely that each artifact will require certain communications, collaboration and decision-support skills. However, before exploring this possibility, the concept of an artifact needs to be examined further.

4. Relating KSAs to artifacts

Considering KSAs in terms of the four phases of an artifact's lifecycle brings the team into focus. This is a useful development as the three factors influencing team performance are: the task, the team and the individual [17, 18]. Team focus is achieved by highlighting the communication, collaboration and decision support skills intrinsic in the lifecycle of an artifact.

1. **Communication.** Obviously, information is conveyed via artifacts. This is particularly the case when the parties involved are not co-located. It should be noted that not all artifacts are produced by the development team. These artifacts may be produced by other organizations (such as industry regulators) or by other divisions within the same company (such as the finance and purchasing departments).

2. **Collaboration.** An artifact is not something that is created by a project stakeholder and never seen by anyone else. At its simplest, it is created by one party and acted upon by another. For more sophisticated artifacts, a myriad of people, from many different organizational functions, may be involved in its creation. Their product is then reviewed by interested parties and the insights from the review go to generating an improved version of the artifact.

3. **Decision Support.** The management team needs to decide on what courses of action to take. They must decide what features are worth investigating further, what features should comprise a release and what actions to take if a project is running behind schedule. Each of these decisions is informed by artifacts – feasibility studies and progress reports in these examples.

It should be noted that technical skills are still required and make up a fourth category in this new taxonomy. Each of these categories plays a critical role in particular phases. Because each artifact has a similar lifecycle, it is not surprising that each of the interviewees has a broadly similar skill set. While specific technical skills are role specific, the communication, collaboration and decision-support skills are required to deal with any artifact in a team environment.

Table 3 highlights the way the collaboration and decision-support skills are so important for dealing with trigger events and the gathering of data in the analysis stage. For instance, a noticeable feature of the interviews is the way teams are formed on an ad-hoc basis, often for the sole purpose of creating a single artifact, such as a feasibility report. The technical skills, those cited by the senior programmers and the systems architects as the most rewarding, are really only called for to synthesize and embody solutions. This suggests that the emphasis that is currently placed on technology-based skills is not equipping team members with the range of knowledge and skills/abilities to deal with the entire artifact lifecycle.

Table 3. Relating KSAs to Artifact Phases

	Artifact Trigger	Artifact Analysis	Artifact Design	Artifact Creation
Communication				
Reading	x	x		
Listening	x	x		
Comprehension	x	x		
Writing				x
Presentation	x			x
Reporting				x
Collaboration				
Evaluation				
Document Review		x		
Code Inspection		x		
Problem Analysis		x		
Management				
Coordination	x	x	x	x
Teamwork	x	x	x	x
Team Leading	x	x	x	x
Negotiation	x	x	x	x
Decision Support				
Management				
Decision Making	x	x	x	x
Coping with Ambiguity		x	x	
Prioritization		x	x	
Risk Identification		x	x	
Planning		x	x	x
Risk Management		x	x	
Business				
Estimation		x		
Budgeting		x		
Purchasing		x		
Logistics		x		
Technical				
Design			x	
Prototyping			x	
Coding			x	
Testing				x
Proof-reading				x

5. Conclusion

The analysis in this paper is motivated by the difficulties encountered in defining individual roles. To create a framework that addresses the problem, two relationships need to be considered: how artifacts are located in the organizational context and how knowledge, skills and abilities (KSAs) are related to phases of the artifacts' lifecycles. While this arrangement does not allow KSAs to be mapped to roles, it does provide a means of identifying the KSAs needed to develop software and the organizational functions that are involved in the effort.

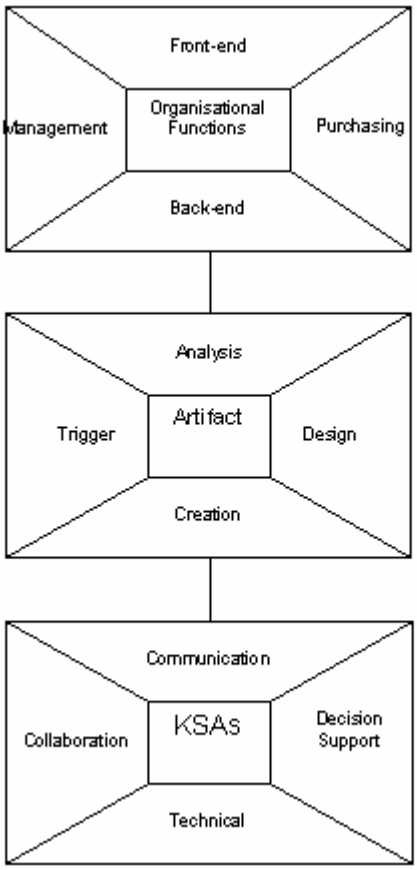


Figure 3. The Artifact-centric Framework

The lifecycle of an artifact is central to this framework. Artifacts are triggered; their goals are set and relevant data gathered; their contents are designed and embodied in artifact form. It is not surprising that most of the artifacts identified in this study are boundary objects, facilitating communication and collaboration between organizational functions and between the organization and the outside world. They also serve to support the decision-making process. These communications, collaboration and decision-support skills are generic to artifacts while the technical skills are more artifact-specific. The framework caters for this difference by allowing each artifact to be considered in turn.

The artifact-centric framework has produced a new taxonomy of knowledge and skills/abilities. MIS skills surveys [19-23] classify KSAs as technical, business, inter-personal and the

application of technology to business. The artifact-centric taxonomy instead incorporates communication, collaboration, decision-support and technical skills, along with the different types of knowledge – technical, product, domain, market, political and commercial. This new taxonomy appears to place its emphasis on team-working skills. However, while being able to inter-work well with other team members is important, the inter-working skills serve as conduits for the technical skills each member brings to the team. Without technical skills, a person would not be able to contribute to the project.

The question of what knowledge, skills and abilities (KSAs) are needed to develop software is an important one for many stakeholders, particularly in the context of globally distributed software development: Educators need to know what skills their graduates will need. Employers need to assess the skills of the people they recruit, and managers need to assess the skills required for new projects. Practitioners are interested in gaining and improving necessary, sought after skills. Professional bodies will want to develop their members' careers so that they tailor their skills to the areas in most demand. Government agencies want to draw attention to the skills available in their respective countries, in order to attract inward investment.

The artifact-centric framework is a theory grounded in the experiences of four Irish telecommunications software teams. It is a descriptive theory in that it seeks to describe what KSAs are actually needed to develop software. Further research is needed, applying the framework in different development contexts, for example, in different application domains, such as automotive software development or using different development methods, such as the Rational Unified Process (RUP), which is based on a specific set of artifact types, or agile development which relies on fewer and less formal artifacts.

6. REFERENCES

[1] P. Kruchten, "Rational Unified Process - An Introduction," Addison-Wesley, Boston, 1999.

[2] M. Jarke and K. Pohl, *Software Engineering Journal*, **9**, 257 - 266 (1994)

[3] J. Downey, Systems Architect and Systems Analyst: Are These Comparible Roles?, K. Kaiser and T. Ryan, Eds., 2006 ACM SIGMIS CPR Conference, Claremont, California, ACM Press, 2006, p. 213-220

[4] Office for National Statistics, "Standard Occupational Classification - Volume 2," The Publishing House, London, ed. v6, 2000b.

[5] Forfas (2003) Responding to Ireland's Skills Needs, The Fourth Report of the Expert Group on Future Skills Needs. Dublin, Forfas, pp. 24-33, 86-94.

[6] Enterprise Strategy Group (2004) Ahead of the Curve: Ireland's Place in the Global Economy. Dublin, Forfas.

[7] Irish Computer Society (2006) Irish Computer Society Website, vol. 2006, pp. http://www.ics.ie.

[8] SFIA Foundation (2006) Skills Framework for an Information Age, vol. 2006, pp. http://www.sfia.org.uk.

[9] J. Downey, A Framework to Elicit the Skills Needed for Software Development, J. E. Moore and S. E. Yager, Eds., 2005 ACM SIGMIS CPR Conference, Atlanta, Georgia, ACM Press, 2005, p. 122-127

[10] M. Norris and P. Rigby, "Software Engineering Explained," John Wiley & Sons, Chichester, England, 1992.

[11] M. M. Cluts, The Evolution of Artifacts in Cooperative Work: Constructing Meaning Through Activity, ACM SIGGROUP 2003, Florida, USA, ACM Press, 2003, p. 144-152

[12] M. Morisio, I. Stamelos and A. Tsoukias, A New Method to Evaluate Software Artifacts Against Predefined Profiles, ACM SIGSSEKE, Ischia, Italy, ACM Press, 2002, p. 811-818

[13] P. Mambrey and M. Robinson, Understanding the Role of Documents in a Hierarchical Flow of Work, ACM SIGGROUP 1997, Phoenix, Arizona, ACM Press, 1997, p. 119-127

[14] D. A. Kolb, "Experiential Learning: Experience as the Source of Learning and Development," Prentice Hall, Englewood Cliffs, New Jersey, 1984.

[15] K. Lewin, "Field Theory in Social Science: Selected Theoretical Papers," Tavistock Publications, New York, 1952.

[16] M. Demarest, *Journal of Long Range Planning*, **30**, 374-384 (1997)

[17] J. Adair, "The Inspirational Leader: How to Motivate, Encourage and Achieve Success," Kogan Page Ltd, London, 2003.

[18] J. Adair, "The John Adair Handbook of Management and Leadership," Thorogood Publishing Ltd, London, 2004.

[19] D. M. S. Lee, E. M. Trauth and D. Farwell, *MIS Quarterly*, **17**, 313-340 (1995)

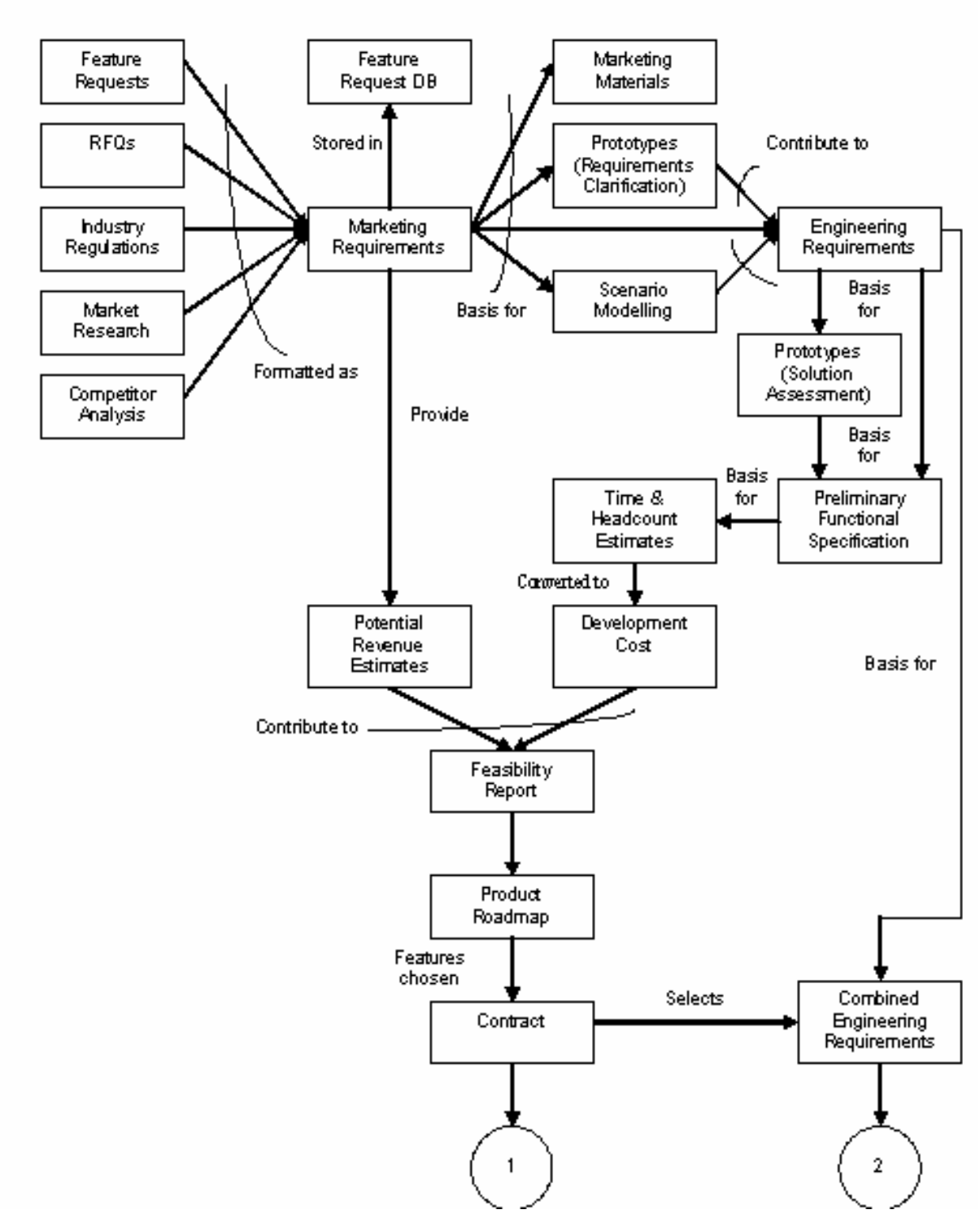
[20] S. Sawyer, K. R. Eschenfelder, A. Diekema and C. R. McClure, *ACM SIGCPR Computer Personnel*, **19**, 27-41 (1998)

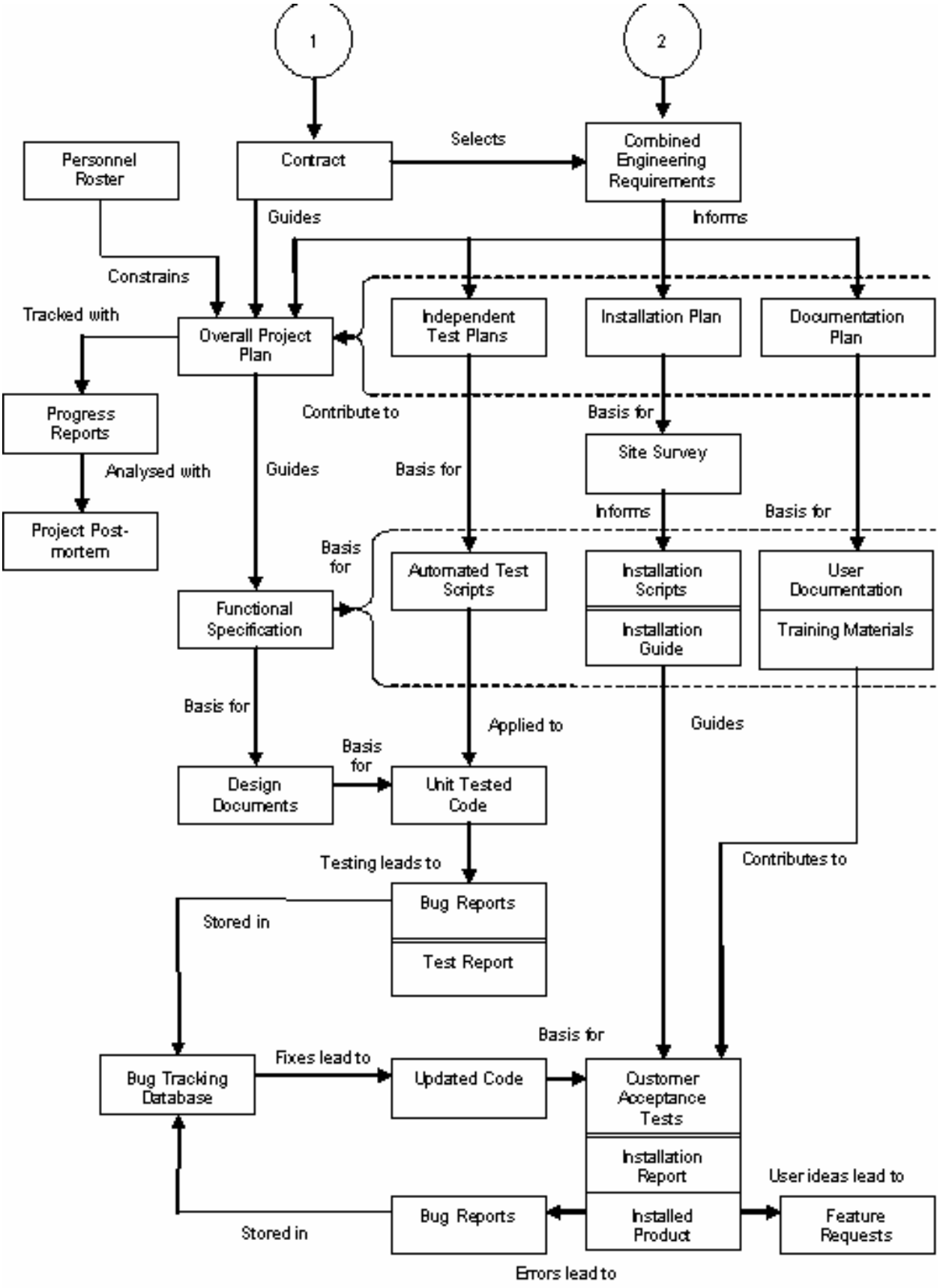
[21] N. Shi and D. Bennett, *ACM SIGCPR Computer Personnel*, **19**, 3-19 (1998)

[22] E. M. Trauth, D. W. Farwell and D. Lee, *MIS Quarterly*, **17**, 293-307 (1993)

[23] C. L. Noll and M. Wilkins, *Journal of Information Technology Education*, **1**, 143-154 (2002)

7. Appendix 1: All Identified Artifacts




1
2
Personnel Roster
Contract
Selects
Combined Engineering Requirements
Guides
Informs
Constrains
Tracked with
Overall Project Plan
Independent Test Plans
Installation Plan
Documentation Plan
Progress Reports
Contribute to
Basis for
Site Survey
Analysed with
Guides
Basis for
Project Post-mortem
Informs
Basis for
Basis for
Automated Test Scripts
Installation Scripts
Installation Guide
User Documentation
Training Materials
Functional Specification
Basis for
Applied to
Guides
Basis for
Design Documents
Unit Tested Code
Testing leads to
Contributes to
Bug Reports
Test Report
Stored in
Basis for
Fixes lead to
Bug Tracking Database
Updated Code
Customer Acceptance Tests
Installation Report
User ideas lead to
Bug Reports
Installed Product
Feature Requests
Stored in
Errors lead to